LilyPond

Le système de gravure musicale

Manuel de notation

L'équipe de développement de LilyPond

Ce document constitue le manuel de notation de GNU LilyPond 2.27.4. Sa lecture requiert une familiarité avec le contenu présenté dans le *Manuel d'initiation*.

Pour connaître la place qu'occupe ce manuel dans la documentation, consultez la page Section "Manuels" dans *Informations générales*.

Si vous ne disposez pas de certains manuels, la documentation complète se trouve sur `https://lilypond.org/`.

Copyright © 1998–2026 par les auteurs.

The translation of the following copyright notice is provided for courtesy to non-English speakers, but only the notice in English legally counts.

La traduction de la notice de droits d'auteur ci-dessous vise à faciliter sa compréhension par le lecteur non anglophone, mais seule la notice en anglais a valeur légale.

> Vous avez le droit de copier, distribuer et/ou modifier ce document selon les termes de la Licence GNU de documentation libre, version 1.1 ou tout autre version ultérieure publiée par la Free Software Foundation, "sans aucune section invariante". Une copie de la licence est fournie à la section "Licence GNU de documentation libre".
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections. A copy of the license is included in the section entitled "GNU Free Documentation License".

Pour LilyPond version 2.27.4

Table des matières

Notation musicale générale

1 Hauteurs ... 3

1.1 Écriture des hauteurs de note ... 3
1.1.1 Hauteurs avec octave absolue ... 3
1.1.2 Octaves relatives ... 4
1.1.3 Altérations ... 8
1.1.4 Hauteurs approximatives ... 10
1.1.5 Nom des notes dans d'autres langues ... 11
1.2 Modification de plusieurs hauteurs ... 13
1.2.1 Vérifications d'octave ... 13
1.2.2 Transposition ... 14
1.2.3 Inversion ... 17
1.2.4 Rétrogradation ... 17
1.2.5 Transformations modales ... 18
1.3 Gravure des hauteurs ... 20
1.3.1 Clefs ... 20
1.3.2 Armure ... 25
1.3.3 Marques d'octaviation ... 27
1.3.4 Instruments transpositeurs ... 30
1.3.5 Altérations accidentelles automatiques ... 32
1.3.6 Glyphes d'altération alternatifs ... 40
1.3.7 Ambitus ... 41
1.4 Têtes de note ... 45
1.4.1 Têtes de note spécifiques ... 45
1.4.2 Têtes de note avec nom de note ... 46
1.4.3 Têtes de note à forme variable ... 48
1.4.4 Improvisation ... 52

2 Rythme ... 53

2.1 Écriture du rythme ... 53
2.1.1 Durées ... 53
2.1.2 N-olets ... 56
2.1.3 Changement d'échelle des durées ... 62
2.1.4 Liaisons de prolongation ... 64
2.2 Écriture des silences ... 68
2.2.1 Silences ... 69
2.2.2 Silences invisibles ... 71
2.2.3 Silences valant une mesure ... 73
2.2.4 Césures ... 77
2.3 Gravure du rythme ... 80
2.3.1 Mesures ... 80
2.3.2 Métrique ... 81
2.3.3 Indication métronomique ... 88
2.3.4 Levées ... 93
2.3.5 Musique sans métrique ... 94
2.3.6 Notation polymétrique ... 96
2.3.7 Découpage automatique des notes ... 100

2.3.8 Gravure de lignes rythmiques ... 101
2.4 Barres de ligature ... 104
2.4.1 Barres de ligature automatiques ... 104
2.4.2 Définition des règles de ligature automatique ... 107
2.4.3 Barres de ligature manuelles ... 117
2.4.4 Liens de croches en soufflet ... 121
2.4.5 Ligatures barrées ... 122
2.5 Mesures ... 123
2.5.1 Barres de mesure ... 123
2.5.2 Barres de mesure automatiques ... 133
2.5.3 Numéros de mesure ... 135
2.5.4 Vérification des limites et numéros de mesure ... 144
2.5.5 Indications de repère ... 144
2.5.6 Compteurs de mesures ... 147
2.5.7 Division en sections ... 150
2.6 Fonctionnalités rythmiques particulières ... 150
2.6.1 Notes d'ornement ... 150
2.6.2 Alignement et cadences ... 156

3 Signes d'interprétation ... 158
3.1 Signes d'interprétation attachés à des notes ... 158
3.1.1 Articulations et ornements ... 158
3.1.2 Nuances ... 162
3.1.3 Personnalisation des indications de nuance ... 171
3.2 Signes d'interprétation sous forme de courbe ... 173
3.2.1 Liaisons d'articulation ... 174
3.2.2 Liaisons de phrasé ... 177
3.2.3 Signes de respiration ... 179
3.2.4 Chutes et sauts ... 180
3.3 Signes d'interprétation sous forme de ligne ... 181
3.3.1 Glissando ... 181
3.3.2 Arpèges ... 186
3.3.3 Trilles ... 189

4 Répétitions et reprises ... 193
4.1 Répétition d'un long passage ... 193
4.1.1 Répétitions explicites ... 193
4.1.2 Répétitions simples ... 194
4.1.3 Fins alternatives ... 195
4.1.4 Autres aspects des fragments répétés ... 197
4.1.5 Reprises *Al-fine* ... 199
4.1.6 Structure d'une reprise *Segno* ... 200
4.1.7 Apparence d'une reprise *Segno* ... 204
4.1.8 Indications de reprise manuelles ... 210
4.2 Autres types de répétition ... 213
4.2.1 Répétitions de mesure ... 213
4.2.2 Répétitions en trémolo ... 217

5 Notes simultanées 221

5.1 Monophonie 221
5.1.1 Notes en accords 221
5.1.2 Répétition d'accords 223
5.1.3 Expressions simultanées 225
5.1.4 Clusters 226
5.2 Plusieurs voix 227
5.2.1 Polyphonie sur une portée 227
5.2.2 Styles de voix 232
5.2.3 Résolution des collisions 233
5.2.4 Fusion de silences 238
5.2.5 Regroupement automatique de parties 238
5.2.6 Saisie de musique en parallèle 244

6 Notation sur la portée 248

6.1 Gravure des portées 248
6.1.1 Initialisation de nouvelles portées 248
6.1.2 Regroupement de portées 250
6.1.3 Imbrication de regroupements de portées 254
6.1.4 Séparation des systèmes 257
6.2 Modification de portées individuelles 258
6.2.1 Symbole de la portée 258
6.2.2 Portées d'ossia 262
6.2.3 Masquage de portées 265
6.3 Écriture de parties séparées 270
6.3.1 Noms d'instrument 270
6.3.2 Citation d'autres voix 274
6.3.3 Mise en forme d'une citation 277
6.3.4 Compression de mesures vides 282

7 Annotations éditoriales 286

7.1 Dans la portée 286
7.1.1 Indication de la taille de fonte musicale 286
7.1.2 Doigtés 290
7.1.3 Glissement de doigt 293
7.1.4 Dictée à trous 297
7.1.5 Coloration d'objets 298
7.1.6 Surlignage de portées 300
7.1.7 Crochets de matériau optionel 304
7.1.8 Parenthèses 304
7.1.9 Hampes 306
7.2 Hors de la portée 307
7.2.1 Nom des notes 307
7.2.2 Info-bulle 309
7.2.3 Quadrillage temporel 311
7.2.4 Crochets d'analyse 313

8 Texte 317

8.1 Ajout de texte 317
8.1.1 Vue d'ensemble des objets textuels 317

8.1.2 Commentaires textuels ... 320
8.1.3 Indication textuelle avec extension ... 321
8.1.4 Marque de section ... 323
8.1.5 Indications textuelles ... 323
8.1.6 Texte indépendant ... 328
8.2 Mise en forme du texte ... 329
8.2.1 Introduction au formatage de texte ... 329
8.2.2 Sélection de la fonte et de la taille ... 333
8.2.3 Alignement du texte ... 337
8.2.4 Éléments graphiques dans du texte formaté ... 342
8.2.5 Notation musicale dans du texte formaté ... 345
8.3 Fontes ... 347
8.3.1 Formats de fontes non pris en charge ... 347
8.3.2 Localisation des fontes ... 348
8.3.3 Familles de fontes ... 348
8.3.4 Fonctionnalités des fontes ... 350
8.3.5 Choix des fontes ... 351

Notation spécialisée

9 Musique vocale ... 355
9.1 Vue d'ensemble de la musique vocale ... 355
9.1.1 Références en matière de musique vocale ... 355
9.1.2 Saisie des paroles ... 356
9.1.3 Alignement des paroles sur la mélodie ... 357
9.1.4 Durée automatique des syllabes ... 359
9.1.5 Durée explicite des syllabes ... 362
9.1.6 Plusieurs syllabes sur une note ... 363
9.1.7 Plusieurs notes pour une même syllabe ... 364
9.1.8 Traits d'union et de prolongation ... 367
9.1.9 Changement graduel de voyelle ... 368
9.2 Situations particulières en matière de paroles ... 368
9.2.1 Travail avec des paroles et variables ... 368
9.2.2 Positionnement vertical des paroles ... 370
9.2.3 Positionnement horizontal des syllabes ... 374
9.2.4 Paroles et reprises ... 377
9.2.5 Paroles alternatives ... 385
9.2.6 Polyphonie et paroles communes ... 386
9.3 Couplets ... 388
9.3.1 Numérotation des couplets ... 388
9.3.2 Rappel de couplet ... 389
9.3.3 Indication de nuance dans les couplets ... 392
9.3.4 Indication du personnage et couplets ... 393
9.3.5 Rythme différent selon le couplet ... 393
9.3.6 Paroles en fin de partition ... 396
9.3.7 Paroles sur plusieurs colonnes en fin de partition ... 398
9.4 Chansons ... 400
9.4.1 Références en matière de chanson ... 400
9.4.2 Feuille de chant ... 400
9.5 Chorale ... 401

9.5.1 Références en matière de chorale 401
9.5.2 Mise en forme d'une partition chorale 402
9.6 Opéras et musiques de scène 404
9.6.1 Références en matière d'opéra et musique de scène 404
9.6.2 Indication du rôle 405
9.6.3 Citation-repère 407
9.6.4 Musique parlée 411
9.6.5 Dialogue et musique 411
9.7 Chants liturgiques 413
9.7.1 Références en matière de chant liturgique 413
9.7.2 Cantiques et hymnes 413
9.7.3 Psalmodie 419
9.7.4 Barre de phrase en musique liturgique 421
9.7.5 Mesure incomplète et musique liturgique 422
9.8 Musique vocale ancienne 425

10 Instruments utilisant des portées multiples 426
10.1 Vue d'ensemble des claviers 426
10.1.1 Généralités sur les instruments à clavier 427
10.1.2 Changement de portée manuel 428
10.1.3 Changement de portée automatique 430
10.1.4 Lignes de changement de portée 432
10.2 Piano 434
10.2.1 Pédales de piano 435
10.3 Orgue 436
10.3.1 Indication de pédalier d'orgue 436
10.4 Accordéon 437
10.4.1 Symboles de jeux 437
10.5 Harpe 438
10.5.1 Généralités sur la harpe 438
10.5.2 Pédales de harpe 438

11 Cordes non frettées 440
11.1 Vue d'ensemble de la notation pour cordes non frettées 440
11.1.1 Références en matière de cordes non frettées 440
11.1.2 Indications d'archet 441
11.1.3 Harmoniques 441
11.1.4 Snap (Bartók) pizzicato 443

12 Instruments à cordes frettées 444
12.1 Vue d'ensemble des cordes frettées 444
12.1.1 Références en matière de cordes frettées 444
12.1.2 Indications du numéro de corde 445
12.1.3 Tablatures par défaut 446
12.1.4 Tablatures personnalisées 465
12.1.5 Tablatures sous forme d'étiquettes 469
12.1.6 Tablatures prédéfinies 479
12.1.7 Tablatures automatiques 490
12.1.8 Doigtés pour la main droite 493
12.2 Guitare 494

12.2.1 Indication de la position et du barré 494
12.2.2 Indication des harmoniques et notes étouffées 495
12.2.3 Indication de *power chord* 497
12.3 Banjo 498
12.3.1 Tablatures pour banjo 498
12.4 Luth 499
12.4.1 Tablatures pour luth 499

13 Percussions 500
13.1 Vue d'ensemble des percussions 500
13.1.1 Références en matière de notation pour percussions 500
13.1.2 Notation de base pour percussions 500
13.1.3 Roulements de tambour 501
13.1.4 Hauteurs en percussions 502
13.1.5 Portées de percussion 502
13.1.6 Personnalisation de portées de percussion 505
13.1.7 Notes fantômes 507

14 Instruments à vent 509
14.1 Vue d'ensemble des instruments à vent 509
14.1.1 Références en matière d'instruments à vent 509
14.1.2 Doigtés pour vents 510
14.2 Cornemuse 512
14.2.1 Définitions pour la cornemuse 512
14.2.2 Exemple pour la cornemuse 513
14.3 Bois 514
14.3.1 Diagrammes pour bois 514

15 Notation des accords 522
15.1 Mode accords 522
15.1.1 Généralités sur le mode accords 522
15.1.2 Accords courants 523
15.1.3 Extension et altération d'accords 524
15.1.4 Renversement d'accord et vocification spécifique 527
15.2 Gravure des accords 527
15.2.1 Impression des noms d'accord 527
15.2.2 Personnalisation des noms d'accord 531
15.2.3 Grilles harmoniques 536
15.3 Basse chiffrée 540
15.3.1 Introduction à la basse chiffrée 540
15.3.2 Saisie de la basse chiffrée 541
15.3.3 Gravure de la basse chiffrée 545

16 Musique contemporaine 549
16.1 Hauteur et harmonie en musique contemporaine 549
16.1.1 Généralités en matière de hauteur et d'harmonie 549
16.1.2 Notation microtonale 549
16.1.3 Armures contemporaines et harmonie 549
16.2 Approches du rythme en musique contemporaine 549
16.2.1 Généralités sur le rythme en musique contemporaine 549

16.2.2 N-olets et musique contemporaine . . . 549
16.2.3 Métriques contemporaines . . . 549
16.2.4 Notation polymétrique étendue . . . 549
16.2.5 Ligatures et musique contemporaine . . . 549
16.2.6 Barres de mesure et musique contemporaine . . . 549
16.3 Notation graphique . . . 549
16.4 Techniques de partition contemporaine . . . 551
16.5 Nouvelles techniques instrumentales . . . 551
16.6 Informations complémentaires et exemples pertinents . . . 551
16.6.1 Ouvrages et articles sur la notation en musique contemporaine . . . 551
16.6.2 Partitions et exemples . . . 551

17 Notations anciennes . . . 552
17.1 Formes de notation ancienne prises en charge . . . 553
17.2 Considérations communes aux musiques anciennes . . . 554
17.2.1 Contextes prédéfinis . . . 554
17.2.2 Ligatures . . . 554
17.2.3 Guidons . . . 555
17.3 Typographie de musique ancienne . . . 556
17.3.1 Contextes de musique mensurale . . . 556
17.3.2 Clefs anciennes . . . 557
17.3.3 Métriques anciennes . . . 558
17.3.4 Têtes de note anciennes . . . 558
17.3.5 Crochets anciens . . . 559
17.3.6 Silences anciens . . . 560
17.3.7 Altérations et armures anciennes . . . 561
17.3.8 Altérations suggérées (*musica ficta*) . . . 561
17.3.9 Aperçu historique des ligatures . . . 562
17.3.10 Ligatures mensurales . . . 563
17.3.11 Lecture des ligatures mensurales blanches . . . 566
17.4 Typographie du chant grégorien . . . 569
17.4.1 Contextes du chant grégorien . . . 569
17.4.2 Clefs grégoriennes . . . 570
17.4.3 Altérations et armures grégoriennes . . . 571
17.4.4 Divisions . . . 571
17.4.5 Articulations grégoriennes . . . 572
17.4.6 Points d'augmentation (*morae*) . . . 573
17.4.7 Neumes et ligatures grégoriennes . . . 573
17.5 Typographie de notation kiévienne . . . 579
17.5.1 Contextes de notation kiévienne . . . 579
17.5.2 Clefs kiéviennes . . . 580
17.5.3 Notes kiéviennes . . . 580
17.5.4 Altérations kiéviennes . . . 581
17.5.5 Barres de mesure kiéviennes . . . 581
17.5.6 Mélismes kiéviens . . . 581
17.6 Réédition de musique ancienne . . . 582
17.6.1 Des incipits . . . 583
17.6.2 Mise en forme de la musique mensurale . . . 583
17.6.3 Transcription de chant grégorien . . . 584
17.6.4 Éditions ancienne et moderne à partir d'une même source . . . 587

18 Musiques du monde ... 591
18.1 Noms des notes et altérations non-occidentaux ... 591
18.1.1 Extension des systèmes de notation et d'accordage ... 591
18.2 Musique arabe ... 591
18.2.1 Références pour la musique arabe ... 591
18.2.2 Noms des notes en arabe ... 592
18.2.3 Armures arabes ... 593
18.2.4 Métriques arabes ... 595
18.2.5 Exemple de musique arabe ... 596
18.2.6 Lectures complémentaires pour la musique arabe ... 597
18.3 Musique classique turque ... 597
18.3.1 Références pour la musique classique turque ... 597
18.3.2 Noms de note en turc ... 598
18.3.3 Armures turques ... 598
18.3.4 Lectures complémentaires pour la musique turque ... 599
18.4 Musique classique perse ... 599
18.4.1 Notation de la musique perse ... 599
18.4.2 Tonalités perses ... 600
18.4.3 Armures perses ... 601
18.4.4 Lectures complémentaires pour la musique perse ... 601

Généralités en matière d'entrée et sortie

19 Modes de saisie ... 605

20 Agencement du code ... 608
20.1 Structure d'une partition ... 608
20.2 Plusieurs partitions dans un même ouvrage ... 609
20.3 Plusieurs éditions pour une même source ... 611
20.4 Nom des fichiers de sortie ... 611
20.5 Structure de fichier ... 612

21 Titres et entêtes ... 615
21.1 Création de titres et entête ou pied de page ... 615
21.1.1 Généralités en matière de titrages ... 615
21.1.2 Mise en forme par défaut des titrages subalternes ... 618
21.1.3 Mise en forme par défaut des entête et pied de page ... 622
21.2 Titrages personnalisés ... 623
21.2.1 Mise en forme personnalisée des champs de titrage ... 623
21.2.2 Mise en forme personnalisée des titrages ... 624
21.2.3 Mise en forme personnalisée des entête et pied de page ... 627
21.3 Création des métadonnées des fichiers de sortie ... 628
21.4 Notes de bas de page ... 629
21.4.1 Notes de bas de page dans une expression musicale ... 629
21.4.2 Notes de bas de page dans du texte indépendant ... 635
21.5 Notes en ligne ... 638
21.6 Référencement des numéros de page ... 639
21.7 Table des matières ... 641

22 Travail sur des fichiers texte ... 644
22.1 Insertion de fichiers LilyPond ... 644
22.2 Différentes éditions à partir d'une même source ... 645
22.2.1 Utilisation de variables ... 645
22.2.2 Utilisation de balises ... 647
22.2.3 Globalisation des réglages ... 658
22.3 Utilisation de fonctions musicales ... 658
22.3.1 Syntaxe d'une fonction de substitution ... 658
22.3.2 Exemples de fonction de substitution ... 659
22.3.3 Comment éviter le partage d'expressions musicales ... 661
22.3.4 Fonctions de subsitution et octave relative ... 662
22.4 Caractères spéciaux ... 664
22.4.1 Codage du texte ... 664
22.4.2 Unicode ... 664
22.4.3 Équivalents ASCII ... 665

23 Contrôle des sorties ... 667
23.1 Extraction de fragments musicaux ... 667
23.2 Ignorer des passages de la partition ... 667
23.3 Formats de sortie alternatifs ... 668
23.3.1 Sortie SVG ... 668
23.4 Intégration de fichiers à la sortie PDF ... 669
23.5 Changement des fontes musicales ... 669

24 Génération de fichiers MIDI ... 672
24.1 Notation prise en compte dans le MIDI ... 672
24.2 Notation non prise en compte dans le MIDI ... 673
24.3 Le bloc MIDI ... 673
24.4 Gestion des nuances en MIDI ... 674
24.4.1 Indication des nuances en MIDI ... 674
24.4.2 Réglage du volume en MIDI ... 675
24.4.3 Réglage de propriétés dans le bloc MIDI ... 677
24.5 Gestion des instruments MIDI ... 678
24.6 Gestion des répétitions en MIDI ... 679
24.7 Affectation des canaux MIDI ... 680
24.8 Propriétés de contextes et effets MIDI ... 682
24.9 Amélioration du rendu MIDI ... 683
24.9.1 Le script `articulate` ... 683
24.9.2 Le script `swing` ... 684

25 Extraction d'informations musicales ... 686
25.1 Affichage de notation au format LilyPond ... 686
25.2 Affichage de la musique sous forme d'expression Scheme ... 686
25.3 Enregistrement d'événements musicaux dans un fichier ... 686

Gestion de l'espace

26 Mise en forme de la page ... 691

26.1 Le bloc \paper ... 691
26.2 Format du papier et adaptation automatique ... 693
26.2.1 Format du papier ... 693
26.2.2 Adaptation automatique au format ... 694
26.3 Variables d'espacement vertical fixe ... 694
26.4 Variables d'espacement vertical fluctuant ... 695
26.4.1 Structure des variables d'espacement vertical fluctuant ... 695
26.4.2 Liste des variables d'espacement fluctuant ... 696
26.5 Variables d'espacement horizontal ... 697
26.5.1 Variables de marge et de largeur ... 697
26.5.2 Variables spécifiques à l'impression recto-verso ... 699
26.5.3 Variables d'indentation et de décalage ... 699
26.6 Autres variables du bloc \paper ... 700
26.6.1 Variables de gestion des sauts de ligne ... 700
26.6.2 Variables de gestion des sauts de page ... 700
26.6.3 Variables de gestion des numéros de page ... 701
26.6.4 Variables supplémentaires d'en-tête et *markup* ... 702
26.6.5 Variables de débogage ... 703

27 Mise en forme de la partition ... 705

27.1 Le bloc \layout ... 705
27.2 Définition de la taille de portée ... 707

28 Sauts ... 711

28.1 Sauts de ligne ... 711
28.2 Sauts de page ... 715
28.2.1 Saut de page manuel ... 715
28.2.2 Optimisation des sauts de page ... 716
28.2.3 Minimisation des sauts de page ... 716
28.2.4 Présentation en page continue ... 717
28.2.5 Présentation en ligne continue ... 717
28.2.6 Présentation en rouleau ... 717
28.2.7 Optimisation des tournes ... 717

29 Espacement vertical ... 719

29.1 Espacement vertical au sein d'un système ... 719
29.1.1 Propriétés d'espacement au sein d'un système ... 719
29.1.2 Espacement de portées isolées ... 722
29.1.3 Espacement de portées regroupées ... 723
29.1.4 Espacement des lignes rattachées à des portées ... 725
29.2 Positionnement explicite des portées et systèmes ... 726
29.3 Résolution des collisions verticales ... 733

30 Espacement horizontal 736
30.1 Généralités sur l'espacement horizontal 736
30.2 Changement d'espacement en cours de partition 737
30.3 Modification de l'espacement horizontal global 738
30.3.1 Étirement uniforme des n-olets 739
30.3.2 Espacement strict des notes 740
30.4 Ajustement de l'espacement horizontal de certains objets de rendu 740
30.4.1 Généralités sur l'ajustement de l'espacement de certains objets 740
30.4.2 Espacement entre éléments adjacents non musicaux 741
30.4.3 Espacement entre empilements adjacents 743
30.5 Largeur de ligne 745
30.6 Notation proportionnelle 746

31 Réduction du nombre de pages de la partition 752
31.1 Mise en évidence de l'espacement 752
31.2 Modification de l'espacement 753

Modification des réglages prédéfinis

32 Personnalisation des sorties 759

33 Contextes d'interprétation 760
33.1 Tout savoir sur les contextes 760
33.1.1 Définitions de la sortie – hiérarchie des contextes 760
33.1.2 `Score` – le père de tous les contextes 760
33.1.3 Contextes de haut niveau – les systèmes 761
33.1.4 Contextes de niveau intermédiaire – les portées 761
33.1.5 Contextes de bas niveau – les voix 762
33.2 Création et référencement d'un contexte 763
33.3 Conservation d'un contexte 767
33.4 Modification des greffons de contexte 769
33.5 Modification des réglages par défaut d'un contexte 771
33.5.1 Modification de tous les contextes d'un même type 771
33.5.2 Modification d'un contexte particulier 773
33.5.3 Ordre de préséance 775
33.6 Définition de nouveaux contextes 776
33.7 Ordonnancement des contextes 779

34 En quoi consiste la référence des propriétés internes 782
34.1 Navigation dans les références du programme 782
34.2 Interfaces de rendu 783
34.3 Détermination de la propriété d'un objet graphique (grob) 784

35 Modification de propriétés 786

35.1 Vue d'ensemble de la modification des propriétés 786
35.2 Les commandes \set et \unset 786
35.3 Les commandes \override et \revert 788
35.4 La commande \once 790
35.5 \set ou \override 790
35.6 Les commandes \tweak et \single 791
35.7 La commande \offset 793
35.8 Modification de listes associatives 798

36 Propriétés et contextes utiles 801

36.1 Direction et positionnement 801
36.1.1 Indicateurs de position d'une articulation 801
36.1.2 La propriété direction 802
36.2 Distances et unités de mesure 803
36.3 Dimensions 803
36.4 Extenseurs et prolongateurs 804
36.4.1 Modification de bandeaux avec rupture 804
36.4.2 Affectation d'une longueur minimale aux extenseurs 806
36.4.3 Contrôle du point d'arrêt des bandeaux 809
36.5 Styles de ligne 809
36.6 Lignes de prolongation 810
36.7 Visibilité des objets 812
36.7.1 Suppression des stencils 812
36.7.2 Transparence des objets 813
36.7.3 Blanchiment des objets 813
36.7.4 Utilisation de break-visibility 815
36.7.5 Considérations spécifiques 816
36.8 Rotation des objets 819
36.8.1 Rotation des objets de mise en forme 819
36.8.2 Rotation des étiquettes 819
36.9 Alignement des objets 820
36.9.1 Détermination directe de X-offset et Y-offset 821
36.9.2 Utilisation de side-position-interface 822
36.9.3 Utilisation de self-alignment-interface 822
36.9.4 Utilisation de break-aligned-interface 826
36.10 Modification des stencils 829
36.11 Modification de l'allure des éléments 830
36.11.1 Modification des liaisons 830

Annexes

A Commandes pour *markups* 837

A.1 Commandes pour *markup* 837
A.1.1 Font markup 837
A.1.2 Markup for text alignment 850
A.1.3 Graphical markup 868
A.1.4 Markup for music and musical symbols 879
A.1.5 Conditional markup 892

A.1.6 Instrument-specific markup 892
A.1.7 Accordion registers 898
A.1.8 Other markup commands 902
A.2 Commandes pour liste de *markups* 914

B Tables du manuel de notation 918
B.1 Table des noms d'accord 918
B.2 Modificateurs d'accord usuels 918
B.3 Accordages prédéfinis 921
B.4 Diagrammes d'accord prédéfinis 922
B.4.1 Diagrammes pour guitare 922
B.4.2 Diagrammes pour ukulele 924
B.4.3 Diagrammes pour mandoline 925
B.5 Formats de papier prédéfinis 927
B.6 Instruments MIDI 929
B.7 Liste des couleurs 930
B.8 La fonte Emmentaler 935
B.8.1 Tables de glyphes modernes 935
B.8.2 Tables des glyphes anciens 937
B.9 Styles de tête de note 937
B.10 Jeux de glyphes d'altération 938
B.11 Styles de clef 939
B.11.1 Clefs standards 939
B.11.2 Clefs pour portée de percussions 940
B.11.3 Clefs pour tablatures 940
Clefs de musique ancienne 941
B.12 Liste des caractères spéciaux 943
B.13 Liste des signes d'articulation 943
Scripts d'articulation 944
Scripts d'ornement 944
Scripts de point d'orgue et point d'arrêt 944
Scripts spécifiques à certains instruments 945
Scripts de reprise et de répétition 945
Scripts pour musique ancienne 945
B.14 Liste des marques de respiration 946
B.15 Notes utilisées en percussion 946
B.16 Liste des types de barre de mesure 948
B.17 Valeurs par défaut de `outside-staff-priority` 952
B.18 Valeurs par défaut de `script-priority` 953
B.19 Objets graphiques et leurs symboles d'alignement après rupture 954
B.20 Glossaire technique 955
B.21 Fonctions musicales prédéfinies 959
B.22 Identificateurs de modification de contexte 979
B.23 Variables pour format imprimable 980
B.24 Conventions de nommage 989
B.25 Types de prédicats prédéfinis 989
B.25.1 R5RS primary predicates 989
B.25.2 R5RS secondary predicates 990
B.25.3 Guile predicates 990
B.25.4 LilyPond scheme predicates 990
B.25.5 LilyPond exported predicates 991

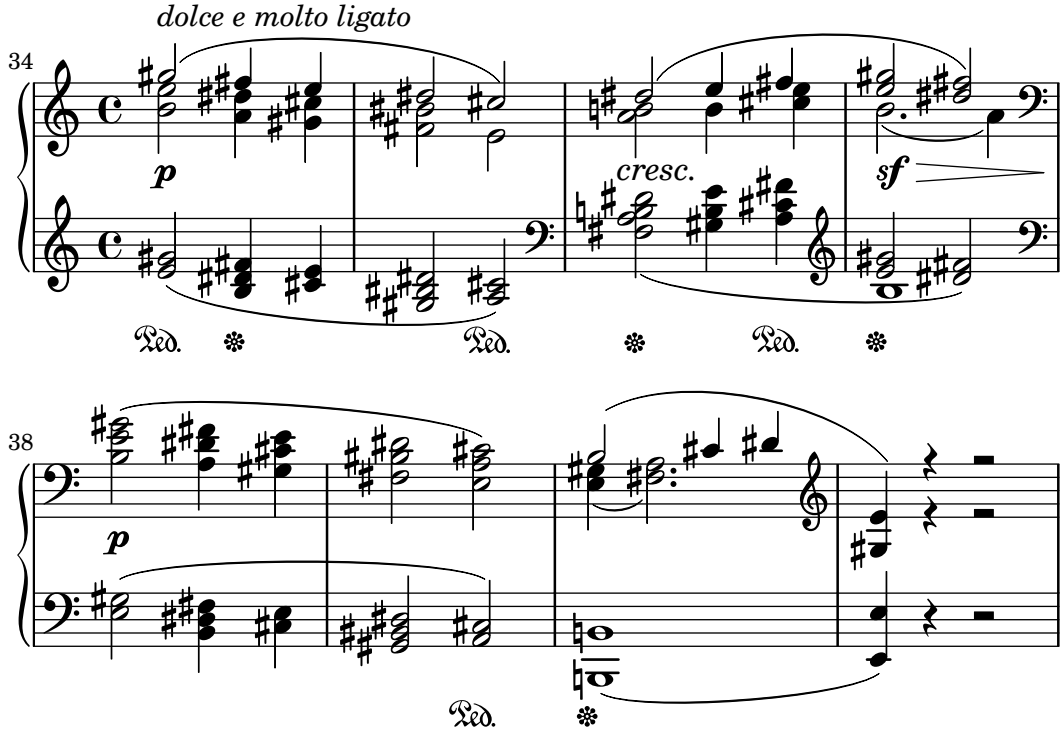
C Aide-mémoire **993**

D GNU Free Documentation License **996**

E Index des commandes et concepts **1003**

Notation musicale générale

1 Hauteurs



Cette section détaille la façon d'indiquer la hauteur des notes, sous trois aspects : la saisie des hauteurs, la modification des hauteurs et les options de gravure.
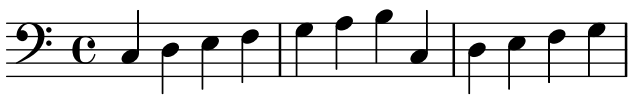
1.1 Écriture des hauteurs de note

Cette section explique la manière d'indiquer les hauteurs de note. Deux modes permettent d'indiquer l'octave des notes : le mode absolu, et le mode relatif. Ce dernier est le plus pratique lors de la saisie d'un fichier source au clavier de l'ordinateur.

1.1.1 Hauteurs avec octave absolue
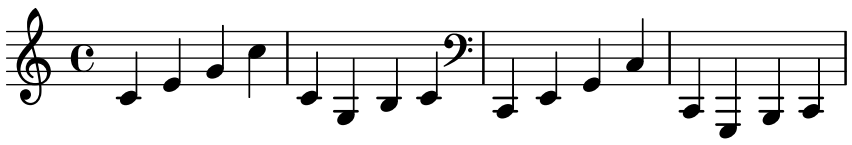
La hauteur s'écrit – à moins de préciser une autre langue – avec la notation batave, en utilisant les lettres de `a` à `g`. Les notes `c` (do) et `b` (si) sont écrites une octave sous le do central.

```
{
  \clef bass
  c4 d e f
  g4 a b c
  d4 e f g
}
```
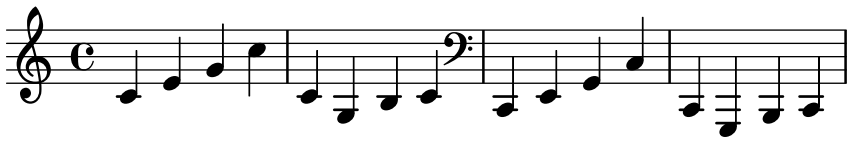


L'octave peut être précisée sous forme d'une série d'apostrophes `'` ou d'une série de virgules `,`. Chaque `'` hausse la note d'une octave ; chaque `,` baisse la note d'une octave.

```
{
  \clef treble
  c'4 e' g' c''
  c'4 g b c'
  \clef bass
  c,4 e, g, c
  c,4 g,, b,, c,
}
```



Les indications d'octave communes peuvent ne se mentionner qu'une fois, en faisant suivre l'instruction `\fixed`, placée avant la musique, d'une hauteur de référence. Les hauteurs d'une section `\fixed` ne nécessitent des `'` ou `,` que lorsqu'elles se trouvent au-dessus ou au-dessous de l'octave de la hauteur de référence.

```
{
  \fixed c' {
    \clef treble
    c4 e g c'
    c4 g, b, c
  }
  \clef bass
  \fixed c, {
    c4 e g c'
    c4 g, b, c
  }
}
```



Les hauteurs d'une expression musicale venant après un `\fixed` ne seront en rien affectées par un éventuel `\relative` qui la contiendrait.

Voir aussi

Glossaire musicologique : Section "Noms des notes" dans *Glossaire*.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

1.1.2 Octaves relatives

Le mode d'écriture en octave absolue requiert d'indiquer l'octave de chaque note. Pour le mode d'écriture en octave relative, par contre, l'octave d'une note est déterminée par rapport à la note précédente : modifier l'octave d'une note aura des répercussions sur toutes les notes à venir.

Une musique peut être déclarée explicitement comme étant en notation relative à l'aide de la commande `\relative` :

`\relative` *`hauteur_de_référence expression_musicale`*

En mode relatif, chaque note est considérée comme étant le plus proche possible de celle qui la précède. L'octave des notes mentionnées dans *expression_musicale* va être calculée de la manière suivante :

- Si aucun signe de changement d'octave n'est utilisé, l'intervalle de base entre la note actuelle et la précédente sera toujours au plus d'une quarte. Cet intervalle est déterminé sans tenir compte des altérations.
- Un signe de changement d'octave `'` ou `,` peut être ajouté pour hausser ou baisser la note d'une octave par rapport à la hauteur calculée sans spécification.
- Ces signes de changement d'octave peuvent être multipliés. Par exemple, `''` ou `,,` ajouteront une octave supplémentaire.
- La première hauteur de *expression_musicale* est déterminée relativement à *hauteur_de_référence*. Cette *hauteur_de_référence* s'exprime en octave absolue ; plusieurs options s'offrent à vous :

  une octave de do (`c`)
  : Un `c'` identifiant le do placé entre les portées d'un piano, il est de fait aisé de déterminer d'autres octaves de `c`. Pour une musique qui commencerait par un sol dièse (`gis`) au-dessus du do suraigu (`c'''`), vous écririez quelque chose comme `\relative c''' { gis' ... }`
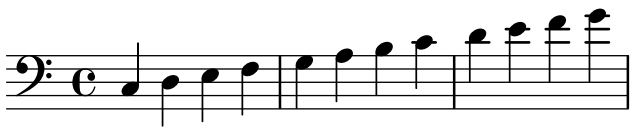
  une octave de la première note de l'expression
  : Écrire `\relative gis'' { gis ... }` permet de déterminer facilement la hauteur absolue de la première note de l'expression.

  pas de hauteur de référence explicite
  : La formulation `\relative { gis''' ... }`) peut se voir comme une version abrégée de l'option précédente : la première note de l'expression est écrite en octave absolue. Cette option est équivalente à prendre un `f` comme hauteur de référence.

  La documentation de LilyPond utilise en règle générale la dernière option.
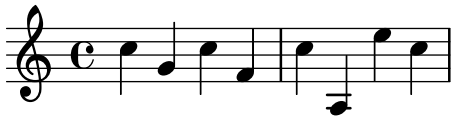
Voici le mode `\relative` en action.

```
\relative {
  \clef bass
  c d e f
  g a b c
  d e f g
}
```
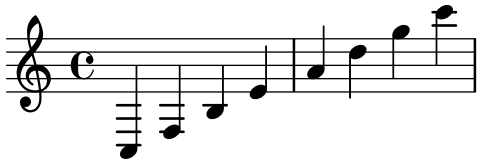


On utilise les signes de changement d'octave pour les intervalles dépassant la quarte.

```
\relative {
  c'' g c f,
  c' a, e'' c
}
```
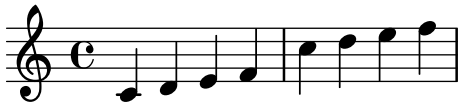


Bien que ne comportant aucun signe de changement d'octave, une séquence de notes peut tout à fait couvrir un intervalle important.

```
\relative {
  c f b e
  a d g c
}
```
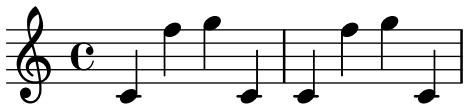


Lorsque plusieurs blocs `\relative` sont imbriqués, le bloc `\relative` inclus dispose de sa propre hauteur de référence indépendamment de celui qui l'englobe.

```
\relative {
  c' d e f
  \relative {
    c'' d e f
  }
}
```



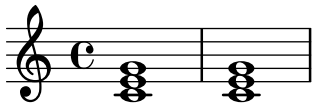
Pour utiliser le mode absolu au sein d'une expression `\relative`, la musique en hauteurs absolues doit se trouver dans dans un bloc `\fixed c { ... }` ; les hauteurs absolues n'affecteront en rien les octaves de la musique relative.

```
\relative {
  c'4 \fixed c { f'' g'' } c |
  c4 \fixed c'' { f g } c
}
```
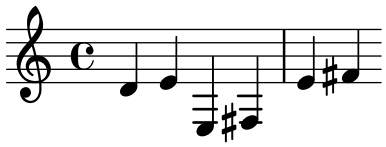


`\relative` n'est pas permis au sein d'un bloc `\chordmode`. Les blocs `\chordmode` au sein d'un bloc `\relative` demeurent inchangés.

```
\new Staff {
  \relative c''' {
    \chordmode { c1 }
  }
  \chordmode { c1 }
}
```
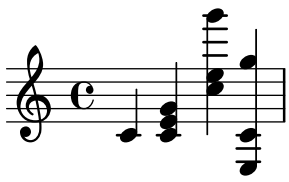


Pour utiliser le mode d'octave relative dans de la musique transposée, une clause `\relative` additionnelle doit être placée au sein du bloc `\transpose`.

```
\relative {
  d' e
  \transpose f g {
    d e
    \relative {
      d' e
    }
  }
}
```
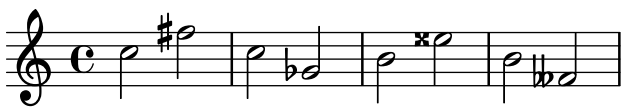


Si l'expression précédente est un accord, c'est la première note de l'accord qui détermine l'emplacement de la première note du prochain accord. À l'intérieur de l'accord, les notes sont placées relativement à celle qui précède. Examinez avec attention l'exemple suivant, et tout particulièrement le positionnement des do.

```
\relative {
  c'
  <c e g>
  <c' e g'>
  <c, e, g''>
}
```



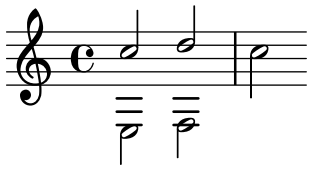
Comme nous l'avons vu, l'octaviation est déterminée sans tenir compte des altérations. Ainsi un mi double-dièse qui suit un si naturel sera placé au-dessus de celui-ci, alors qu'un fa double-bémol se retrouvera en dessous. En d'autres termes, une quarte doublement augmentée demeure considérée comme un intervalle plus petit qu'une quinte diminuée, bien que la quarte doublement augmentée soit de sept demi-tons et la quinte diminuée de seulement six demi-tons.

```
\relative {
  c''2 fis
  c2 ges
  b2 eisis
  b2 feses
}
```



Dans certaines situation complexes, il peut être souhaitable de revenir à une hauteur déterminée sans tenir compte de ce qui se passait auparavant, à l'aide d'un `\resetRelativeOctave` :

```
\relative {
  <<
    { c''2 d }
    \\
    { e,,2 f }
  >>
  \resetRelativeOctave c''
  c2
}
```



Voir aussi

Glossaire musicologique : Section “quinte” dans *Glossaire*, Section “intervalle” dans *Glossaire*, Section “nom des notes” dans *Glossaire*.

Manuel de notation : Section 1.2.1 [Vérifications d’octave], page 13.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.
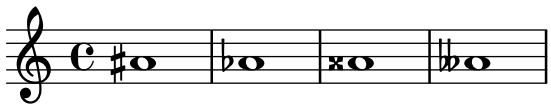
Référence des propriétés internes : Section “RelativeOctaveMusic” dans *Référence des propriétés internes*.

1.1.3 Altérations

> **Note :** Les nouveaux utilisateurs sont parfois déroutés par la gestion des altérations et de l’armure. Pour LilyPond, un nom de note spécifie une hauteur ; l’armure et la clef ne feront que déterminer comment ces hauteurs seront retranscrites. Un simple `c` signifie tout bonnement « do naturel » quelles que soient l’armure et la clef en vigueur. Pour plus d’information, reportez-vous au chapitre Section “Hauteurs et armure” dans *Manuel d’initiation*.
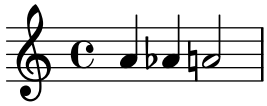
Dans la notation par défaut, un *dièse* est formé en ajoutant `is` après le nom de note, un *bémol* en ajoutant `es`. Les *double-dièses* et *double-bémols* sont obtenus en ajoutant respectivement `isis` ou `eses` au nom de note. Ce sont les noms de note hollandais. Pour les autres langues, consultez Section 1.1.5 [Nom des notes dans d’autres langues], page 11.

```
\relative c'' { ais1 aes aisis aeses }
```



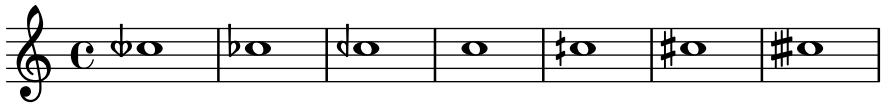
Une hauteur naturelle se saisit comme un simple nom de note, sans suffixe. Un bécarre sera imprimé si besoin est, que ce soit pour annuler les effets d’un précédente altération accidentelle ou pour déroger à l’armure.

```
\relative c'' { a4 aes a2 }
```



Les demi-bémols et demi-dièses s'écrivent en ajoutant respectivement `eh` et `ih`. Voici une série de dos altérés en hauteurs croissantes :
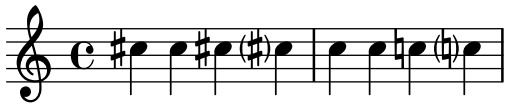
```
\relative c'' { ceseh1 ces ceh c cih cis cisih }
```



Les micro-intervalles sont aussi exportés dans le fichier MIDI.

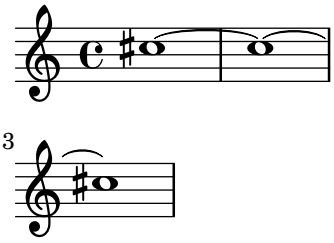
Normalement, les altérations sont imprimées automatiquement, mais il se peut que vous vouliez les imprimer manuellement. On peut forcer l'impression d'une altération, dite « de précaution », en ajoutant un point d'exclamation `!` après la hauteur de note. Une altération entre parenthèses peut être obtenue en ajoutant un point d'interrogation `?` après la hauteur de note.

```
\relative c'' { cis cis cis! cis? c c c! c? }
```



Lorsqu'une note est prolongée par une liaison de tenue, l'altération ne sera réimprimée que s'il y a un saut de ligne.

```
\relative c'' {
  cis1 ~ 1 ~
  \break
  cis
}
```
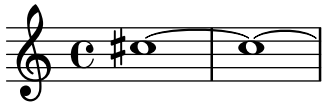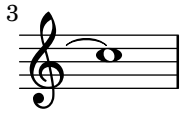


Morceaux choisis

Non répétition de l'altération après saut de ligne sur liaison de prolongation

Cet exemple illustre comment, lorsqu'une note affublée d'une altération accidentelle est prolongée, ne pas répéter cette altération après un saut de ligne.

```
\relative c'' {
  \override Accidental.hide-tied-accidental-after-break = ##t
  cis1~ cis~
  \break
  cis
}
```
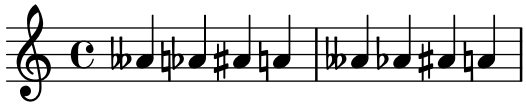




Suppression des bécarres superflus

En accord avec les règles traditionnelles de l'écriture musicale, on grave un bécarre avant un dièse ou un bémol si la note était auparavant affublée d'un double-dièse ou double-bémol. Pour adopter un comportement plus contemporain, la propriété `extraNatural` du contexte `Staff` doit se voir attribuer la valeur `#f` (faux).

```
\relative c'' {
  aeses4 aes ais a
  \set Staff.extraNatural = ##f
  aeses4 aes ais a
}
```



Voir aussi

Glossaire musicologique : Section "dièse" dans *Glossaire*, Section "bémol" dans *Glossaire*, Section "double dièse" dans *Glossaire*, Section "double bémol" dans *Glossaire*, Section "Nom des notes" dans *Glossaire*, Section "quart de ton" dans *Glossaire*.

Manuel d'initiation : Section "Hauteurs et armure" dans *Manuel d'initiation*.

Manuel de notation : Section 1.3.5 [Altérations accidentelles automatiques], page 32, Section 17.3.8 [Altérations suggérées (*musica ficta*)], page 561, Section 1.1.5 [Nom des notes dans d'autres langues], page 11.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Références des propriétés internes : Section "Accidental_engraver" dans *Référence des propriétés internes*, Section "Accidental" dans *Référence des propriétés internes*, Section "AccidentalCautionary" dans *Référence des propriétés internes*, Section "accidental-interface" dans *Référence des propriétés internes*.
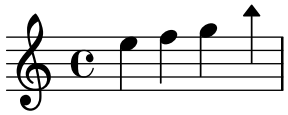
Problèmes connus et avertissements

Il n'y a pas de standard universellement accepté pour noter le bémol et demi (qui abaisse la hauteur trois quarts de ton), le symbole de LilyPond n'est donc conforme à aucun standard.

1.1.4 Hauteurs approximatives

L'écriture d'une note dont on ne connaît pas exactement la hauteur, comme la plus haute note qu'un chanteur peut émettre, s'obtient par la commande `\approximatePitch` à laquelle est transmise une note dont on évalue la hauteur. LilyPond imprime cette note – sous la forme d'un triangle et sans ligne supplémentaire – à la hauteur spécifiée.

```
\relative c'' {
  e4 f g \approximatePitch c
}
```



Voir aussi
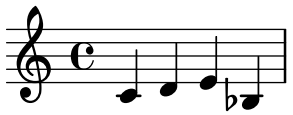
Référence des propriétés internes : Section “ApproximatePitchNoteHead” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*.

1.1.5 Nom des notes dans d’autres langues

Vous disposez de jeux prédéfinis de noms de note et altérations pour plusieurs autres langues. Pour les utiliser, il suffit de déclarer, en début de fichier, la langue que vous utilisez. Voici comment, par exemple, utiliser l’italien pour votre saisie :

```
\language "italiano"

\relative {
  do' re mi sib
}
```



Les langues disponibles ainsi que les noms de note utilisés sont les suivantes :

Lange	Nom des notes
nederlands	c d e f g a bes b
català ou catalan	do re mi fa sol la sib si
deutsch	c d e f g a b h
english	c d e f g a bf/b-flat b
español ou espanol	do re mi fa sol la sib si
français	do ré/re mi fa sol la sib si
italiano	do re mi fa sol la sib si
norsk	c d e f g a b h
português ou portugues	do re mi fa sol la sib si
suomi	c d e f g a b h
svenska	c d e f g a b h
vlaams	do re mi fa sol la sib si

et les suffixes d’altération correspondants :
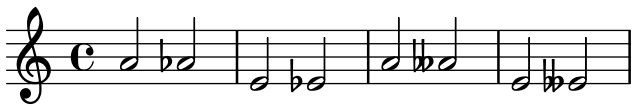
Langue	dièse	bémol	double dièse	double bémol
nederlands	is	es	isis	eses
català ou catalan	d/s	b	dd/ss	bb
deutsch	is	es	isis	eses
english	s/-sharp	f/-flat	ss/x/-sharpsharp	ff/-flatflat
español ou espanol	s	b	ss/x	bb
français	d	b	dd/x	bb
italiano	d	b	dd	bb
norsk	iss/is	ess/es	ississ/isis	essess/eses
português ou portugues	s	b	ss	bb
suomi	is	es	isis	eses

svenska	iss	ess	ississ	essess
vlaams	k	b	kk	bb

Notez qu'en hollandais, en allemand, en norvégien et en finnois, un *la* altéré de *bémol* se note aes et se contracte en as ; pour le hollandais et le norvégien, LilyPond accepte cependant les deux formes. Il en va de même pour es et ees, aeses et ases, ainsi que pour eeses et eses.

En allemand et en finnois, LilyPond fournit la forme plus couramment utilisée de asas pour ases.

```
\relative c'' { a2 as e es a ases e eses }
```
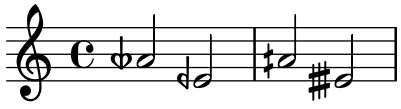


Certaines musiques utilisent des microtonalités, pour lesquelles les altérations sont des fractions de dièse ou bémol « normaux ». Le tableau suivant répertorie les noms de note en quart de ton, tels que définis dans les fichiers linguistiques. Les préfixes *semi-* et *sesqui-* correspondent au *demi-* et *trois demis*.

Langue	**semi-dièse**	**semi-bémol**	**sesqui-dièse**	**sesqui-bémol**
nederlands	ih	eh	isih	eseh
català ou catalan	qd/qs	qb	tqd/tqs	tqb
deutsch	ih	eh	isih	eseh
english	qs	qf	tqs	tqf
español ou espanol	cs	cb	tcs	tcb
français	sd	sb	dsd	bsb
italiano	sd	sb	dsd	bsb
norsk	ih	eh	issih/isih	esseh/eseh
português ou portugues	sqt	bqt	stqt	btqt
suomi	ih	eh	isih	eseh
svenska	ih	eh	issih	esseh
vlaams	hk	hb	khk	bhb

En allemand, les contractions de microtonalités sont identiques à celles des hauteurs normales indiquées ci-dessus.

```
\language "deutsch"

\relative c'' { asah2 eh aih eisih }
```



La plupart des langues dont nous venons de parler correspondent à la musique classique occidentale au tempérament égal – le concept de *Common Practice Period* en anglais. LilyPond prend néanmoins en charge d'autres systèmes de notation, comme indiqué au chapitre Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

Voir aussi

Glossaire musicologique : Section "Nom des notes" dans *Glossaire*, Section "Common Practice Period" dans *Glossaire*.

Manuel de notation : Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

Fichiers d'initialisation : `scm/define-note-names.scm`.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

1.2 Modification de plusieurs hauteurs
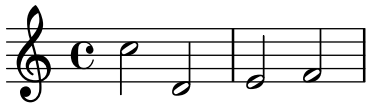
Cette partie traite de la manière de modifier les hauteurs de note.

1.2.1 Vérifications d'octave

Les tests d'octave rendent la correction d'erreurs d'octave plus facile dans le mode d'octave `relative` – un `,` ou un `'` oublié, ça n'arrive pas qu'aux autres !
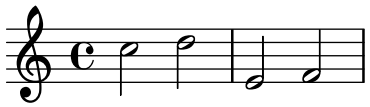
Une note peut être suivie de =*apostrophes/virgules* pour indiquer à quelle **octave absolue** elle devrait être. Dans l'exemple suivant, le premier `d` générera un avertissement, puisqu'on attend un `d''` – intervalle inférieur à la quarte – mais qu'on obtient un `d'`. Sur la partition, l'octave sera corrigée pour donner un `d'` et la prochaine note sera calculée en fonction de ce `d'` et non de `d''`.

```
\relative {
  c''2 d='
  e2 f
}
```



Il existe aussi une vérification d'octave qui ne produit pas de musique imprimée, ayant pour syntaxe `\octaveCheck` *`hauteur_référence`* – *`hauteur_référence`* étant spécifiée en mode absolu. Cette commande vérifie que l'intervalle entre la note qui précède et *`hauteur_référence`* est inférieur à la quinte comme il se doit en mode relatif. Dans le cas contraire, un message sera émis. Bien que la note précédente ne sera pas modifiée, les notes suivantes seront positionnées relativement à la valeur corrigée.

```
\relative {
  c''2 d
  \octaveCheck c'
  e2 f
}
```
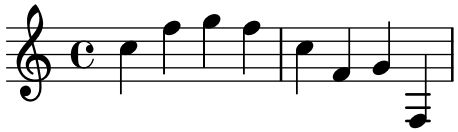


Dans les deux mesures qui suivent, les premier et troisième `\octaveCheck` échouent, mais le deuxième est concluant.

```
\relative {
  c''4 f g f

  c4
  \octaveCheck c'
  f
  \octaveCheck c'
  g
```

```
  \octaveCheck c'
  f
}
```



Voir aussi

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “RelativeOctaveCheck” dans *Référence des propriétés internes*.

1.2.2 Transposition

Une expression musicale peut être transposée avec `\transpose`. En voici la syntaxe :
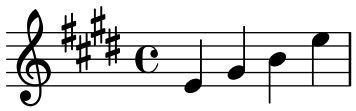
```
\transpose note_de_départ note_d_arrivée expression_musicale
```

Cela signifie que *`expression_musicale`* est transposé de l'intervalle séparant *`note_de_départ`* et *`note_d_arrivée`* : toute note dont la hauteur était `note_de_départ` est changée en `note_d_arrivée` ; les autres notes seront changées selon le même intervalle. Les deux hauteurs s'expriment en octave absolue.

> **Note :** La musique contenue dans un bloc `\transpose` est en octaves absolues, sauf à inclure dans ce même bloc une clause `\relative`.
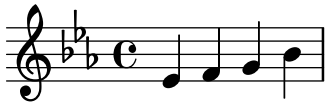
Prenons comme exemple une pièce écrite en ré majeur. Si cette pièce est un peu trop basse pour l'interprète, elle peut être transposée en mi majeur. Vous noterez que l'armure est automatiquement modifiée.

```
\transpose d e {
  \relative {
    \key d \major
    d'4 fis a d
  }
}
```



Regardons maintenant une partie écrite pour violon – un instrument en ut. Si cette partie doit être jouée par une clarinette en la (écrite à la tierce mineure supérieure, un do écrit donnant un la réel), la transposition suivante créera la partie appropriée.
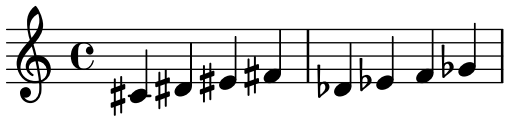
```
\transpose a c' {
  \relative {
    \key c \major
    c'4 d e g
  }
}
```



La présence de `\key c \major` s'explique par le fait que, bien que les notes soient effectivement transposées, l'armure ne sera imprimée que dans la mesure où elle est explicitement mentionnée.

`\transpose` fait la distinction entre les notes enharmoniques : `\transpose c cis` et `\transpose c des` transposeront la pièce un demi-ton plus haut, au détail près que la première version écrira des dièses et la deuxième des bémols.

```
music = \relative { c' d e f }
\new Staff {
  \transpose c cis { \music }
  \transpose c des { \music }
}
```



On peut aussi utiliser `\transpose` pour entrer des notes écrites pour un instrument transpositeur. Normalement, les hauteurs dans LilyPond sont écrites en ut, c'est à dire en sons réels, mais elles peuvent être écrites dans un autre ton. Prenons l'exemple d'un morceau pour trompette en si bémol commençant sur un ré à l'oreille ; on pourrait écrire

```
musiqueEnSiBemol = { e4 ... }
\transpose c bes, \musiqueEnSiBemol
```

Pour imprimer cette musique en fa – et de ce fait produire une partie de cor au lieu d'un conducteur en notes réelles – on utilisera un deuxième `\transpose` :

```
musiqueEnSiBemol = { e4 ... }
\transpose f c' { \transpose c bes, \musiqueEnSiBemol }
```

Pour plus d'information à ce sujet, consultez Section 1.3.4 [Instruments transpositeurs], page 30.

Morceaux choisis

Transposition et réduction du nombre d'altérations accidentelles

Cet exemple, grâce à un peu de code Scheme, donne la priorité aux enharmoniques afin de limiter le nombre d'altérations supplémentaires. La règle applicable est :

- Les altérations doubles sont supprimées
- Si dièse -> Do
- Mi dièse -> Fa
- Do bémol -> Si
- Fa bémol -> Mi

Cette façon de procéder aboutit à plus d'enharmoniques naturelles.

```
#(define (naturalize-pitch p)
   (let ((o (ly:pitch-octave p))
         ;; `ly:pitch-alteration` returns quarter tone steps.
         (a (* 4 (ly:pitch-alteration p)))
         (n (ly:pitch-notename p)))
     (cond
      ((and (> a 1)
            (or (eqv? n 6) (eqv? n 2)))
       (set! a (- a 2))
       (set! n (+ n 1)))
      ((and (< a -1)
```
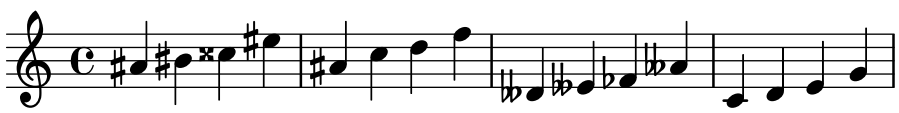
```
            (or (eqv? n 0) (eqv? n 3)))
        (set! a (+ a 2))
        (set! n (- n 1))))
      (cond
       ((> a 2)
        (set! a (- a 4))
        (set! n (+ n 1)))
       ((< a -2)
        (set! a (+ a 4))
        (set! n (- n 1))))
      (when (< n 0)
        (set! o (- o 1))
        (set! n (+ n 7)))
      (when (> n 6)
        (set! o (+ o 1))
        (set! n (- n 7)))
      (ly:make-pitch o n (/ a 4))))

#(define (naturalize music)
   (let ((es (ly:music-property music 'elements))
         (e (ly:music-property music 'element))
         (p (ly:music-property music 'pitch)))
     (when (pair? es)
       (ly:music-set-property! music 'elements
                               (map naturalize es)))
     (when (ly:music? e)
       (ly:music-set-property! music 'element
                               (naturalize e)))
     (when (ly:pitch? p)
       (set! p (naturalize-pitch p))
       (ly:music-set-property! music 'pitch p))
     music))

naturalizeMusic =
#(define-music-function (m) (ly:music?)
   (naturalize m))

music = \relative c' { c4 d e g }

\new Staff {
  \transpose c ais { \music }
  \naturalizeMusic \transpose c ais { \music }
  \transpose c deses { \music }
  \naturalizeMusic \transpose c deses { \music }
}
```



Voir aussi

Manuel de notation : Section 1.3.4 [Instruments transpositeurs], page 30, Section 1.2.3 [Inversion], page 17, Section 1.1.2 [Octaves relatives], page 4, Section 1.2.4 [Rétrogradation], page 17, Section 1.2.5 [Transformations modales], page 18.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriété internes : Section “TransposedMusic” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Si vous voulez utiliser en même temps `\transpose` et `\relative`, vous devez mettre `\transpose` en dehors de `\relative`, puisque `\relative` n'aura aucun effet sur la musique apparaissant dans un `\transpose`.

La fonction `\transpose` ne permet pas d'imprimer des altérations triples ; elle les remplacera par un « équivalent enharmonique » – par exemple ré bémol au lieu de mi triple bémol.
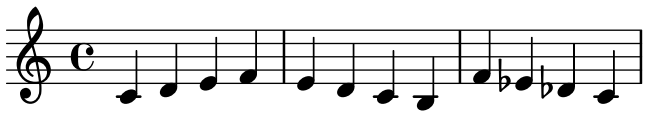
1.2.3 Inversion

Une expression musicale peut s'inverser et être transposée à l'aide de l'instruction

`\inversion` *`hauteur-pivot hauteur-arrivée expression_musicale`*

L'*`expression_musicale`* sera alors inversée, intervalle par intervalle, puis transposée de telle sorte que *`hauteur-pivot`* devienne *`hauteur-arrivée`*.

```
music = \relative { c' d e f }
\new Staff {
  \music
  \inversion d' d' \music
  \inversion d' ees' \music
}
```



Note : Le motif à inverser doit être exprimé en hauteur absolue, à moins d'avoir été préalablement inclus explicitement dans un bloc `\relative`.

Voir aussi
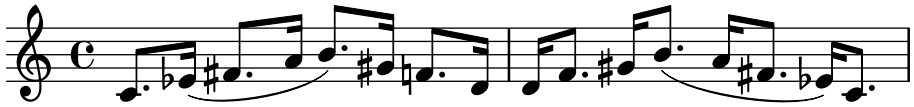
Manuel de notation : Section 1.2.4 [Rétrogradation], page 17, Section 1.2.5 [Transformations modales], page 18, Section 1.2.2 [Transposition], page 14.

1.2.4 Rétrogradation

Une expression musicale peut se renverser et se présenter sous forme rétrograde :

```
music = \relative { c'8. ees16( fis8. a16 b8.) gis16 f8. d16 }

\new Staff {
  \music
  \retrograde \music
}
```



Problèmes connus et avertissements

La fonction `\retrograde` est un outil plutôt simpliste. Dans la mesure où de nombreux événements se reflètent au lieu d'être échangés, les ajustements et indicateurs de positionnement à l'entame d'un objet étendu devront être répétés à leur terminaison : `^(` devra se terminer par `^)`, tout `\<` ou `\cresc` devra se terminer par un `\!` ou un `\endcresc` et tout `\>` ou `\decr` devra se terminer par un `\enddecr`. Les dérogations ou commandes modifiant les propriétés sur la durée peuvent avoir des effets surprenants.

Voir aussi

Manuel de notation : Section 1.2.3 [Inversion], page 17, Section 1.2.5 [Transformations modales], page 18, Section 1.2.2 [Transposition], page 14.

1.2.5 Transformations modales

Dans une composition basée sur une gamme, un même motif est transformé à plusieurs reprises et selon des schémas différents. Il peut être *transposé* pour partir de différents points de la gamme ou bien être *inversé* à partir d'une note pivot dans la gamme. Il peut aussi être renversé pour produire une rétrogradation.

> **Note :** Toute note qui ne ferait pas partie de la gamme en question ne sera pas transformée.

Transposition modale
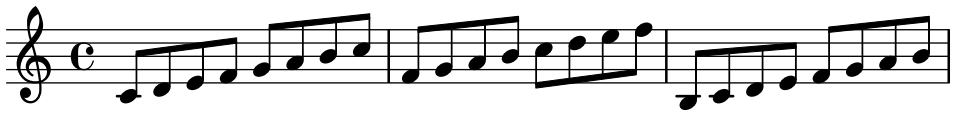
Un motif peut se transposer selon une gamme donnée :

`\modalTranspose` *hauteur-départ hauteur-arrivée gamme motif*

Les notes de *motif* seront décalées à l'intérieur de la *gamme* selon leur degré, déterminé par l'intervalle entre *hauteur-départ* et *hauteur-arrivée* :

```
diatonicScale = \relative { c' d e f g a b }
motif = \relative { c'8 d e f g a b c }
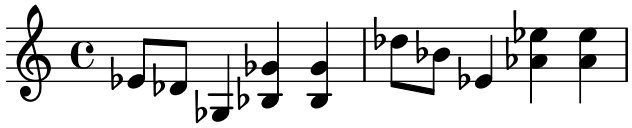
\new Staff {
  \motif
  \modalTranspose c f \diatonicScale \motif
  \modalTranspose c b, \diatonicScale \motif
}
```



Il est ainsi possible de déterminer une gamme ascendante, quels qu'en soient l'amplitude et les différents intervalles :

```
pentatonicScale = \relative { ges aes bes des ees }
motif = \relative { ees'8 des ges,4 <ges' bes,> <ges bes,> }
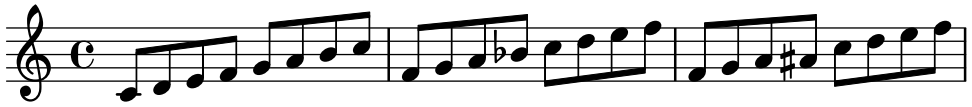
\new Staff {
  \motif
  \modalTranspose ges ees' \pentatonicScale \motif
}
```



L'utilisation de \modalTranspose avec une gamme chromatique produit les mêmes effets qu'un \transpose, à ceci près que les notes seront alors prédéterminées :

```
chromaticScale = \relative { c' cis d dis e f fis g gis a ais b }
motif = \relative { c'8 d e f g a b c }

\new Staff {
  \motif
  \transpose c f \motif
  \modalTranspose c f \chromaticScale \motif
}
```



Inversion modale

Un motif peut s'inverser selon une certaine gamme et à partir d'un pivot déterminé, puis transposé, le tout en une seule opération :
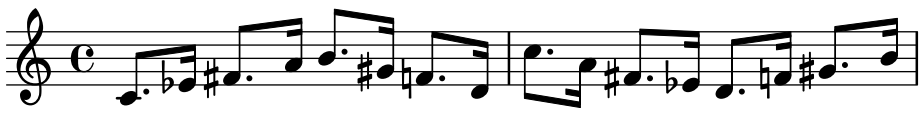
\modalInversion *hauteur-pivot hauteur-arrivée gamme motif*

Les notes de *motif* se retrouvent au même degré par rapport à *hauteur-pivot* dans la *gamme*, toutefois dans le sens opposé, puis décalées dans cette même *gamme* de l'intervalle séparant *hauteur-départ* et *hauteur-arrivée*.

Il est donc possible de simplement inverser à partir d'une des notes de la gamme en donnant la même valeur à *hauteur-départ* et *hauteur-arrivée* :

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }
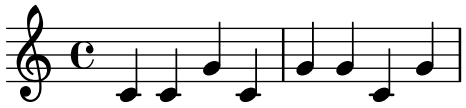
\new Staff {
  \motif
  \modalInversion fis' fis' \octatonicScale \motif
}
```



Pour permuter deux notes de la gamme, il suffit donc d'inverser à partir de l'une des notes et de transposer d'un degré de la gamme. Les deux notes spécifiées peuvent s'interpréter comme étant les bornes du pivot.

```
scale = \relative { c' g' }
motive = \relative { c' c g' c, }
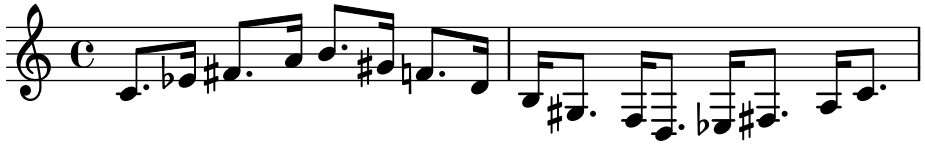
\new Staff {
  \motive
  \modalInversion c' g' \scale \motive
}
```



L'opération conjointe d'une inversion et d'une rétrogradation produit une rétrogradation inversée :

```
octatonicScale = \relative { ees' f fis gis a b c d }
motif = \relative { c'8. ees16 fis8. a16 b8. gis16 f8. d16 }

\new Staff {
  \motif
  \retrograde \modalInversion c' c' \octatonicScale \motif
}
```



Voir aussi

Manuel de notation : Section 1.2.3 [Inversion], page 17, Section 1.2.4 [Rétrogradation], page 17, Section 1.2.2 [Transposition], page 14.
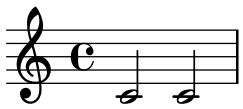
1.3 Gravure des hauteurs

Nous allons voir dans cette partie comment influer sur la gravure des hauteurs.

1.3.1 Clefs

La clef indique quelles lignes de la portée correspondent à quelles hauteurs. En l'absence de commande explicite, LilyPond utilise par défaut la clef de sol.
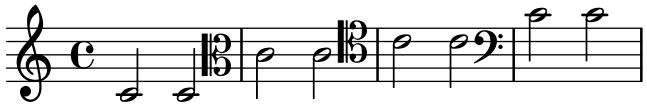
```
c'2 c'
```



La clef se modifie à l'aide de la commande `\clef` suivie d'un nom approprié. Pour chacun des exemples suivants est indiquée la position du do médium.

```
\clef treble
c'2 c'
\clef alto
c'2 c'
\clef tenor
c'2 c'
\clef bass
```

```
c'2 c'
```
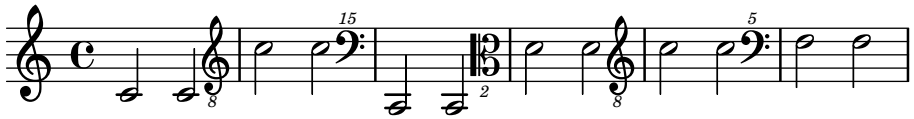


Les différents noms possibles sont répertoriés à l'annexe Section B.11 [Styles de clef], page 939.

Des clefs spéciales, telles que celles rencontrées en musique ancienne, sont abordées dans Section 17.3.2 [Clefs anciennes], page 557, et Section 17.4.2 [Clefs grégoriennes], page 570. La musique requérant des clefs de tablature est traitée dans Section 12.1.3 [Tablatures par défaut], page 446, et Section 12.1.4 [Tablatures personnalisées], page 465.

Les citations peuvent demander une modification de clef à l'aide des commandes `\cueClef` et `\cueDuringWithClef` – voir Section 6.3.3 [Mise en forme d'une citation], page 277.
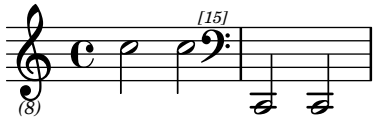
En ajoutant `_8` ou `^8` au nom de la clef, celle-ci est transposée à l'octave respectivement inférieure ou supérieure, et `_15` ou `^15` la transpose de deux octaves. D'autres nombres entiers peuvent être utilisés selon les besoins. L'argument *clefname* doit être mis entre guillemets lorsqu'il contient des caractères supplémentaires. Par exemple,

```
\clef treble
c'2 c'
\clef "treble_8"
c'2 c'
\clef "bass^15"
c'2 c'
\clef "alto_2"
c'2 c'
\clef "G_8"
c'2 c'
\clef "F^5"
c'2 c'
```



Une indication d'octaviation optionnelle s'obtient en entourant l'argument numérique par des parenthèses ou des crochets :

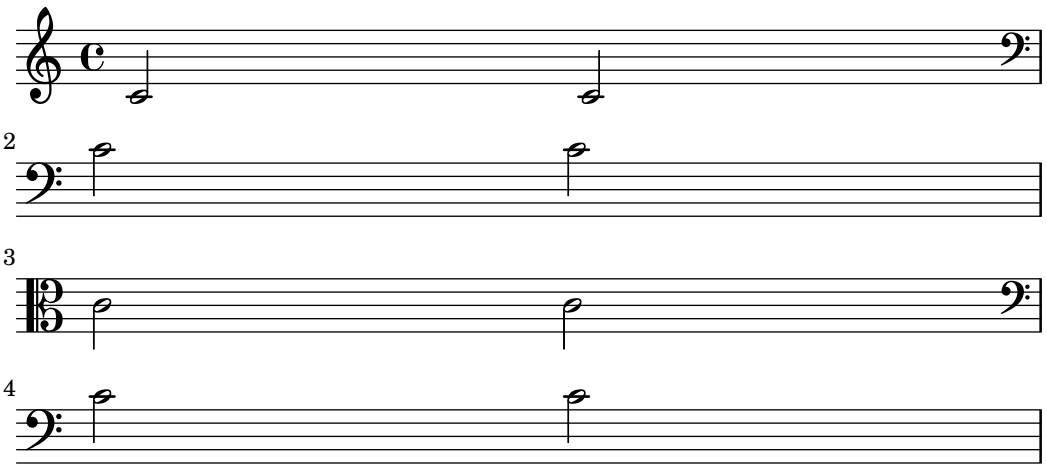
```
\clef "treble_(8)"
c'2 c'
\clef "bass^[15]"
c'2 c'
```



Les hauteurs seront affichées comme si l'argument numérique n'avait pas été encadré de parenthèses ou crochets.
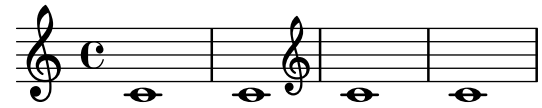
Lorsqu'un changement de clef intervient en même temps qu'un saut de ligne, la nouvelle clef est imprimée à la fois en fin de ligne et au début de la suivante. Vous pouvez toujours supprimer cette « clef de précaution ».

```
\clef treble c'2 c'  \break
\clef bass c'2 c' \break
\set Staff.explicitClefVisibility = #end-of-line-invisible
\clef alto c'2 c' \break
\unset Staff.explicitClefVisibility
\clef bass c'2 c' \break
```



Lorsqu'une clef a déjà été imprimée et qu'aucune autre clef n'a depuis été imprimée, LilyPond ignorera toute réitération de la commande `\clef`. Forcer la réimpression de la clef s'obtient à l'aide de la commande `\set Staff.forceClef = ##t`.

```
\clef treble
c'1
\clef treble
c'1
\set Staff.forceClef = ##t
c'1
\clef treble
c'1
```
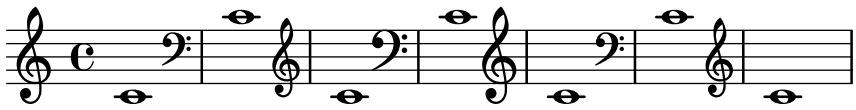


Pour être plus précis, la commande `\clef` n'a pas pour fonction d'imprimer une clef ; elle détermine ou modifie une propriété attachée au graveur de clefs (le `Clef_engraver`), qui décide de son propre chef quand doit être affichée une clef dans la portée en cours. La propriété `forceClef` ne vient que forcer la décision de réimprimer une fois la clef en un point donné.

Le symbole imprimé lors d'un changement de clef est plus petit que la clef initiale. La taille peut toutefois être ajustée.

```
\clef "treble"
c'1
\clef "bass"
c'1
\clef "treble"
c'1
\override Staff.Clef.full-size-change = ##t
\clef "bass"
c'1
```

```
\clef "treble"
c'1
\revert Staff.Clef.full-size-change
\clef "bass"
c'1
\clef "treble"
c'1
```



Morceaux choisis

Affinage des propriétés d'une clef

Modifier le glyphe, la position de la clef ou son octaviation ne changeront pas la position des notes ; il faut pour y parvenir modifier aussi la position du do médium. La redéfinition préalable de `middleCClefPosition` permet de placer l'armure sur les bonnes lignes. Le positionnement est relatif à la ligne médiane, un nombre positif faisant monter, un nombre négatif abaissant.

Par exemple, la commande `\clef "treble_8"` équivaut à définir les propriétés de contexte `clefGlyph`, `clefPosition` – qui contrôle la position verticale de la clef –, `middleCPosition` et `clefOctavation`. Une nouvelle clef apparaîtra dès lors que l'une de ces propriétés, à l'exception de `middleCPosition`, aura été modifiée.

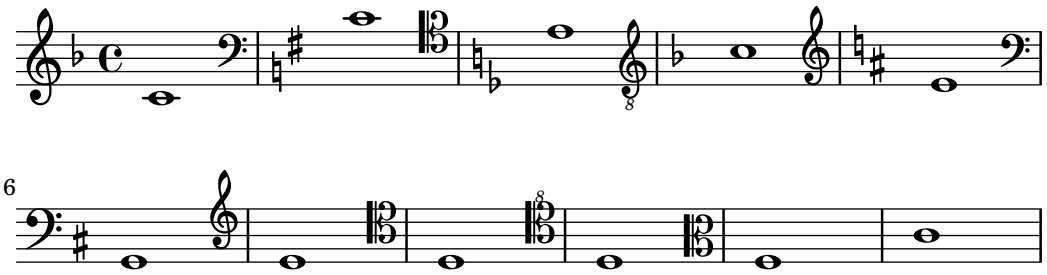
Les exemples qui suivent illustrent les différentes possibilités de définir ces propriétés manuellement. Sur la première ligne, la position relative des notes par rapport aux clefs est préservée, ce qui n'est pas le cas pour la deuxième ligne.

```
{
  % The default treble clef.
  \key f \major
  c'1
  % The standard bass clef
  \set Staff.clefGlyph = "clefs.F"
  \set Staff.clefPosition = 2
  \set Staff.middleCPosition = 6
  \set Staff.middleCClefPosition = 6
  \key g \major
  c'1
  % The baritone clef.
  \set Staff.clefGlyph = "clefs.C"
  \set Staff.clefPosition = 4
  \set Staff.middleCPosition = 4
  \set Staff.middleCClefPosition = 4
  \key f \major
  c'1
  % The standard choral tenor clef.
  \set Staff.clefGlyph = "clefs.G"
  \set Staff.clefPosition = -2
  \set Staff.clefTransposition = -7
  \set Staff.middleCPosition = 1
  \set Staff.middleCClefPosition = 1
  \key f \major
```

```
  c'1
  % A non-standard clef.
  \set Staff.clefPosition = 0
  \set Staff.clefTransposition = 0
  \set Staff.middleCPosition = -4
  \set Staff.middleCClefPosition = -4
  \key g \major
  c'1 \break

  % The following clef changes do not preserve
  % the normal relationship between notes, key signatures
  % and clefs.
  \set Staff.clefGlyph = "clefs.F"
  \set Staff.clefPosition = 2
  c'1
  \set Staff.clefGlyph = "clefs.G"
  c'1
  \set Staff.clefGlyph = "clefs.C"
  c'1
  \set Staff.clefTransposition = 7
  c'1
  \set Staff.clefTransposition = 0
  \set Staff.clefPosition = 0
  c'1

  % Return to the normal clef.
  \set Staff.middleCPosition = 0
  c'1
}
```



Voir aussi

Manuel de notation : Section 17.3.2 [Clefs anciennes], page 557, Section 17.4.2 [Clefs grégoriennes], page 570, Section 6.3.3 [Mise en forme d'une citation], page 277, Chapitre 17 [Notations anciennes], page 552, Section 12.1.3 [Tablatures par défaut], page 446, Section 12.1.4 [Tablatures personnalisées], page 465.
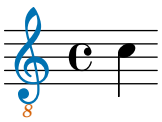
Fichiers d'initialisation : `scm/parser-clef.scm`.

Morceaux choisis: Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Clef_engraver” dans *Référence des propriétés internes*, Section “Clef” dans *Référence des propriétés internes*, Section “ClefModifier” dans *Référence des propriétés internes*, Section “clef-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L'indicateur d'octaviation attaché à la clef est un objet graphique en lui même. Par voie de conséquence, tout `\override` affectant l'objet `Clef` devra être manuellement répercuté sur l'objet `ClefModifier`.



1.3.2 Armure

> **Note :** Les nouveaux utilisateurs sont parfois déroutés par la gestion des altérations et de l'armure. Pour LilyPond, une hauteur n'est que du matériau brut ; l'armure et la clef ne feront que déterminer comment ce matériau sera retranscrit. Un simple `c` signifie tout bonnement « do naturel » quelles que soient l'armure et la clef en question. Pour plus d'information, reportez-vous au chapitre Section "Hauteurs et armure" dans *Manuel d'initiation*.
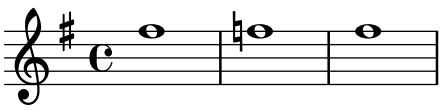
L'armure indique la tonalité dans laquelle la pièce doit être jouée. Elle comprend un ensemble d'altérations (dièses ou bémols) à la clef, c'est-à-dire au début de la portée. Elle peut varier en cours de morceau.

On définit ou modifie l'armure avec la commande `\key` :

```
\key hauteur mode
```

Ici, *mode* doit être `\major` ou `\minor` afin d'avoir respectivement *hauteur*-majeur ou *hauteur*-mineur. Vous pouvez aussi avoir recours aux modes anciens que sont `\ionian`, `\locrian`, `\aeolian`, `\mixolydian`, `\lydian`, `\phrygian` et `\dorian`.
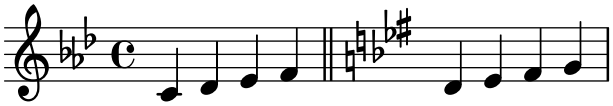
```
\relative {
  \key g \major
  fis''1
  f
  fis
}
```



Rien n'empêche de définir d'autres modes, en listant l'altération de chacun des degrés de la gamme en partant du do.

```
freygish = #`((0 . ,NATURAL) (1 . ,FLAT) (2 . ,NATURAL)
    (3 . ,NATURAL) (4 . ,NATURAL) (5 . ,FLAT) (6 . ,FLAT))
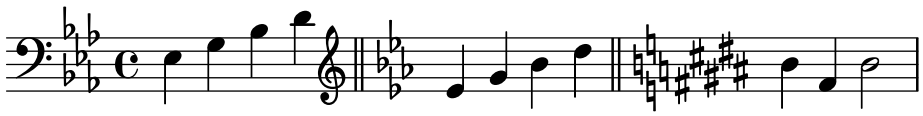
\relative {
  \key c \freygish c'4 des e f
  \bar "||" \key d \freygish d es fis g
}
```



Les altérations à la clef peuvent s'imprimer à des octaves différents de leur position traditionnelle ou à plusieurs octaves, à l'aide des propriétés `flat-positions` et `sharp-positions` de l'objet `KeySignature`. Les entrées fournies à ces propriétés définissent l'amplitude des positions sur la portée où les altérations seront imprimées. Dans le cas où l'entrée est constituée d'une position unique, les altérations seront placées à l'intérieur de l'octave finissant à cette position sur la portée.

```
\override Staff.KeySignature.flat-positions = #'((-5 . 5))
\override Staff.KeyCancellation.flat-positions = #'((-5 . 5))
\clef bass \key es \major es g bes d'
\clef treble \bar "||" \key es \major es' g' bes' d''

\override Staff.KeySignature.sharp-positions = #'(2)
\bar "||" \key b \major b' fis' b'2
```
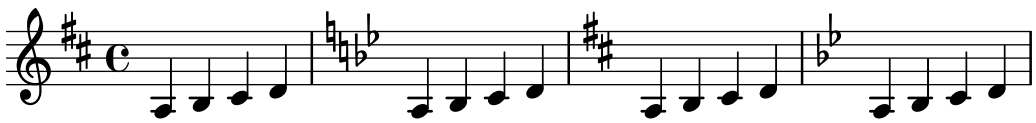


Morceaux choisis

Suppression des bécarres superflus lors d'un changement de tonalité

Après un changement de tonalité, un bécarre est imprimé pour annuler toute altération précédente. Ce comportement s'annule en désactivant la propriété `printKeyCancellation` du contexte `Staff`.

```
\relative c' {
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
  \set Staff.printKeyCancellation = ##f
  \key d \major
  a4 b cis d
  \key g \minor
  a4 bes c d
}
```



Armures inhabituelles

La commande `\key` détermine la propriété `keyAlterations` d'un contexte `Staff`.

Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété. Il s'agit en l'occurrence de définir une liste :

```
\set Staff.keyAlterations =
  #`(((octave . pas) . altération) ((octave . pas) . altération) ...)
```
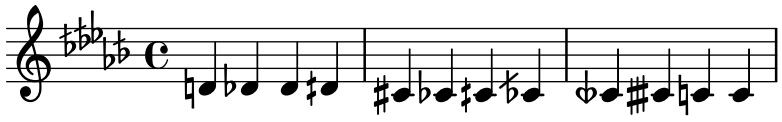
dans laquelle, et pour chaque élément, *octave* spécifie l'octave (0 pour celle allant du do médium au si supérieur), *pas* la note dans cette octave (0 pour do et 6 pour si), et *altération* sera `,SHARP` ou `,FLAT` ou `,DOUBLE-SHARP`, etc. (attention à la virgule en préfixe).

Une formulation abrégée – (*pas . altération*) – signifie que l'altération de l'élément en question sera valide quelle que soit l'octave.

En ce qui concerne les gammes microtonales dans lesquelles un « dièse » n'est pas d'un centième, *altération* se réfère à un deux-centième de ton entier.

```
\include "arabic.ly"

\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                 (1 . ,SEMI-FLAT)
                                 (2 . ,FLAT)
                                 (5 . ,FLAT)
                                 (6 . ,SEMI-FLAT))
  % \set Staff.extraNatural = ##f
  re reb \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dodsd do do |
}
```



Voir aussi

Glossaire musicologique : Section “mode d'église” dans *Glossaire*, Section “scordatura” dans *Glossaire*.

Manuel d'initiation : Section “Hauteurs et armure” dans *Manuel d'initiation*.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Key_engraver” dans *Référence des propriétés internes*, Section “Key_performer” dans *Référence des propriétés internes*, Section “KeyCancellation” dans *Référence des propriétés internes*, Section “KeyChangeEvent” dans *Référence des propriétés internes*, Section “KeySignature” dans *Référence des propriétés internes*, Section “key-signature-interface” dans *Référence des propriétés internes*.
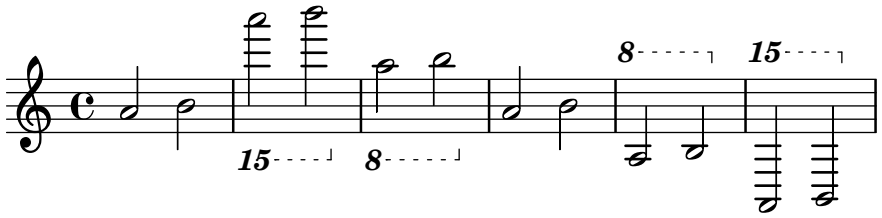
1.3.3 Marques d'octaviation

Les marques d'octaviation, *Ottava*, haussent ou abaissent les notes affichées d'une ou plusieurs octaves, tout en préservant leur hauteur réelle. Le but des octaviations est de réduire le nombre de lignes supplémentaires nécessaires, tout particulièrement en présence d'un intervalle important à l'occasion d'un court passage. L'instruction `\ottava` prend en argument un entier positif lorsque les notes sont plus aiguës que les notes imprimées ; il sera négatif pour qu'elles soient jouées plus bas. Les effets d'un crochet d'octaviation durent tant que n'est pas explicitée une nouvelle octave de référence, ce qui peut s'obtenir à l'aide d'un `\ottava 0`.

Dans l'exemple suivant, toutes les notes sonnent à la même hauteur.

```
\relative c'' {
  a2 b
  \ottava -2
  a2 b
  \ottava -1
  a2 b
  \ottava 0
  a2 b
```
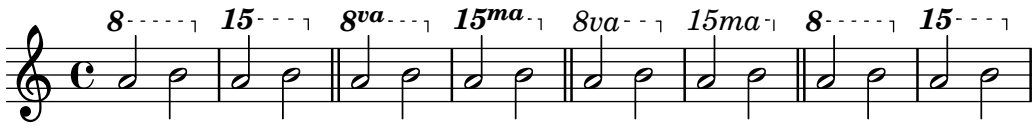
```
  \ottava 1
  a2 b
  \ottava 2
  a2 b
}
```



Par défaut s'imprimera un simple nombre en début de crochet. Ce réglage est modifiable afin d'obtenir un nombre ordinal dont l'abréviation peut se positionner en petite lettre haute ou en caractère normal ; la graisse par défaut de ces caractères est elle aussi modifiable – voir Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

L'exemple suivant illustre différentes options, ainsi que le moyen de retrouver le comportement par défaut.

```
\relative c'' {
  \ottava 1
  a'2 b
  \ottava 2
  a'2 b
  \bar "||"
  \set Staff.ottavationMarkups = #ottavation-ordinals
  \ottava 1
  a,2 b
  \ottava 2
  a'2 b
  \bar "||"
  \override Staff.OttavaBracket.font-series = #'normal
  \set Staff.ottavationMarkups = #ottavation-simple-ordinals
  \ottava 1
  a,2 b
  \ottava 2
  a'2 b
  \bar "||"
  \revert Staff.OttavaBracket.font-series
  \set Staff.ottavationMarkups = #ottavation-numbers
  \ottava 1
  a,2 b
  \ottava 2
  a'2 b
}
```



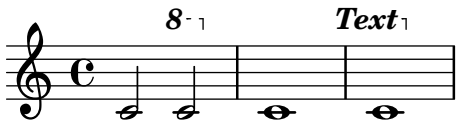
Morceaux choisis

Modification du texte des marques d'octaviation

En interne, la fonction `\ottava` détermine les propriétés `ottavation` (par ex. en `"8va"` ou `"8vb"`) et `middleCPosition`. Vous pouvez modifier le texte d'une marque d'octaviation en définissant `ottavation` après avoir fait appel à `ottava`.

Un texte bref est particulièrement utile lorsque l'octaviation est courte.

```
{
  c'2
  \ottava 1
  \set Staff.ottavation = "8"
  c''2
  \ottava 0
  c'1
  \ottava 1
  \set Staff.ottavation = "Text"
  c''1
}
```



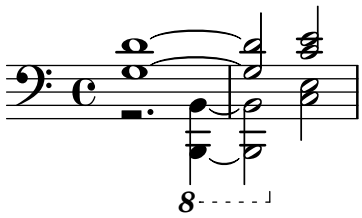
Ajout d'une indication d'octave pour une seule voix

Lorsque plusieurs voix cohabitent sur une même portée, déterminer l'octaviation d'une voix affecte la position des notes de toutes les voix, jusqu'à la fin du crochet d'octaviation. Si l'octaviation ne doit s'appliquer qu'à une seule voix, le `Ottava_spanner_engraver` devrait être déplacé dans le contexte `Voice`.

```
\layout {
  \context {
    \Staff
    \remove Ottava_spanner_engraver
  }
  \context {
    \Voice
    \consists Ottava_spanner_engraver
  }
}

{
  \clef bass
  << { <g d'>1~ q2 <c' e'> }
  \\
    {
      r2.
      \ottava -1
      <b,,, b,,>4 ~ |
      q2
      \ottava 0
      <c e>2
    }
```

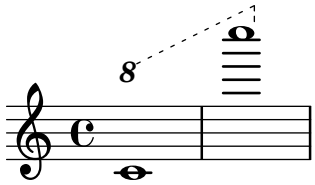
```
  >>
}
```



Modification de la pente de l'extension d'octaviation

Il est possible d'adapter la pente d'une indication d'octaviation.

```
\relative c'' {
  \override Staff.OttavaBracket.stencil = #ly:line-spanner::print
  \override Staff.OttavaBracket.bound-details =
    #`((left . ((Y . 0)
                (attach-dir . ,LEFT)
                (padding . 0)
                (stencil-align-dir-y . ,CENTER)))
       (right . ((Y . 5.0) ; Change the number here
                 (padding . 0)
                 (attach-dir . ,RIGHT)
                 (text . ,(make-draw-dashed-line-markup
                           (cons 0 -1.2))))))
  \override Staff.OttavaBracket.left-bound-info =
     #ly:horizontal-line-spanner::calc-left-bound-info-and-text
  \override Staff.OttavaBracket.right-bound-info =
     #ly:horizontal-line-spanner::calc-right-bound-info
  \ottava 1
  c1
  c'''1
}
```



Voir aussi

Glossaire musicologique : Section "octaviation" dans *Glossaire*.

Manuel de notation : Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Ottava_spanner_engraver" dans *Référence des propriétés internes*, Section "OttavaBracket" dans *Référence des propriétés internes*, Section "ottava-bracket-interface" dans *Référence des propriétés internes*.

1.3.4 Instruments transpositeurs

Lorsque l'on saisit une partition d'ensemble incluant des instruments transpositeurs, certaines parties peuvent être dans une autre tonalité que la *tonalité de concert*. Il faudra en pareil cas indiquer la tonalité spécifique de ces *instruments transpositeurs*, sous peine de fichier MIDI
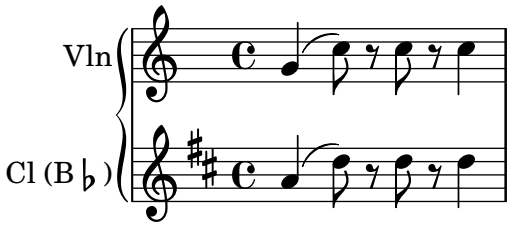
erroné et de citations incorrectes. Pour plus de détails sur les citations, consultez le chapitre Section 6.3.2 [Citation d'autres voix], page 274.

`\transposition` *hauteur*

La hauteur donnée en argument à `\transposition` doit correspondre à la note entendue lorsqu'un `do` écrit sur la portée est joué par l'instrument transpositeur. Cette hauteur doit être mentionnée en *mode absolu*. Par exemple, lorsque vous saisissez une partition en notes réelles, toutes les voix devraient être en ut ; si un instrument joue un ton au dessus, il faudra lui ajouter un `\transposition d'`. La commande `\transposition` s'utilise **si, et seulement si** les notes à saisir **ne sont pas** dans la tonalité de concert.

Voici un fragment pour violon et clarinette en si bémol (*B-flat*) pour lequel les parties respectives ont été recopiées à partir du conducteur. Les deux instruments sont à l'unisson.

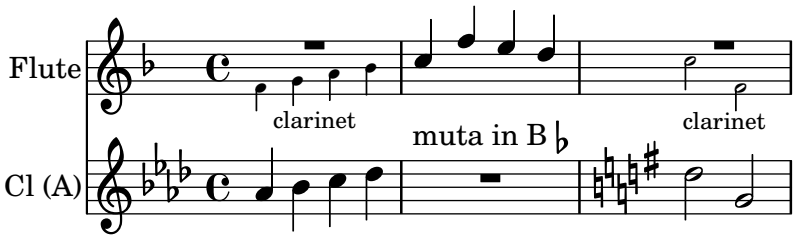
```
\new GrandStaff <<
  \new Staff = "violin" \with {
    instrumentName = "Vln"
    midiInstrument = "violin"
   }
  \relative c'' {
    % not strictly necessary, but a good reminder
    \transposition c'
    \key c \major
    g4( c8) r c r c4
  }
  \new Staff = "clarinet" \with {
    instrumentName = \markup { Cl (B\flat) }
    midiInstrument = "clarinet"
  }
  \relative c'' {
    \transposition bes
    \key d \major
    a4( d8) r d r d4
  }
>>
```



La `\transposition` peut évoluer au cours d'un morceau. Un clarinettiste peut être amené à jongler avec une clarinette en la et une autre en si bémol.

```
flute = \relative c'' {
  \key f \major
  \cueDuring "clarinet" #DOWN {
    R1 _\markup\tiny "clarinet"
    c4 f e d
    R1 _\markup\tiny "clarinet"
  }
}
clarinet = \relative c'' {
```

```
  \key aes \major
  \transposition a
  aes4 bes c des
  R1^\markup { muta in B\flat }
  \key g \major
  \transposition bes
  d2 g,
}
\addQuote "clarinet" \clarinet
<<
  \new Staff \with { instrumentName = "Flute" }
    \flute
  \new Staff \with { instrumentName = "Cl (A)" }
    \clarinet
>>
```



Voir aussi

Glossaire musicologique : Section “tonalité de concert” dans *Glossaire*, Section “instrument transpositeur” dans *Glossaire*.

Manuel de notation : Section 6.3.2 [Citation d’autres voix], page 274, Section 1.2.2 [Transposition], page 14.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

1.3.5 Altérations accidentelles automatiques

LilyPond dispose d’une fonction chargée de regrouper les règles suivant lesquelles s’impriment les altérations. Elle s’invoque de la manière suivante :

```
\new Staff <<
  \accidentalStyle voice
  { ... }
>>
```

La règle de gestion des altérations s’applique normalement au contexte `Staff` en cours, exception faite des styles `choral`, `piano` et `piano-cautionary` comme nous allons le voir. Cette fonction accepte un éventuel argument supplémentaire chargé de spécifier le champ d’action de la règle à suivre. À titre d’exemple, il faudra utiliser, pour que toutes les portées d’un même système – contexte `StaffGroup` – soient soumises à la même règle :

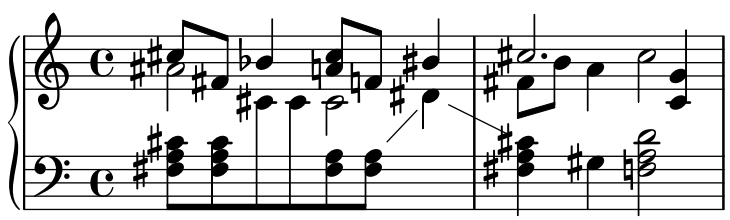
```
\accidentalStyle StaffGroup.voice
```

Nous vous présentons ci-après les différentes règles d’altération prises en charge. Pour les besoins de la démonstration, nous partirons de l’exemple suivant :

```
musicA = {
  <<
    \relative {
      cis''8 fis, bes4 <a cis>8 f bis4 |
      cis2. <c, g'>4 |
    }
    \\
    \relative {
      ais'2 cis, |
      fis8 b a4 cis2 |
    }
  >>
}

musicB = {
  \clef bass
  \new Voice {
    \voiceTwo \relative {
      <fis a cis>8[ <fis a cis>
      \change Staff = up
      cis' cis
      \change Staff = down
      <fis, a> <fis a>]
      \showStaffSwitch
      \change Staff = up
      dis'4 |
      \change Staff = down
      <fis, a cis>4 gis <f a d>2 |
    }
  }
}

\new PianoStaff {
  <<
    \new Staff = "up" {
      \accidentalStyle default
      \musicA
    }
    \new Staff = "down" {
      \accidentalStyle default
      \musicB
    }
  >>
}
```

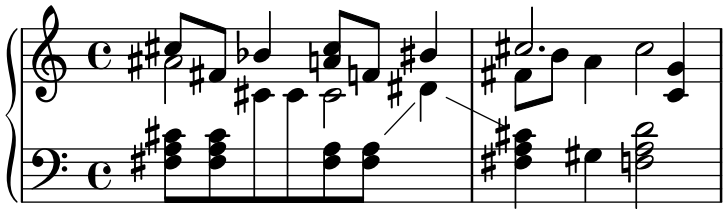


Notez bien que pour appliquer le même style aux deux portées, seules les dernières lignes de cet exemple nous intéressent.

```
\new PianoStaff {
  <<
    \new Staff = "haut" {
      %%% voici la ligne à modifier en conséquence :
      \accidentalStyle Score.default
      \musicA
    }
    \new Staff = "bas" {
      \musicB
    }
  >>
}
```

`default`

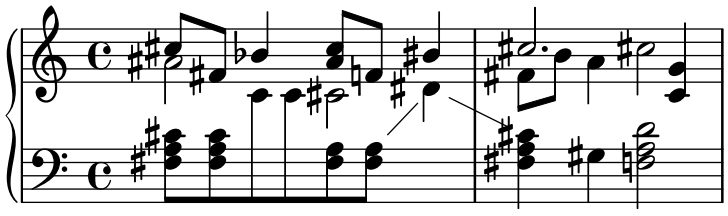
C'est la règle d'impression par défaut, qui se rapporte à l'usage en vigueur au XVIIIe siècle : les altérations accidentelles sont valables toute une mesure, et uniquement à leur propre octave. C'est la raison pour laquelle il n'y a pas de bécarre avant le *si* de la deuxième mesure, ni avant le dernier *do*.



`voice`

En principe, LilyPond se souvient de toutes les altérations présentes sur la portée (contexte `Staff`). Avec cette règle, cependant, les altérations sont indépendantes pour chacune des voix tout en obéissant à la règle `default`.

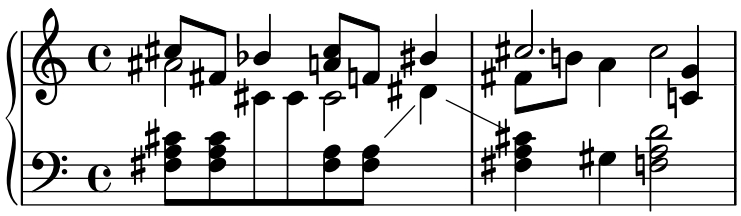
Les altérations d'une voix sont de fait ignorées dans les autres voix, ce qui peut donner lieu à un résultat malencontreux. Dans l'exemple suivant, il est difficile de dire si le deuxième *la* est dièse ou naturel. La règle `voice` n'est donc à envisager que dans le cas de voix devant être lues par des musiciens différents. S'il s'agit d'un « conducteur », ou d'une portée destinée à un seul musicien, il vaut mieux utiliser `modern` ou `modern-cautionary`.



`modern`

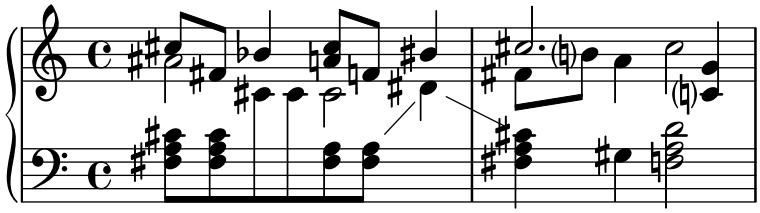
Cette règle est la plus courante au XXe siècle. Certains bécarres ne sont pas imprimés, comme il était d'usage lorsqu'une note diésée suit cette même note flanquée d'un double dièse, ou bien un bémol un double bémol. Le style `modern` suit la même règle que le style `default`, avec deux additions afin de lever les ambiguïtés : lorsqu'une note non altérée apparaît à une octave différente, ou bien dans la mesure suivante,

des bécarres de précaution sont ajoutés. Dans l'exemple suivant, notez ainsi les deux bécarres dans la deuxième mesure de la main droite.



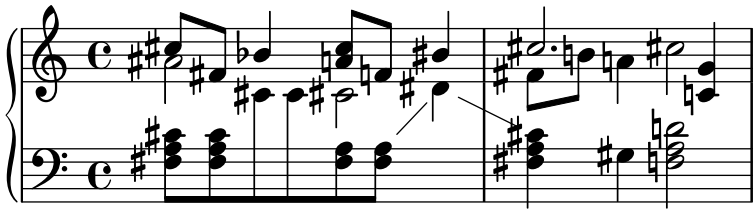
`modern-cautionary`

Cette règle est équivalente à `modern`, mais les bécarres de précaution (absents dans la règle `default`) sont imprimés entre parenthèses. Ils peuvent aussi adopter une taille différente, au moyen de la propriété `font-size` de l'objet `AccidentalSuggestion`.



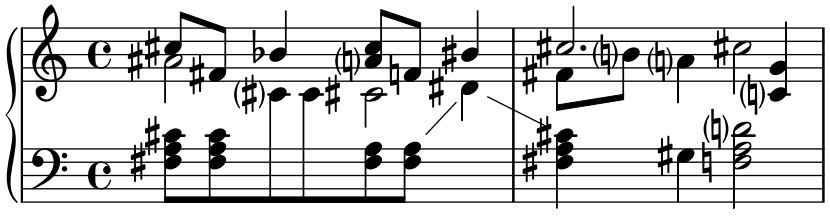
`modern-voice`

Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix, mais les autres voix d'un même contexte `Staff` en *tiennent compte* cette fois. C'est pourquoi le *la* de la dernière mesure est affublé d'un bécarre bien qu'il y en ait déjà eu un dans la mesure précédente, et que le *ré* de la main gauche en ait un alors que le dièse qu'il avait auparavant concernait la main droite.



`modern-voice-cautionary`

Cette règle est similaire à la précédente, mais les altérations de précaution (celles que n'aurait pas ajoutées `voice`), sont imprimées de façon particulière. On retrouve donc toutes les altérations qu'imprimerait `default`, mais certaines sont considérées comme étant « de précaution ».

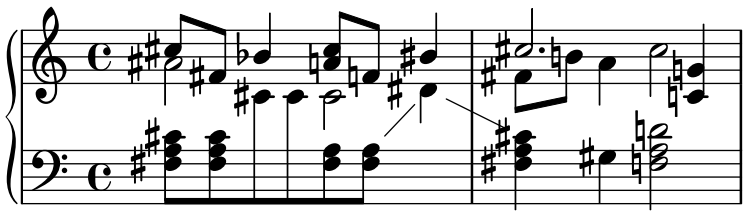


`piano`

Cette règle est communément employée pour les partitions de piano au XXe siècle. Très similaire à `modern` de par son comportement, elle s'en distingue en ce que

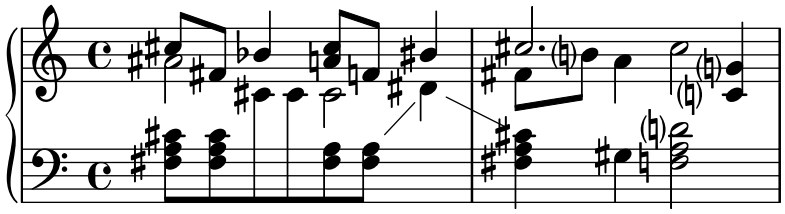
les altérations tiennent compte des autres portées du contexte `GrandStaff` ou `PianoStaff`.

Cette règle s'applique par défaut dans un `GrandStaff` et dans un `PianoStaff` à moins d'être qualifiée d'un second argument.



`piano-cautionary`

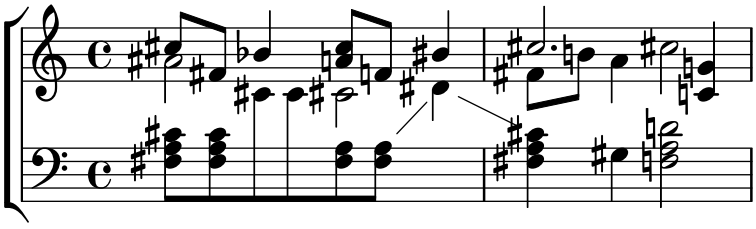
Identique au style `piano`, mais les altérations de précaution sont imprimées différemment.



`choral`

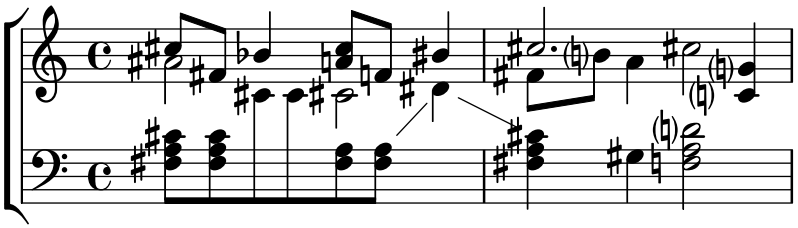
Cette règle est une combinaison des styles `modern-voice` et `piano`. Les altérations accidentelles sont indiquées aussi bien pour un chanteur qui suit seulement sa voix, que pour un lecteur suivant toutes les voix d'un `ChoirStaff`.

Ce style d'altération s'applique, par défaut, au `ChoirStaff` en cours à moins d'être qualifié d'un second argument.



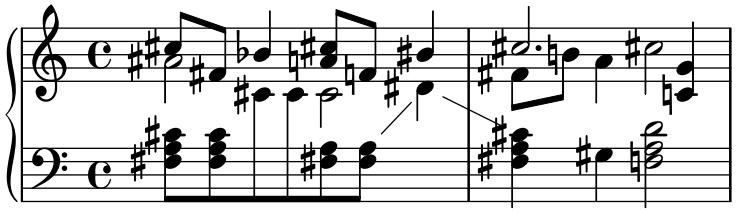
`choral-cautionary`

Identique au style `choral`, mais les altérations de précaution sont imprimées différemment.



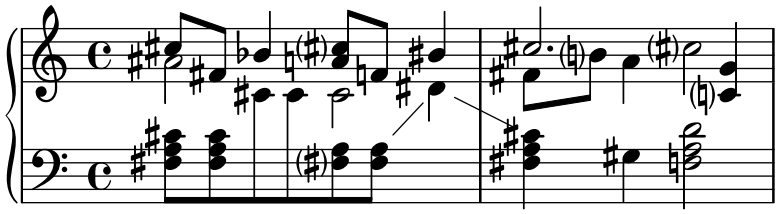
`neo-modern`

Cette règle suit les pratiques de la musique contemporaine : les altérations accidentelles apparaissent comme dans le style `modern`, à ceci près qu'elles sont répétées dans la même mesure – sauf si elles concernent deux notes consécutives.



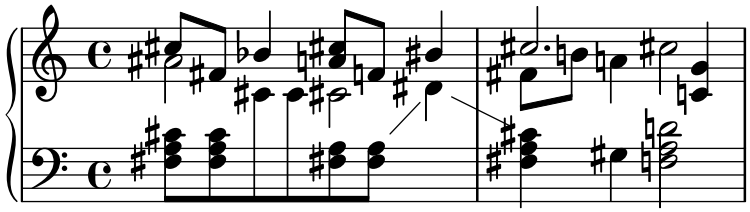
`neo-modern-cautionary`

Identique au style `neo-modern`, mais les altérations de précaution sont imprimées entre parenthèses. Elles peuvent aussi adopter une taille différente, au moyen de la propriété `font-size` de l'objet `AccidentalSuggestion`.



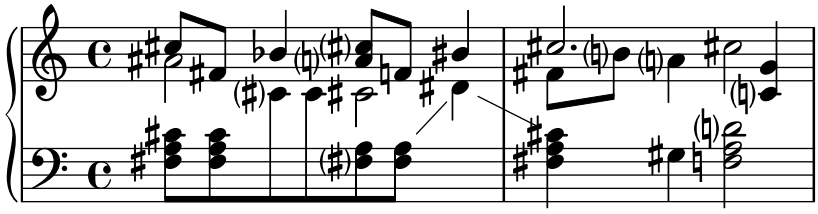
`neo-modern-voice`

Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix comme avec le style `neo-modern` mais les autres voix dans le même contexte `Staff` en tiennent aussi compte.



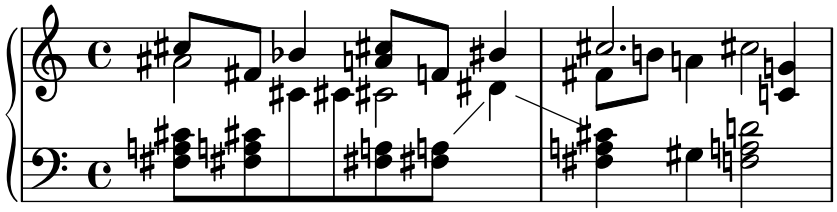
`neo-modern-voice-cautionary`

Cette règle est identique à `neo-modern-voice`, mais les altérations de précaution sont imprimées soit entre parenthèses (par défaut), soit en plus petit.



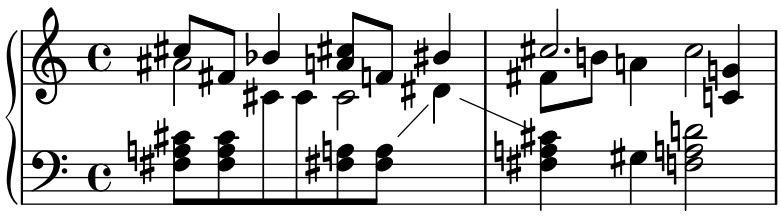
`dodecaphonic`

Cette règle reproduit ce que certains compositeurs du début du XXe siècle ont introduit dans leur désir d'abolir la distinction entre les notes naturelles ou non. Ainsi, **chaque** note est affublée d'une altération, même si elle est naturelle.



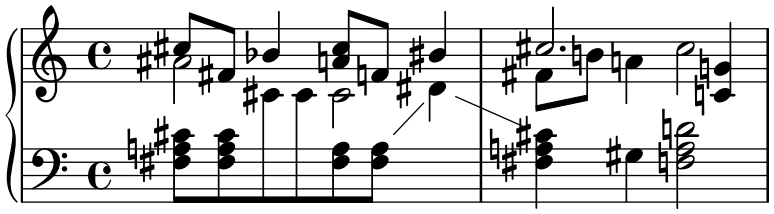
`dodecaphonic-no-repeat`

Comme dans le cas du style `dodecaphonic`, chaque note est par défaut affublée d'une altération. Celle-ci sera toutefois omise lorsque la même hauteur est immédiatement répétée dans la même portée.
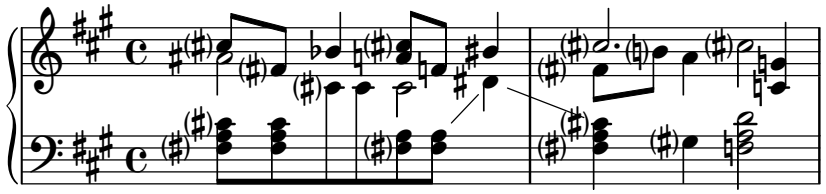


`dodecaphonic-first`

Comme dans le cas du style `dodecaphonic`, chaque note est par défaut affublée d'une altération. Cette altération n'apparaîtra que pour la première occurrence dans la mesure et seront répétés en cas d'octave différente.



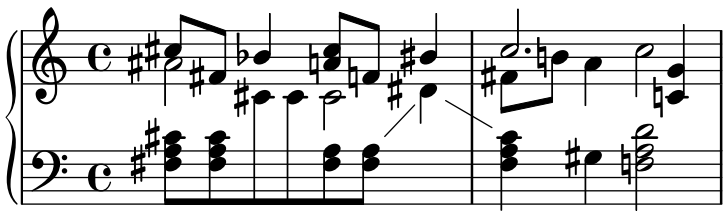
`teaching`

Cette règle est à usage pédagogique : l'impression d'une simple gamme fera apparaître une altération de précaution pour toute note altérée. Les altérations accidentelles sont imprimées selon le style `modern`, et une altération de précaution est ajoutée pour chaque dièse ou bémol à la clef – sauf dans le cas de notes consécutives.



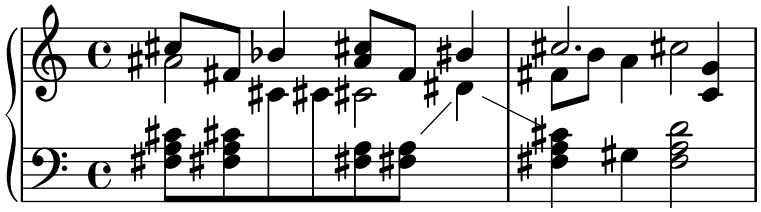
`no-reset`

C'est la même règle que `default`, mais l'effet des altérations accidentelles ne cesse jamais, même dans les mesures suivantes.



`forget`

Tout le contraire de `no-reset` : l'effet des altérations cesse aussitôt ; toutes les altérations, quelle que soit leur place dans la mesure, sont de ce fait imprimées en fonction de l'éventuelle armure.



Voir aussi
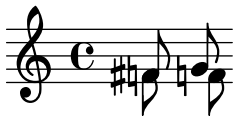
Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Accidental” dans *Référence des propriétés internes*, Section “Accidental_engraver” dans *Référence des propriétés internes*, Section “GrandStaff” dans *Référence des propriétés internes* et Section “PianoStaff” dans *Référence des propriétés internes*, Section “Staff” dans *Référence des propriétés internes*, Section “AccidentalSuggestion” dans *Référence des propriétés internes*, Section “AccidentalPlacement” dans *Référence des propriétés internes*, Section “accidental-suggestion-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

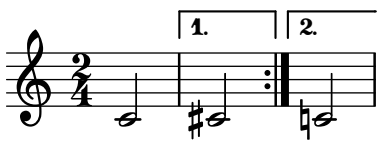
Les notes simultanées sont considérées comme des évènements séquentiels. Ceci implique que, dans un accord, les altérations accidentelles seront imprimées comme si les notes de l’accord apparaissaient l’une après l’autre, en fonction de l’ordre dans lequel elles ont été saisies – ce qui peut poser problème lorsqu’au sein d’un accord certaines altérations dépendent les unes des autres. Ce problème est à résoudre manuellement, en insérant des ‘!’ et des ‘?’ après les notes concernées, tel `<f! fis!>`.

```
\relative c' <<
  { fis8 g } \\
  { f! f }
>>
```



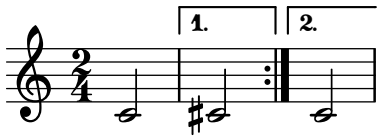
Une solution plus sophistiquée est proposée sur le LilyPond Wiki (`//wiki.lilypond.community/wiki/Accidental_adjustments_for_single-voice_polyphony`).

Dans le cas d’une reprise, la logique voudrait que la présence d’altération de précaution d’une alternative soit basée sur la dernière mesure *jouée* plutôt que la dernière *imprimée*. Dans l’exemple qui suit, vous conviendrez que le do de la seconde alternative ne nécessite pas son bécarre.



L’astuce suivante, qui définit temporairement le recours au style `forget`, permet d’obtenir quelque chose de présentable.

```
forget = #(define-music-function (music) (ly:music?) #{
  \accidentalStyle forget
  #music
  \accidentalStyle modern
#})
{
  \accidentalStyle modern
  \time 2/4
  \repeat volta 2 {
    c'2
  }
  \alternative {
    \volta 1 { cis' }
    \volta 2 { \forget c' }
  }
}
```
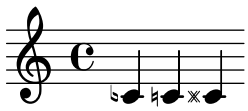


1.3.6 Glyphes d'altération alternatifs

Les systèmes de notation non occidentaux ou anciens disposent de leurs propres altérations. Leurs glyphes sont contrôlés par la propriété `alterationGlyphs` attachée à un contexte `Staff` ou équivalent. Les valeurs prédéfinies de cette propriété sont répertoriées à l'annexe Section B.10 [Jeux de glyphes d'altération], page 938.

```
\layout {
  \context {
    \Staff
    alterationGlyphs = #alteration-vaticana-glyph-name-alist
  }
}

{ ces' c' cis' }
```
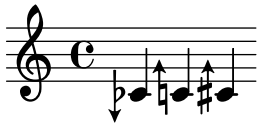


On peut également donner à cette propriété une valeur personnalisée sous la forme d'une liste associative affectant une altération à un nom de glyphe. Les altérations sont exprimées comme fractions d'un ton entier. Les différents glyphes sont répertoriés à l'annexe [Accidental glyphs], page 935.

```
\layout {
  \context {
    \Staff
    alterationGlyphs =
      #'((-1/2 . "accidentals.flat.arrowdown")
         (0 . "accidentals.natural.arrowup")
         (1/2 . "accidentals.sharp.arrowup"))
  }
```

```
    }

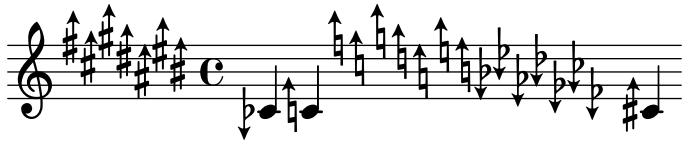
    { ces' c' cis' }
```



La propriété padding-pairs des objets KeySignature et KeyCancellation est une liste associative affectant à une paire de glyphes le décalage qui doit s'appliquer à ces glyphes lorsqu'ils apparaissent consécutivement dans l'armure.

```
\layout {
  \context {
    \Staff
    alterationGlyphs =
      #'((-1/2 . "accidentals.flat.arrowdown")
         (0 . "accidentals.natural.arrowup")
         (1/2 . "accidentals.sharp.arrowup"))
    \override KeySignature.padding-pairs =
      #'((("accidentals.sharp.arrowup" . "accidentals.sharp.arrowup")
            . 0.25)
         (("accidentals.flat.arrowdown" . "accidentals.flat.arrowdown")
            . 0.3))
    \override KeyCancellation.padding-pairs =
      #'((("accidentals.natural.arrowup" . "accidentals.natural.arrowup")
            . 0.7))
  }
}

{
  \key cis \major
  ces' c'
  \key ces \major
  cis'
}
```



Voir aussi

Manuel de notation : [Accidental glyphs], page 935, Section B.10 [Jeux de glyphes d'altération], page 938.

Référence des propriétés internes : Section "accidental-switch-interface" dans *Référence des propriétés internes*, Section "Alteration_glyph_engraver" dans *Référence des propriétés internes*, Section "key-signature-interface" dans *Référence des propriétés internes*.

1.3.7 Ambitus
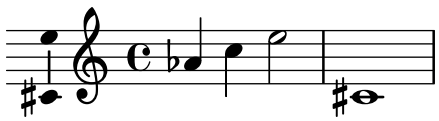
L'*ambitus* est l'amplitude des hauteurs d'une voix donnée dans une partition. Ce terme peut aussi désigner la tessiture qu'un instrument est capable d'atteindre. Souvent, cet ambitus est

imprimé au début des partitions vocales, afin que les exécutants puissent voir au premier coup d'œil s'ils sont en mesure de tenir la partie en question.

Pour exprimer l'ambitus d'une pièce, on indique avant la clef deux têtes de note représentant la hauteur la plus basse et la plus haute. Les éventuelles altérations accidentelles seront automatiquement ajoutées.

```
\layout {
  \context {
    \Voice
    \consists Ambitus_engraver
  }
}

\relative {
  aes' c e2
  cis,1
}
```
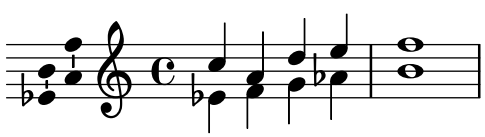


Morceaux choisis

Un ambitus par voix

L'ambitus peut être individualisé par voix. Il faut en pareil cas éviter qu'ils se chevauchent.
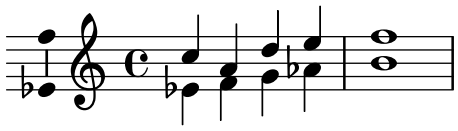
```
\new Staff <<
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus.X-offset = 2.0
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \with {
    \consists "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Ambitus sur plusieurs voix

Si plusieurs voix se trouvent sur une même portée, on peut attribuer le graveur Ambitus_engraver au contexte Staff afin d'obtenir l'ambitus sur toutes les voix cumulées, non d'une seule des voix actives.

```
\new Staff \with {
  \consists "Ambitus_engraver"
  }
<<
  \new Voice \relative c'' {
    \voiceOne
    c4 a d e
    f1
  }
  \new Voice \relative c' {
    \voiceTwo
    es4 f g as
    b1
  }
>>
```



Réglage de l'affichage d'un ambitus

L'affichage d'un *ambitus* peut s'affiner pour répondre à vos préférences en matière d'esthétique.

```
\layout {
  \context {
    \Voice
    \consists "Ambitus_engraver"
  }
}

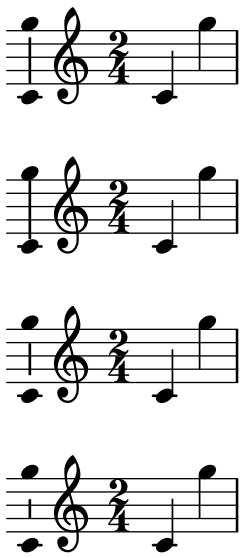
\new Staff {
  \time 2/4
  % Default setting
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = 0
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = 1
  c'4 g''
}

\new Staff {
  \time 2/4
  \override AmbitusLine.gap = 1.5
  c'4 g''
```
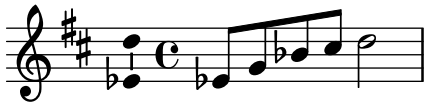
```
}
```



Ambitus après l'armure

L'ambitus se place par défaut à gauche de la clef. La fonction `\ambitusAfter` permet cependant de modifier ce positionnement ; sa syntaxe est `\ambitusAfter` *`grob-interface`* – voir Section "Graphical Object Interfaces" dans *Référence des propriétés internes* pour une liste des valeurs de *grob-interface* possibles.

L'un des cas d'usage est d'insérer l'ambitus entre l'armure et la métrique.

```
\new Staff \with {
  \consists Ambitus_engraver
} \relative {
  \ambitusAfter key-signature
  \key d \major
  es'8 g bes cis d2
}
```



Voir aussi

Glossaire musicologique : Section "ambitus" dans *Glossaire*.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Ambitus_engraver" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*, Section "Ambitus" dans *Référence des propriétés internes*, Section "AmbitusAccidental" dans *Référence des propriétés internes*, Section "AmbitusLine" dans *Référence des propriétés internes*, Section "AmbitusNoteHead" dans *Référence des propriétés internes*, Section "ambitus-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

LilyPond ne gère pas les collisions entre plusieurs ambitus présents sur une même portée.
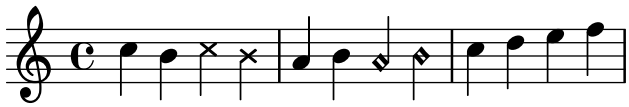
1.4 Têtes de note

Nous allons voir dans ce chapitre comment modifier l'aspect des têtes de note.

1.4.1 Têtes de note spécifiques

L'apparence des têtes de note peut évoluer au cours de la partition :

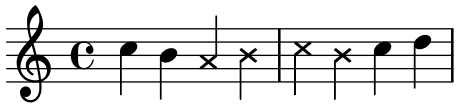
```
\relative c'' {
  c4 b
  \override NoteHead.style = #'cross
  c4 b
  \revert NoteHead.style
  a b
  \override NoteHead.style = #'harmonic
  a b
  \revert NoteHead.style
  c4 d e f
}
```



Pour une liste exhaustive des styles de tête de note, consultez Section B.9 [Styles de tête de note], page 937.

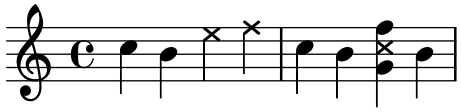
Certains instruments utilisent des têtes de note différentes à des fins spécifiques – des croix (style `cross`) pour le *parlato* des chanteurs ou les notes étouffées des guitaristes :

```
\relative {
  c''4 b
  \xNotesOn
   a b c4 b
  \xNotesOff
  c4 d
}
```



Cette commande opère aussi bien sur des notes isolées qu'au sein d'un accord, dans une portée traditionnelle ou dans un contexte de tablature :
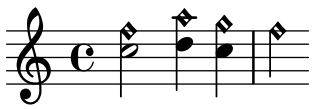
```
\relative {
  c''4 b
  \xNote { e f }
  c b < g \xNote c f > b
}
```



Vous pouvez utiliser, en lieu et place de `\xNote`, `\xNotesOn` et `\xNotesOff`, les commandes `\deadNote`, `\deadNotesOn` et `\deadNotesOff`.

Il existe un raccourci pour les notes en losange :

```
\relative c'' {
  <c f\harmonic>2 <d a'\harmonic>4 <c g'\harmonic> f\harmonic
}
```



Commandes prédéfinies

`\harmonic`, `\xNotesOn`, `\xNotesOff`, `\xNote`.

Voir aussi

Manuel de notation : Section 12.2.2 [Indication des harmoniques et notes étouffées], page 495, Section 5.1.1 [Notes en accords], page 221, Section B.9 [Styles de tête de note], page 937.

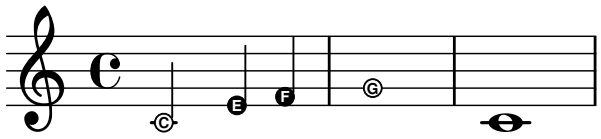
Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “note-event” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “Ledger_line_engraver” dans *Référence des propriétés internes*, Section “NoteHead” dans *Référence des propriétés internes*, Section “LedgerLineSpanner” dans *Référence des propriétés internes*, Section “note-head-interface” dans *Référence des propriétés internes*, Section “ledger-line-spanner-interface” dans *Référence des propriétés internes*.

1.4.2 Têtes de note avec nom de note

Les notes « easy play » comportent le nom de la note à l’intérieur de la tête. On l’utilise dans des partitions pour débutants. L’impression doit être de plus grande taille, afin que les lettres soient lisibles. Voir à ce propos Section 27.2 [Définition de la taille de portée], page 707.

```
#(set-global-staff-size 26)
\relative c' {
  \easyHeadsOn
  c2 e4 f
  g1
  \easyHeadsOff
  c,1
}
```



Commandes prédéfinies

`\easyHeadsOn`, `\easyHeadsOff`.

Morceaux choisis

Easy play – chiffres en lieu et place des lettres

En mode « easy play », les têtes de note utilisent la propriété `note-names` attachée à l’objet `NoteHead` pour déterminer ce qui apparaîtra dans la tête. Intervenir sur cette propriété permet d’imprimer un chiffre correspondant au degré dans la gamme.

La création d’un graveur dédié permet de traiter toutes les notes.

```
#(define Ez_numbers_engraver
   (make-engraver
    (acknowledgers
     ((note-head-interface engraver grob source-engraver)
      (let* ((context (ly:translator-context engraver))
             (tonic-pitch (ly:context-property context 'tonic))
             (tonic-name (ly:pitch-notename tonic-pitch))
             (grob-pitch
              (ly:event-property (event-cause grob) 'pitch))
             (grob-name (ly:pitch-notename grob-pitch))
             (delta (modulo (- grob-name tonic-name) 7))
             (note-names
              (make-vector 7 (number->string (1+ delta)))))
        (ly:grob-set-property! grob 'note-names note-names))))))

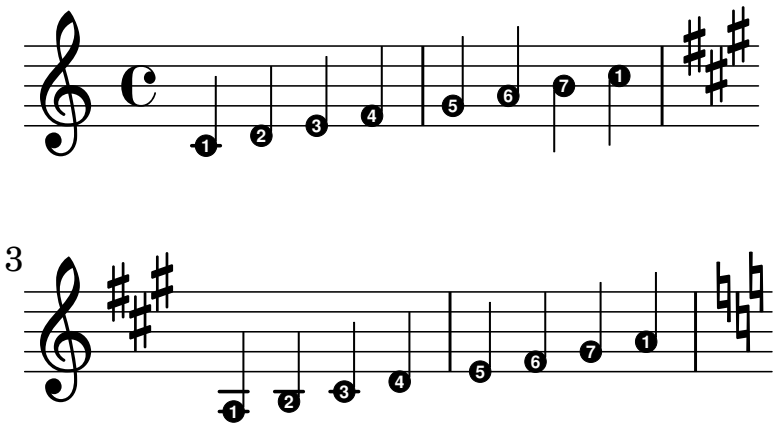
#(set-global-staff-size 30)

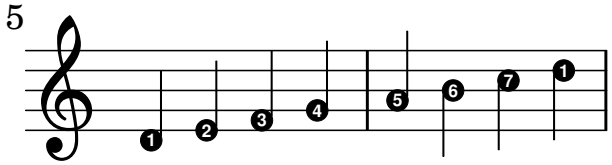
\layout {
  ragged-right = ##t
  \context {
    \Voice
    \consists \Ez_numbers_engraver
  }
}

\relative c' {
  \easyHeadsOn
  c4 d e f
  g4 a b c \break

  \key a \major
  a,4 b cis d
  e4 fis gis a \break

  \key d \dorian
  d,4 e f g
  a4 b c d
}
```





Voir aussi

Manuel de notation : Section 27.2 [Définition de la taille de portée], page 707.
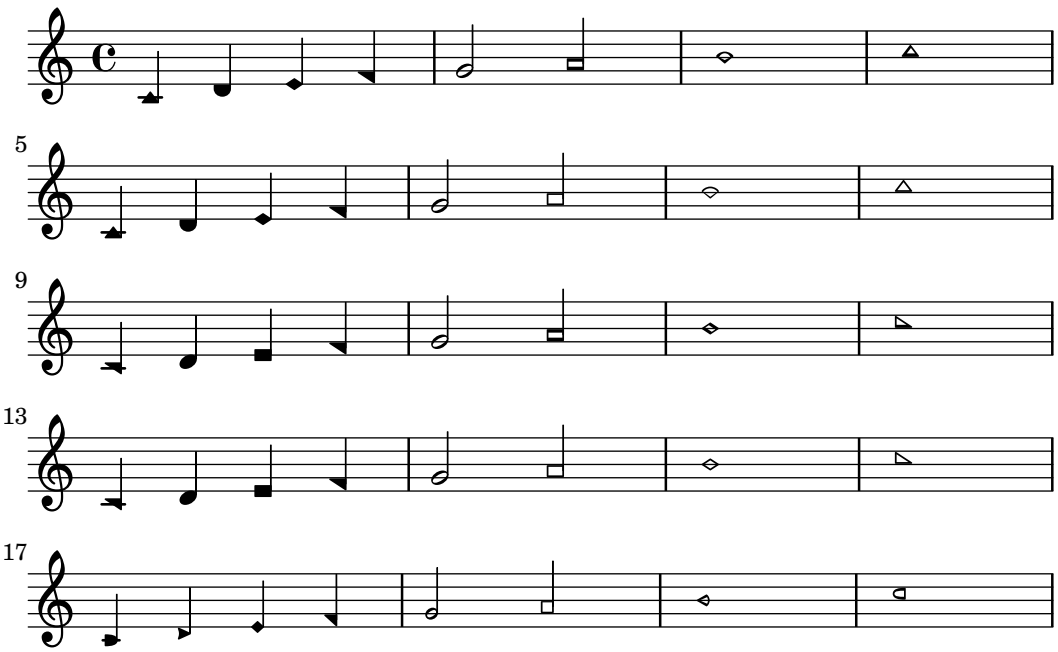
Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

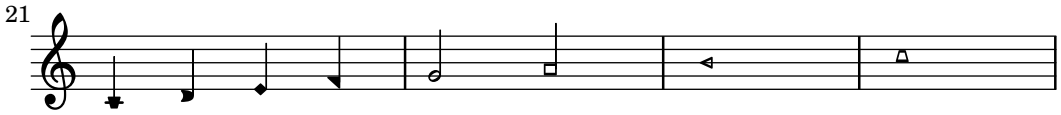
Référence des propriétés internes : Section “note-event” dans *Référence des propriétés internes*, Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “NoteHead” dans *Référence des propriétés internes*, Section “note-head-interface” dans *Référence des propriétés internes*.

1.4.3 Têtes de note à forme variable

En notation profilée, le profil d’une tête de note correspond à la fonction harmonique de cette note dans la gamme. Ce style de notation était très en vogue dans les recueils de chansons américains du XIXe siècle. Voici comment procéder :
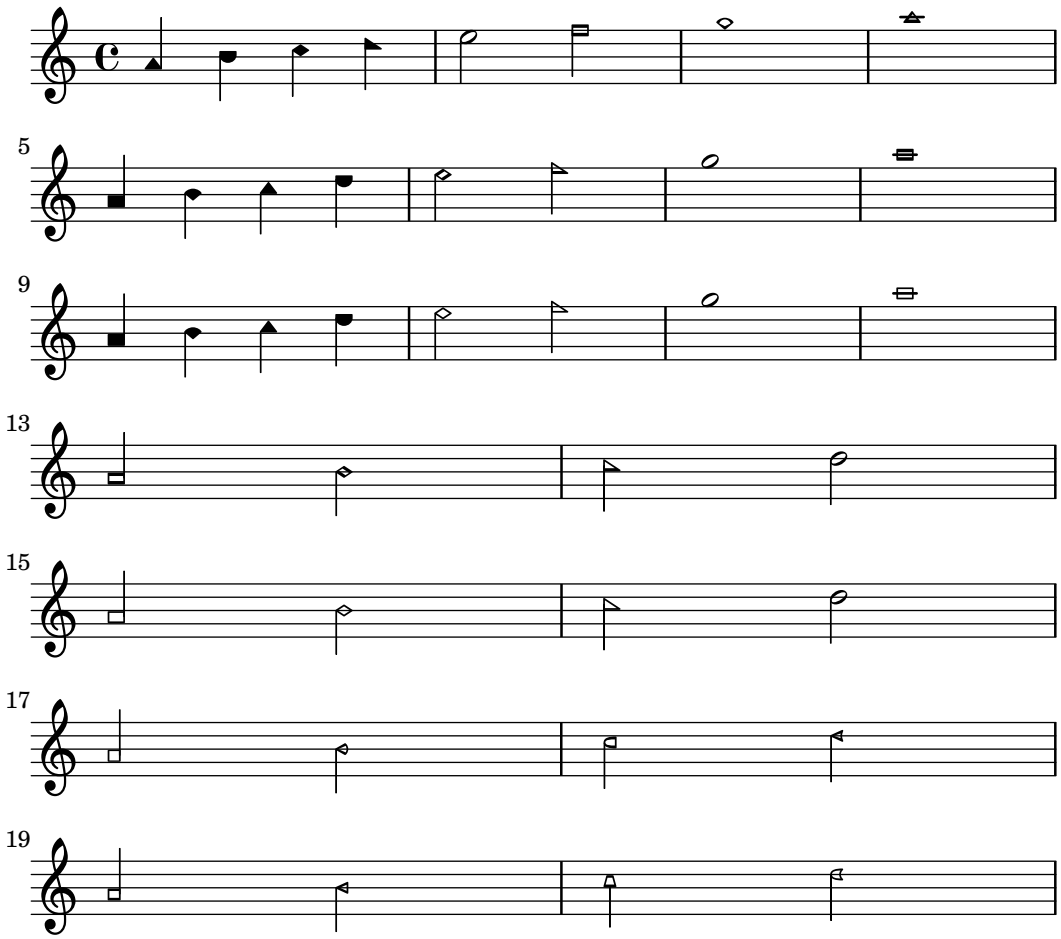
```
\relative c'' {
  \aikenHeads
  c, d e f g2 a b1 c \break
  \aikenThinHeads
  c,4 d e f g2 a b1 c \break
  \sacredHarpHeads
  c,4 d e f g2 a b1 c \break
  \southernHarmonyHeads
  c,4 d e f g2 a b1 c \break
  \funkHeads
  c,4 d e f g2 a b1 c \break
  \walkerHeads
  c,4 d e f g2 a b1 c \break
}
```





Les profils sont déterminés par la hauteur dans la gamme, le premier degré étant défini par la commande \key. Pour une tonalité mineure, les degrés sont déterminés par rapport au relatif majeur :

```
\relative c'' {
  \key a \minor
  \aikenHeads
  a b c d e2 f g1 a \break
  \aikenHeadsMinor
  a,4 b c d e2 f g1 a \break
  \aikenThinHeadsMinor
  a,4 b c d e2 f g1 a \break
  \sacredHarpHeadsMinor
  a,2 b c d \break
  \southernHarmonyHeadsMinor
  a2 b c d \break
  \funkHeadsMinor
  a2 b c d \break
  \walkerHeadsMinor
  a2 b c d \break
}
```



Commandes prédéfinies

\aikenHeads, \aikenHeadsMinor, \aikenThinHeads, \aikenThinHeadsMinor, \funkHeads, \funkHeadsMinor, \sacredHarpHeads, \sacredHarpHeadsMinor, \southernHarmonyHeads, \southernHarmonyHeadsMinor, \walkerHeads, \walkerHeadsMinor.
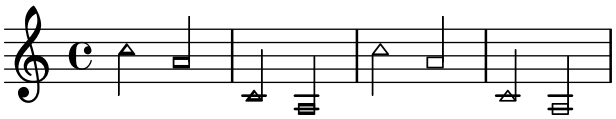
Morceaux choisis

Variante fine des notes profilées Aiken

Les notes profilées Aiken, lorsqu'elles sont blanches, deviennent difficile à distinguer avec des tailles de portée réduites, notamment en présence de ligne supplémentaire. Perdant du blanc sur leur intérieur les fait alors ressembler à des noires.

```
\score {
  {
    \aikenHeads
    c''2 a' c' a

    % Switch to thin-variant noteheads
    \set shapeNoteStyles = ##(doThin reThin miThin
                              faThin sol laThin tiThin)
    c'' a' c' a
  }
}
```
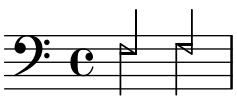


Orientation de la tête d'un « fa » profilé en cas de fusion

Grâce à la propriété NoteCollision.fa-merge-direction, l'orientation de la tête des notes profilées « fa » (fa, faThin, etc.) peut se gérer indépendamment de la direction des hampes lorsque deux voix ayant la même hauteur et des directions de hampes différentes sont fusionnées. Lorsque cette propriété n'est pas activée sera utilisée la variante « descendante ».

```
{
  \clef bass

  << { \aikenHeads
       f2
       \override Staff.NoteCollision.fa-merge-direction = #UP
       f2 }
  \\ { \aikenHeads
       f2
       f2 }
  >>
}
```



Profilage des notes selon leur degré dans la gamme

La propriété shapeNoteStyles permet d'affecter un profil particulier à chaque degré de la gamme – à partir de l'armure ou de la propriété tonic. Ses valeurs sont constituées d'une liste

de symboles, qu'il s'agisse de formes géométriques (`triangle`, `cross` ou `xcircle`) ou basés sur la tradition des graveurs américains (avec quelques noms de note latins).

LilyPond dispose de deux raccourcis, `\aikenHeads` et `\sacredHarpHeads`, permettant de reproduire d'anciens recueils de chansons américaines.

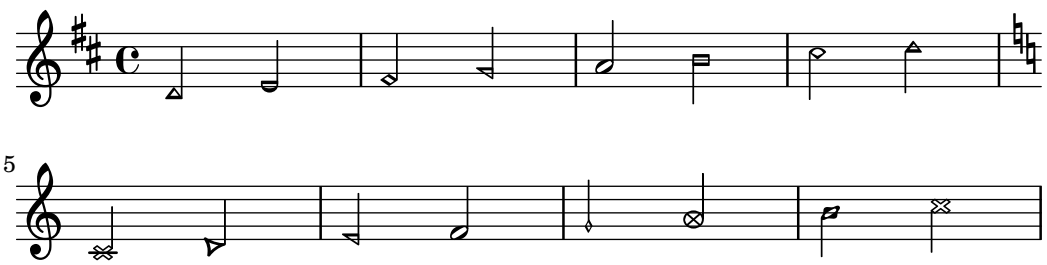
L'exemple suivant montre plusieurs manières de profiler les têtes de note, ainsi que la capacité de transposer tout en respectant la fonction harmonique de chaque note dans la gamme.

```
fragment = {
  \key c \major
  c2 d
  e2 f
  g2 a
  b2 c
}

\new Staff {
  \transpose c d
  \relative c' {
    \set shapeNoteStyles = ##(do re mi fa
                              #f la ti)
    \fragment
  }

  \break

  \relative c' {
    \set shapeNoteStyles = ##(cross triangle fa #f
                              mensural xcircle diamond)
    \fragment
  }
}
```



Pour une liste exhaustive des styles de tête de note, consultez Section B.9 [Styles de tête de note], page 937.

Voir aussi

Manuel de notation : Section B.9 [Styles de tête de note], page 937.

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "note-event" dans *Référence des propriétés internes*, Section "Note_heads_engraver" dans *Référence des propriétés internes*, Section "NoteHead" dans *Référence des propriétés internes*, Section "note-head-interface" dans *Référence des propriétés internes*.

1.4.4 Improvisation

L'improvisation peut quelquefois s'indiquer à l'aide de notes de forme allongée (*slash*). L'interprète jouera alors les notes qu'il veut, en respectant toutefois le rythme affiché. Ces têtes de notes sont créées ainsi :

```
\new Voice \with {
  \consists Pitch_squash_engraver
} \relative {
  e''8 e g a a16( bes) a8 g
  \improvisationOn
  e8 ~
  2 ~ 8 f4 f8 ~
  2
  \improvisationOff
  a16( bes) a8 g e
}
```



Commandes prédéfinies
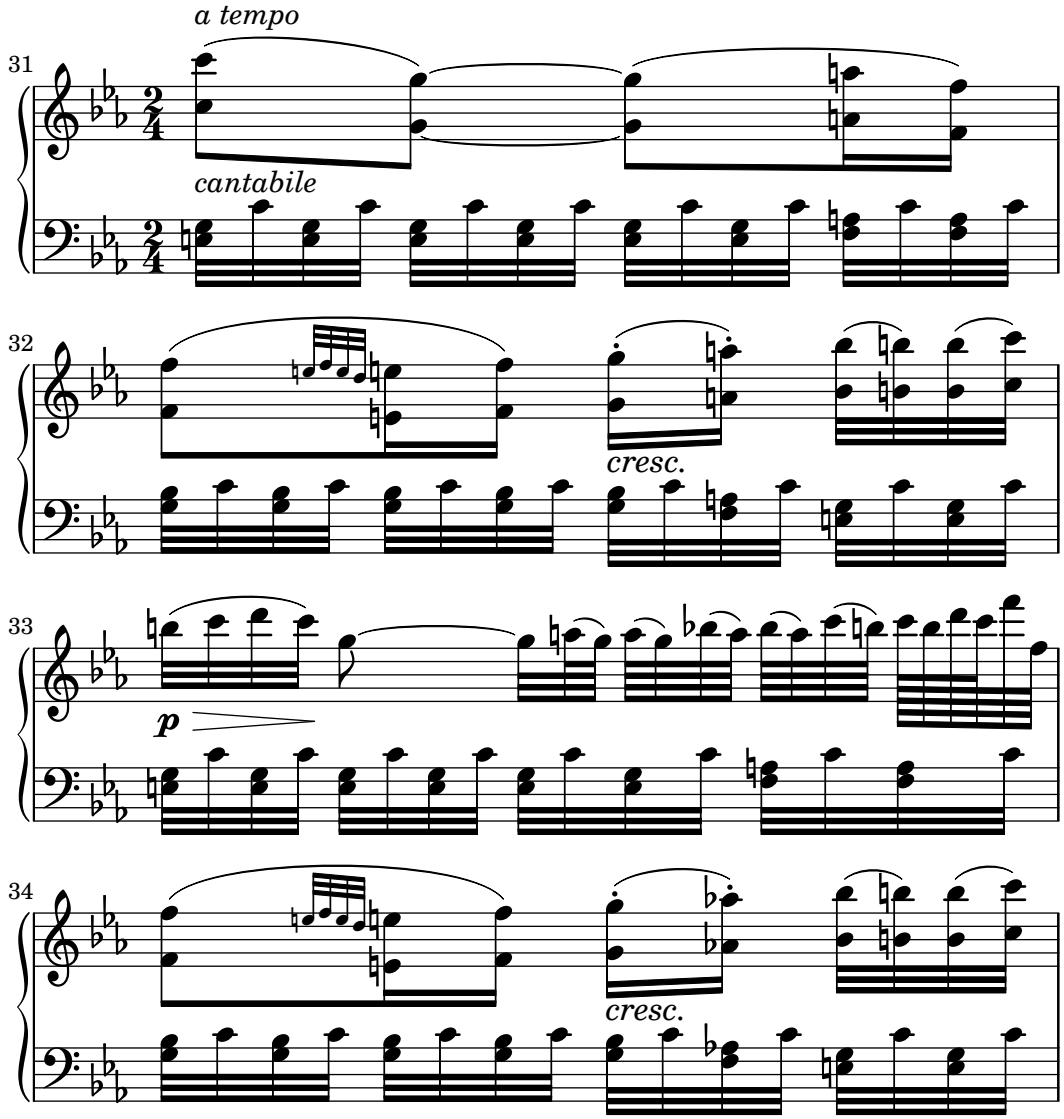
`\improvisationOn`, `\improvisationOff`.

Voir aussi

Morceaux choisis : Section "Hauteurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Pitch_squash_engraver" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*, Section "RhythmicStaff" dans *Référence des propriétés internes*.

2 Rythme



Cette section traite du rythme : durées, silences, barres de ligature et de mesure.
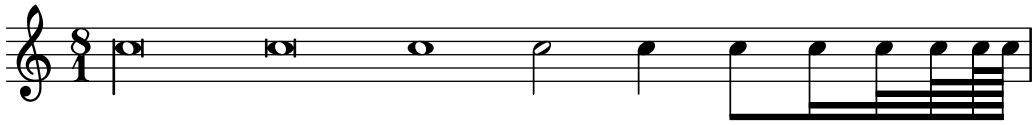
2.1 Écriture du rythme

2.1.1 Durées

Dans les modes de notes, d'accords et de paroles, les durées sont écrites avec des chiffres et des points : les durées sont indiquées par leur valeur fractionnaire par rapport à la durée d'une ronde. Une noire, par exemple, qui équivaut à un 1/4 de ronde – *quarter note* en anglais – s'écrit 4, alors qu'une blanche – *half-note*, 1/2 ronde – s'écrit 2. Des durées plus courtes que la quintuple croche – 1/128 de ronde – sont possibles, à condition de les ligaturer – voir Section 2.4 [Barres de ligature], page 104.
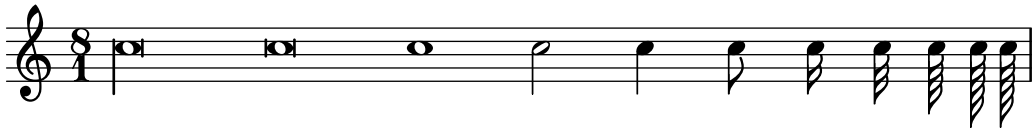
Pour des notes plus longues qu'une ronde, vous devrez utiliser les commandes `\longa` pour une longue, et `\breve` pour une brève, aussi appelée carrée. Une note dont la durée est de quatre brèves s'obtient par la commande `\maxima` ; celle-ci n'est toutefois disponible que dans le cadre de la notation ancienne. Pour plus de détails, voir Chapitre 17 [Notations anciennes], page 552.

```
\relative {
  \time 8/1
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```
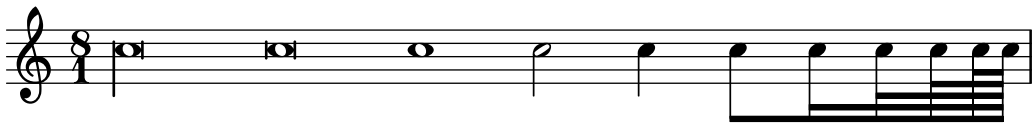


Voici ces mêmes durées sans la fonction de ligature automatique.

```
\relative {
  \time 8/1
  \autoBeamOff
  c''\longa c\breve c1 c2
  c4 c8 c16 c32 c64 c128 c128
}
```
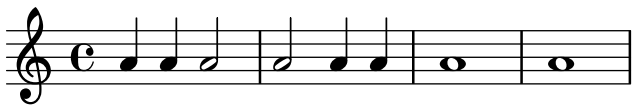


Lorsque une note ou un accord est suivi d'une succession de durées individuelles, celles-ci adopteront la ou les dernières hauteurs mentionnées.

```
\relative {
  \time 8/1
  c'' \longa \breve 1 2
  4 8 16 32 64 128 128
}
```
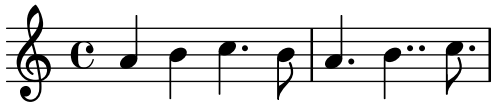


Si la durée d'une note n'est pas précisée, elle est alors assimilée à la durée de la note précédente. La valeur par défaut pour la première note est la noire (4).

```
\relative { a' a a2 a a4 a a1 a }
```



Pour obtenir des notes pointées, ajoutez simplement un point (.) au chiffre. Les notes doublement pointées sont créées de la même façon avec un deux points (..).

```
\relative { a'4 b c4. b8 a4. b4.. c8. }
```



Les points sont normalement haussés pour éviter les lignes de portée, sauf dans certaines polyphonies. Des commandes prédéfinies permettent de manuellement forcer un positionnement particulier, comme indiqué au chapitre Section 36.1 [Direction et positionnement], page 801.

Certaines durées ne peuvent s'obtenir à partir de chiffres et de points, mais uniquement en « liant » deux ou plusieurs notes entre elles. Voir Section 2.1.4 [Liaisons de prolongation], page 64, à ce sujet.

Quant à la manière de spécifier la durée des syllabes ou bien d'aligner des paroles par rapport aux notes, reportez vous au chapitre Chapitre 9 [Musique vocale], page 355.

Espacer les notes selon leur durée relative est tout à fait possible. Pour plus de détails à ce sujet et sur les autres réglages propres à cette forme de notation, reportez vous à Section 30.6 [Notation proportionnelle], page 746.

Commandes prédéfinies

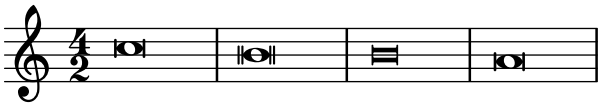
`\autoBeamOn`, `\autoBeamOff`, `\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Morceaux choisis

Brève alternative, avec deux barres verticales

Voici comment obtenir une brève – aussi appelée note carée – flanquée de deux barres verticales, au lieu d'une comme habituellement.
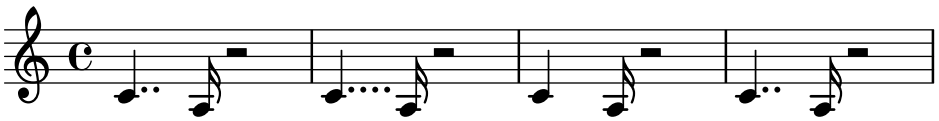
```
\relative c'' {
  \time 4/2
  c\breve |
  \override Staff.NoteHead.style = #'altdefault
  b\breve
  \override Staff.NoteHead.style = #'baroque
  b\breve
  \revert Staff.NoteHead.style
  a\breve
}
```



Spécification du nombre de points d'augmentation d'une note

Le nombre de points d'augmentation affectés à une note en particulier peut se modifier indépendamment des points placés après la note, à l'aide de la propriété `dot-count` de l'objet `Dots`.

```
\relative c' {
  c4.. a16 r2 |
  \override Dots.dot-count = 4
  c4.. a16 r2 |
  \override Dots.dot-count = 0
  c4.. a16 r2 |
  \revert Dots.dot-count
  c4.. a16 r2 |
}
```



Voir aussi

Glossaire musicologique : Section “breve” dans *Glossaire*, Section “longa” dans *Glossaire*, Section “maxima” dans *Glossaire*, Section “valeur des notes” dans *Glossaire*, Section “Noms de durée (notes et silences)” dans *Glossaire*.

Manuel de notation : Section 2.4 [Barres de ligature], page 104, Section 2.1.4 [Liaisons de prolongation], page 64, Section 7.1.9 [Hampes], page 306, Section 2.1 [Écriture du rythme], page 53, Section 2.2 [Écriture des silences], page 68, Chapitre 9 [Musique vocale], page 355, Chapitre 17 [Notations anciennes], page 552, Section 30.6 [Notation proportionnelle], page 746.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Dots” dans *Référence des propriétés internes*, Section “DotColumn” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Il n’existe pas à proprement parler de limite (inférieure ou supérieure) en terme de durée d’un silence. Cependant, le nombre de glyphes disponibles ne couvre que les silences allant de 1 024e de pause (256e de soupir) à la maxime (valant huit pauses).

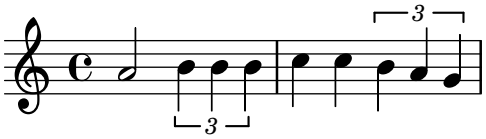
2.1.2 N-olets

Les n-olets – triolets, quintolets, etc. – sont obtenus en multipliant la vitesse d’une expression musicale par une fraction.

```
\tuplet fraction { expression_musicale }
```
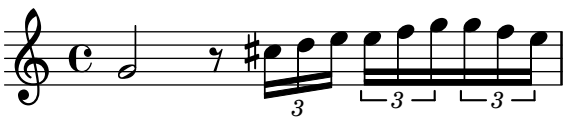
Le numérateur de cette fraction sera imprimé au-dessus ou au-dessous des notes, parfois avec un crochet. Le n-olet le plus courant est le triolet, dans lequel trois notes occupent la durée de deux.

```
\relative {
  a'2 \tuplet 3/2 { b4 4 4 }
  c4 c \tuplet 3/2 { b4 a g }
}
```



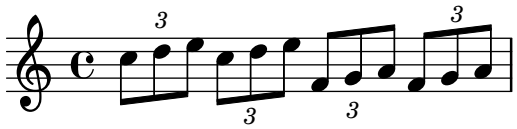
Dans le cas d’une succession de n-olets, saisir la commande `\tuplet` pour chacun des n-olets devient vite fastidieux. LilyPond permet de stipuler la durée de base d’un n-olet juste avant l’expression musicale, de telle sorte que les n-olets seront formés automatiquement :

```
\relative {
  g'2 r8 \tuplet 3/2 8 { cis16 d e e f g g f e }
}
```
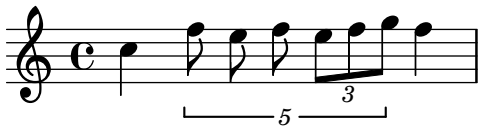


Des commandes prédéfinies permettent de déroger au positionnement automatique du crochet en surplomb ou au-dessous des notes :

```
\relative {
  \tupletUp \tuplet 3/2 { c''8 d e }
  \tupletNeutral \tuplet 3/2 { c8 d e }
  \tupletDown \tuplet 3/2 { f,8 g a }
  \tupletNeutral \tuplet 3/2 { f8 g a }
}
```



Les n-olets peuvent être imbriqués comme ici par exemple :
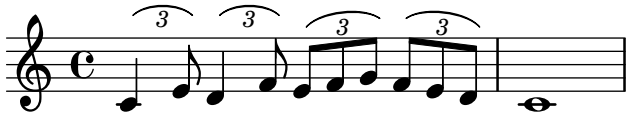
```
\autoBeamOff
c4 \tuplet 5/4 { f8 e f \tuplet 3/2 { e[ f g] } } f4 |
```



Lorsque, dans une imbrication, les n-olets débutent au même instant, il faut recourir à la commande `\tweak` – voir Section 35.6 [Les commandes \tweak et \single], page 791.

Le crochet de n-olet peut être remplacé par une liaison, à l'instar des éditions anciennes :

```
\relative {
  \tuplet 3/2 4 {
    \override TupletBracket.tuplet-slur = ##t
    c'4 e8 d4 f8
    \override TupletBracket.bracket-visibility = ##t
    e f g f e d
  } c1
}
```



Un crochet sera imprimé en l'absence de ligature couvrant toute sa longueur. Dans certains cas cependant, par exemple en présence d'une liaison comme ci-dessus, il peut être préférable de modifier ce comportement à l'aide de la propriété `bracket-visibility` comme indiqué dans les exemples qui suivent.

De manière générale, aussi bien les objets `TupletBracket` que `TupletNumber` peuvent se masquer comme indiqué dans Section 36.7 [Visibilité des objets], page 812 ; on peut néanmoins interférer sur la durée des notes sans imprimer de crochet, comme indiqué au chapitre Section 2.1.3 [Changement d'échelle des durées], page 62.

Commandes prédéfinies

`\tupletUp`, `\tupletDown`, `\tupletNeutral`.

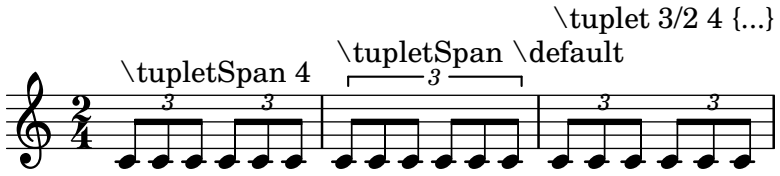
Morceaux choisis

Plusieurs triolets avec une seule commande `\tuplet`

La propriété `tupletSpannerDuration` spécifie la longueur voulue de chaque crochet. Avec elle, vous pouvez faire plusieurs n-olets en ne tapant `\tuplet` qu'une fois, ce qui évite une longue saisie.

Il existe différents moyens de définir `tupletSpannerDuration`. La commande `\tupletSpan` lui affecte une durée arbitraire qui sera réinitialisée dès l'intervention d'une durée à `\default`. Vous pouvez aussi opter pour fournir un argument supplémentaire à la commande `\tuplet`.

```
\relative c' {
  \time 2/4
  \tupletSpan 4
  \tuplet 3/2 { c8^"\\tupletSpan 4" c c c c c }
  \tupletSpan \default
  \tuplet 3/2 { c8^"\\tupletSpan \\default" c c c c c }
  \tuplet 3/2 4 { c8^"\\tuplet 3/2 4 {...}" c c c c c }
}
```

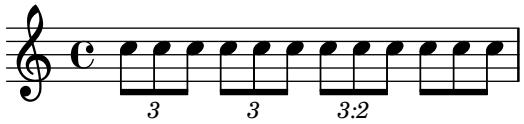


Modifier l'apparence du chiffre de n-olet

LilyPond imprime par défaut le numérateur de la fraction fournie en argument à la commande `\tuplet` du côté du crochet de n-olet.

Il est toutefois possible d'imprimer la fraction entière *num*:*den*, voire de ne rien imprimer du tout.

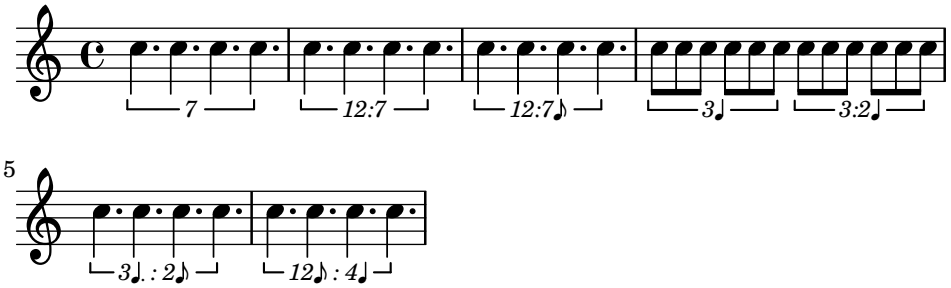
```
\relative c'' {
  \tuplet 3/2 { c8 c c }
  \tuplet 3/2 { c8 c c }
  \override TupletNumber.text = #tuplet-number::calc-fraction-text
  \tuplet 3/2 { c8 c c }
  \omit TupletNumber
  \tuplet 3/2 { c8 c c }
}
```



N-olets au chiffrage inhabituel

LilyPond sait aussi gérer des n-olets dont le chiffrage imprimé ne correspond pas exactement à la fraction de mesure à laquelle ils se réfèrent, tout comme ceux auxquels une valeur de note vient en complément du chiffre.

```
\relative c'' {
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-denominator-text 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-tuplet-fraction-text 12 7)
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      (tuplet-number::non-default-tuplet-fraction-text 12 7)
      (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-denominator-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::append-note-wrapper
      tuplet-number::calc-fraction-text
      (ly:make-duration 2 0))
  \tuplet 3/2  { c8 c8 c8 c8 c8 c8 }
  \once \override TupletNumber.text =
    #(tuplet-number::fraction-with-notes
      (ly:make-duration 2 1) (ly:make-duration 3 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
  \once \override TupletNumber.text =
    #(tuplet-number::non-default-fraction-with-notes 12
      (ly:make-duration 3 0) 4 (ly:make-duration 2 0))
  \tuplet 3/2  { c4. c4. c4. c4. }
}
```



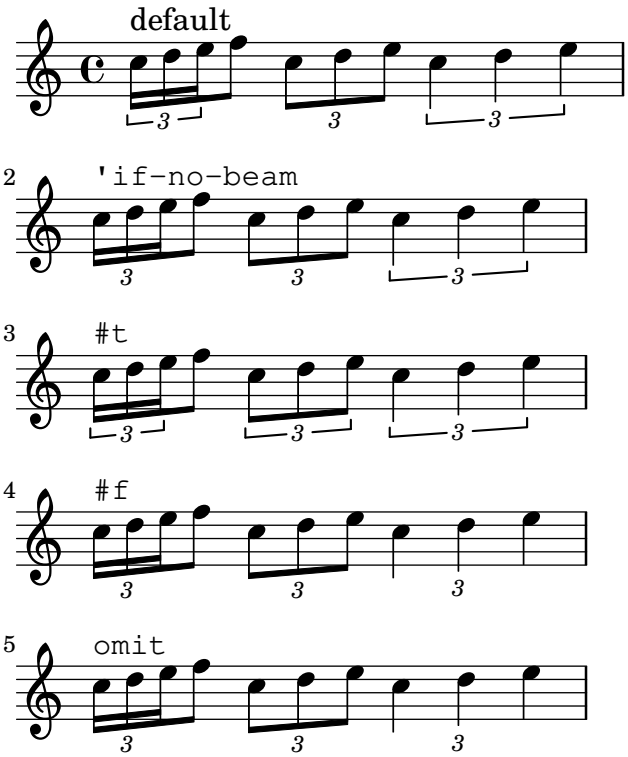
Contrôle de l'impression des crochets de n-olet

Selon la tradition, les crochets indicateurs de n-olet sont toujours imprimés, sauf dans le cas où ils seraient de la même longueur qu'une ligature.

LilyPond permet, au travers de la propriété `bracket-visibility`, de contôler précisément leur affichage : déterminée à `#t`, ils seront toujours imprimés ; `#f` permet de ne jamais les imprimer – donc omettre l'objet `TupletBracket` –, et `if-no-beam` les imprimera en l'absence de ligature (comportement par défaut).

```
music = \relative c'' {
  \tuplet 3/2 { c16[ d e } f8]
  \tuplet 3/2 { c8 d e }
  \tuplet 3/2 { c4 d e }
}

\new Voice {
  \relative c' {
    \override Score.TextMark.non-musical = ##f
    \textMark "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    \textMark \markup \typewriter "'if-no-beam" \music
    \override TupletBracket.bracket-visibility = ##t
    \textMark \markup \typewriter "#t" \music
    \override TupletBracket.bracket-visibility = ##f
    \textMark \markup \typewriter "#f" \music
    \omit TupletBracket
    \textMark \markup \typewriter "omit" \music
  }
}
```
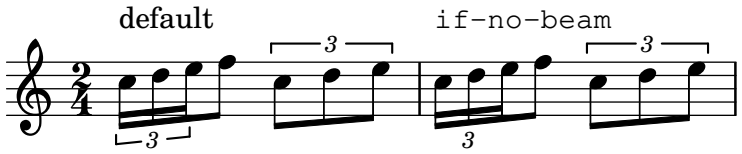


Affichage du crochet de n-olet du côté des têtes de note

Quelle que soit l'option choisie pour contrôler la visibilité d'une indication de n-olet, elle s'affichera ou sera masquée indépendamment de son positionnement (côté hampe ou tête de note). Toutefois, lorsqu'elle se place du côté des têtes de note, certains auteurs recommandent de toujours afficher un crochet. L'option `visible-over-note-heads` permet d'y parvenir.

```
music = \relative c'' {
  \tupletNeutral \tuplet 3/2 { c16[ d e } f8]
  \tupletUp \tuplet 3/2 { c8 d e }
}

\new Voice {
  \relative c' {
    \override TextScript.staff-padding = #2.5

    \time 2/4
    \override TupletBracket.visible-over-note-heads = ##t
    \override Score.TextMark.non-musical = ##f
    <>^\markup "default" \music
    \override TupletBracket.bracket-visibility = #'if-no-beam
    <>^\markup \typewriter "if-no-beam" \music
  }
}
```



Saut de ligne au milieu d'un n-olet avec ligature

Ces exemples peu académiques démontrent comment il est possible d'insérer un saut de ligne dans un n-olet portant une ligature et qui ne sauraient se découper de manière exacte.

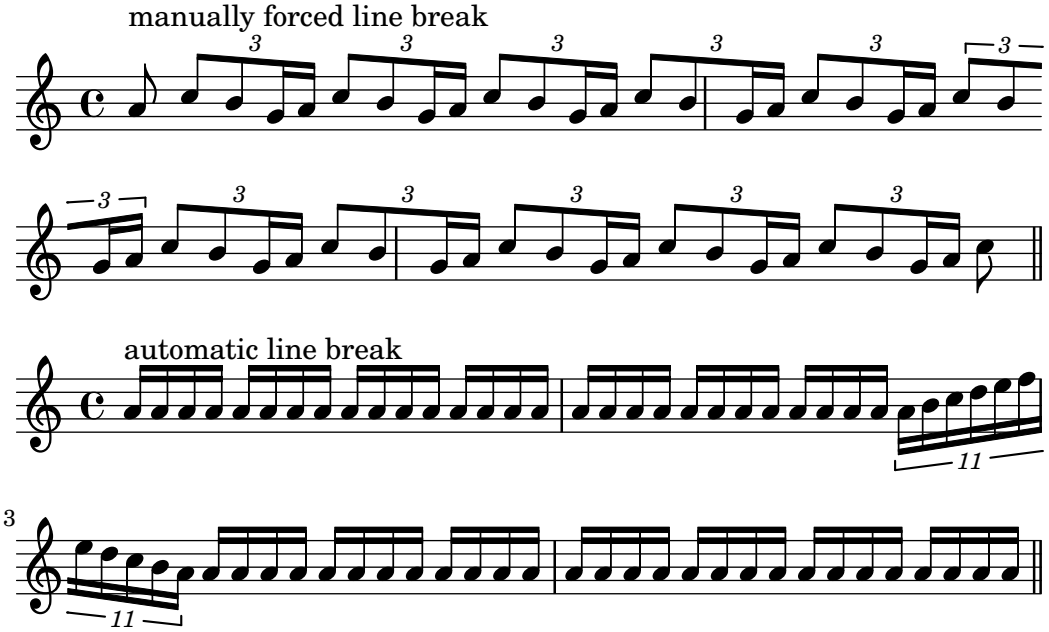
Cette fonctionnalité n'est opérationnelle qu'avec des ligatures manuelles.

```
\layout {
  \context {
    \Voice
    % Permit automatic line breaks within tuplets.
    \remove "Forbid_line_break_engraver"
    % Allow beams to be broken at line breaks.
    \override Beam.breakable = ##t
  }
}

\relative c'' {
  <>^"manually forced line break"
  a8
  \*5 { \tuplet 3/2 { c8[ b g16 a] } }
  \tuplet 3/2 { c8[ b \break g16 a] }
  \*5 { \tuplet 3/2 { c8[ b g16 a] } }
  c8 \bar "||"
}

\relative c'' {
  <>^"automatic line break"
  \*28 a16
  \tuplet 11/8 { a16[ b c d e f e d c b a] }
  \*28 a16 \bar "||"
```

```
}
```



Voir aussi

Glossaire musicologique : Section “triolet” dans *Glossaire*, Section “n-olet” dans *Glossaire*, Section “polymétrie” dans *Glossaire*.

Manuel d’initiation : Section “Méthodes de retouche” dans *Manuel d’initiation*.

Manuel de notation : Section 2.1.3 [Changement d’échelle des durées], page 62, Section 36.1 [Direction et positionnement], page 801, Section 2.3.1 [Mesures], page 80, Section 35.6 [Les commandes `\tweak` et `\single`], page 791, Section 2.3.6 [Notation polymétrique], page 96, Section 36.7 [Visibilité des objets], page 812.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “TupletBracket” dans *Référence des propriétés internes*, Section “TupletNumber” dans *Référence des propriétés internes*, Section “TimeScaledMusic” dans *Référence des propriétés internes*.

2.1.3 Changement d’échelle des durées

La durée des notes, silences ou accords peut se modifier en lui adjoignant une fraction *N/D*, donnant « ∗*N/D* » – ou « ∗*N* » si *D=1*. Ce facteur peut aussi s’exprimer à l’aide d’une expression Scheme évaluant à un nombre ou un temps musical comme `*#(ly:music-length` *musique*`)`. Cette solution est pratique pour échelonner à une durée de ‘`1`’ et laisser une note ou un silence multimesure s’étirer sur une longueur dérivée d’une variable musicale.

L’ajout de ce facteur ne modifiera en rien l’apparence des notes ou silences produits, mais affectera le positionnement de l’objet dans la mesure ainsi que le rendu MIDI. Cette fraction peut elle-même être multipliée, ce qui donne quelque chose du style `*M*N/D`. Ce facteur d’échelonnement est partie intégrante de la durée : en l’absence de durée explicite à la note suivante, cette durée échelonnée est considérée comme valeur par défaut.

Dans l’exemple suivant, les trois premières notes prennent exactement deux temps, mais aucun triolet n’est imprimé.
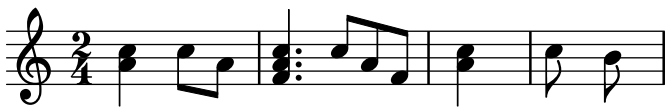
```
\relative {
  \time 2/4
  % Alter durations to triplets
  a'4*2/3 gis a
  % Normal durations
  a4 a
  % Double the duration of chord
  <a d>4*2
  % Duration of quarter, appears like sixteenth
  b16*4 c4
}
```



La durée d'un silence invisible ou saut de notes (*skip*) peut elle aussi être affectée d'un multiplicateur. Cette technique permet tout simplement de sauter plusieurs mesures, comme par exemple un `s1*23`.

Il est tout à fait possible d'échelonner des fragments musicaux plus ou moins longs à l'aide d'une simple fraction, comme si chaque note, accord ou silence était affecté de ce même quotient. L'apparence de cette musique ne sera en rien modifiée ; seule la durée des notes est multipliée en interne par le facteur d'échelle donné – généralement *numérateur*/*dénominateur*. Voici un exemple illustrant la manière de comprimer ou étirer de la musique :

```
\relative {
  \time 2/4
  % Durée normale
  <c'' a>4 c8 a
  % Musique échelonnée à 2/3
  \scaleDurations 2/3 {
    <c a f>4. c8 a f
  }
  % Musique échelonnée par 2
  \scaleDurations 2 {
    <c' a>4 c8 b
  }
}
```



Cette technique est tout à fait appropriée à la notation polymétrique – voir Section 2.3.6 [Notation polymétrique], page 96.

Voir aussi

Manuel de notation : Section 2.1.2 [N-olets], page 56, Section 2.3.6 [Notation polymétrique], page 96, Section 2.2.2 [Silences invisibles], page 71.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Problèmes connus et avertissements

Le calcul de la position au sein d'une mesure doit prendre en considération tous les facteurs d'échelonnement appliqués aux notes de cette mesure ainsi que tous les reliquats des mesures précédentes. Ce calcul utilise des nombres rationnels. Dès lors qu'un calcul rencontrera un numérateur ou dénominateur intermédiaire d'une valeur supérieure à 2^30, LilyPond s'arrêtera à ce point précis sans pour autant signaler d'erreur.

2.1.4 Liaisons de prolongation
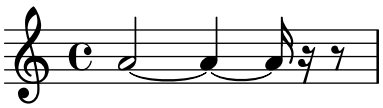
Une liaison de tenue (ou de prolongation) relie deux notes adjacentes de même hauteur. Dans les faits, elle prolonge la durée d'une note.

Une liaison de tenue sans note d'arrivée est une articulation de *laissez vibrer* – voir [Laissez vibrer], page 65, pour la commande `\laissezVibrer`. Une liaison de tenue venant de nulle part, comme celles qui peuvent apparaître en début de section alternative, s'indique par la commande `\repeatTie` – voir [Tenue et reprise], page 65.

> **Note :** Une liaison de tenue ne doit pas être confondue avec une liaison d'**articulation** ou de **phrasé**. Une liaison de tenue est un moyen parmi d'autres pour prolonger la durée d'une note, tout comme les points.
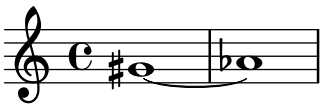
Une liaison de tenue s'indique au moyen d'un tilde (~) qui vient s'adjoindre à la première note de chacune des paires de notes à lier. Ceci indique que la note en question sera liée à la suivante, qui doit être de la même hauteur. Il est à noter que les liaisons de tenue savent interpréter la « dernière hauteur explicite » des durées isolées :

```
{ a'2~ 4~ 16 r r8 }
```
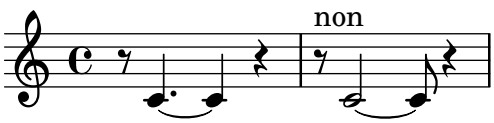


LilyPond sait aussi gérer les liaisons de tenue entre harmoniques :

```
{ gis'1~ | as'1 }
```



Les liaisons de tenue sont utilisées soit lorsque la note dépasse de la mesure, soit quand les points ne suffisent pas à donner la bonne durée. Lorsque l'on utilise ces liaisons, les valeurs rythmiques les plus longues doivent s'aligner sur les subdivisions de la mesure, comme ici :

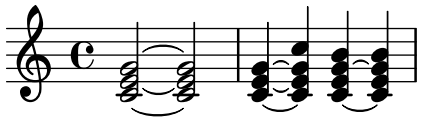
```
\relative {
  r8 c'4.~ 4 r4 |
  r8^"non" c2~ 8 r4
}
```



Lorsque l'on doit lier de nombreuses notes sur plusieurs mesures, il devient plus facile d'avoir recours à la division automatique des notes – voir Section 2.3.7 [Découpage automatique des notes], page 100. Ce procédé divise automatiquement les notes trop longues, et les lie par-delà les barres de mesure.

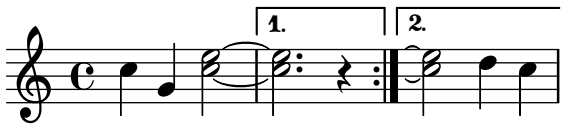
Quand une liaison de tenue se trouve entre deux accords, toutes les notes de même hauteur entre ces deux accords sont reliées. S'il n'y en a aucune, aucune liaison n'est créée. Il est également possible de lier partiellement deux accords, en mettant les liaisons à l'intérieur des accords.

```
\relative c' {
  <c e g>2~ 2 |
  <c e g>4~ <c e g c>
    <c~ e g~ b> <c e g b> |
}
```
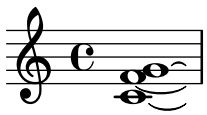


Lorsqu'une mesure d'alternative après une reprise commence sur une note liée, la liaison doit être répétée, comme ici :

```
\relative {
  \repeat volta 2 { c'' g <c e>2~ }
  \alternative {
    % La note tenue est liée normalement
    \volta 1 { <c e>2. r4 }
    % la note qui suit est pourvue d'une liaison de reprise
    \volta 2 { <c e>2\repeatTie d4 c }
  }
}
```



Les liaisons « Laissez vibrer » (*L.v.*) sont utilisées pour le piano, la harpe et certains instruments de percussion. Elles indiquent à l'instrumentiste de laisser sonner la note ou l'accord au lieu de l'étouffer. Elles s'indiquent de la manière suivante :

```
<c' f' g'>1\laissezVibrer
```
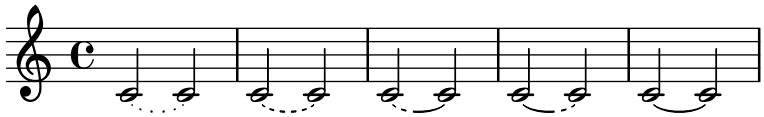


Le positionnement vertical des liaisons de prolongation peut être affiné – voir à ce sujet les « commandes prédéfinies » et, pour de plus amples détails, Section 36.1 [Direction et positionnement], page 801.

Les liaisons de prolongation peuvent s'imprimer sous la forme de ligne continue, discontinue ou pointillée.
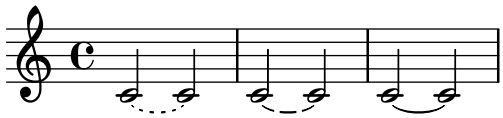
```
\relative c' {
  \tieDotted
  c2~ 2
  \tieDashed
  c2~ 2
  \tieHalfDashed
  c2~ 2
  \tieHalfSolid
  c2~ 2
```

```
  \tieSolid
  c2~ 2
}
```



Il est même possible d'en personnaliser l'allure :
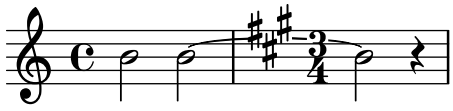
```
\relative c' {
  \tieDashPattern 0.3 0.75
  c2~ 2
  \tieDashPattern 0.7 1.5
  c2~ 2
  \tieSolid
  c2~ 2
}
```



Que ce soit pour une tenue ou un phrasé, le motif d'une ligne discontinue formant une liaison se définit de la même manière. Pour de plus amples détails à ce sujet, reportez vous au chapitre Section 3.2.1 [Liaisons d'articulation], page 174.

Dans le cas où une liaison est recouverte par d'autres éléments de la portée, une adaptation des propriétés *whiteout* et *layer* permet d'obtenir une meilleure lisibilité.

```
\relative {
  \override Tie.layer = -2
  \override Staff.TimeSignature.layer = -1
  \override Staff.KeySignature.layer = -1
  \override Staff.TimeSignature.whiteout = ##t
  \override Staff.KeySignature.whiteout = ##t
  b'2 b~
  \time 3/4
  \key a \major
  b r4
}
```



Commandes prédéfinies
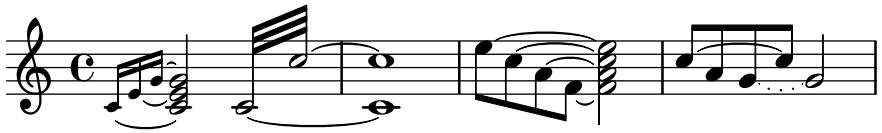
\tieUp, \tieDown, \tieNeutral, \tieDotted, \tieDashed, \tieDashPattern, \tieHalfDashed, \tieHalfSolid, \tieSolid.

Morceaux choisis

Liaison de tenue et arpège

Les liaisons de tenue servent parfois à rendre un accord arpégé. Dans ce cas, les notes liées ne sont pas toutes consécutives. Il faut alors assigner à la propriété `tieWaitForNote` la valeur `#t` (*true* pour « vrai »). Cette même méthode peut servir, par exemple, à lier un trémolo à un accord.

```
\relative c' {
  \set tieWaitForNote = ##t
  \grace { c16[ ~ e ~ g] ~ } <c, e g>2
  \repeat tremolo 8 { c32 ~ c' ~ } <c c,>1
  e8 ~ c ~ a ~ f ~ <e' c a f>2
  \tieUp
  c8 ~ a
  \tieDown
  \tieDotted
  g8 ~ c g2
}
```



Dessin à main levée de liaisons de tenue

Il est possible de graver manuellement les liaisons de tenue, en modifiant la propriété `staff-position` (un décalage) de l'objet `Tie`. Lorsqu'il y en a plusieurs, elles peuvent être ajustées manuellement à l'aide de la propriété `tie-configuration` de l'objet `TieColumn`, constituée d'une liste de paires. Pour chaque paire, le premier nombre indique la distance à la portée, en espaces de portée, et le second la direction (1 pour haut, −1 pour bas).

Notez bien que LilyPond fait la distinction, au niveau du décalage, entre valeur exacte et valeur inexacte. Dans le cas d'une valeur exacte – autrement dit un entier ou une fraction comme `(/ 4 5)` – celle-ci servira de position verticale brute, ensuite afinée par LilyPond de sorte à éviter les lignes de la portée. Dans le cas d'une valeur inexacte, tel un nombre à virgule flottante, c'est elle qui servira à positionner verticalement, sans ajustement.

```
\relative c' {
  <>^"default"
  g'1 ^~ g

  <>^"0"
  \once \override Tie.staff-position = 0
  g1 ^~ g

  <>^"0.0"
  \once \override Tie.staff-position = 0.0
  g1 ^~ g

  <>^"reset"
  \revert Tie.staff-position
  g1 ^~ g
}
```

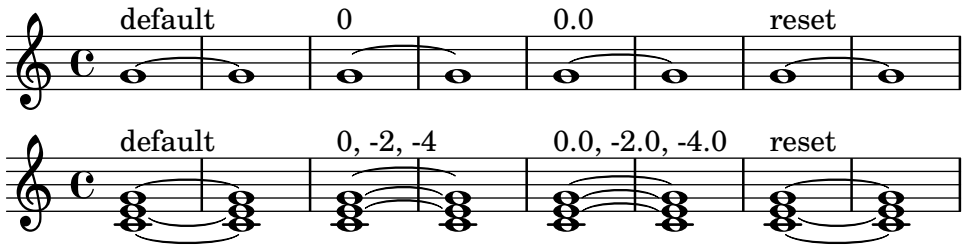
```
\relative c' {
  \override TextScript.outside-staff-priority = ##f
  \override TextScript.padding = 0

  <>^"default"
  <c e g>1~ <c e g>

  <>^"0, -2, -4"
  \override TieColumn.tie-configuration =
    #'((0 . 1) (-2 . 1) (-4 . 1))
  <c e g>1~ <c e g>

  <>^"0.0, -2.0, -4.0"
  \override TieColumn.tie-configuration =
    #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
  <c e g>1~ <c e g>

  <>^"reset"
  \override TieColumn.tie-configuration = ##f
  <c e g>1~ <c e g>
}
```



Voir aussi

Glossaire musicologique : Section “liaison de tenue” dans *Glossaire*, Section “laissez vibrer” dans *Glossaire*.

Manuel de notation : Section 2.3.7 [Découpage automatique des notes], page 100, Section 3.2.1 [Liaisons d’articulation], page 174.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

Référence des propriétés internes : Section “LaissezVibrerTie” dans *Référence des propriétés internes*, Section “LaissezVibrerTieColumn” dans *Référence des propriétés internes*, Section “TieColumn” dans *Référence des propriétés internes*, Section “Tie” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un changement de portée, lorsqu’une liaison de tenue est active, ne peut produire une liaison oblique.

Un changement de clef ou d’octave pendant une liaison de tenue produit un résultat indéfini. Dans ces cas là, il est préférable d’utiliser un *legato*.
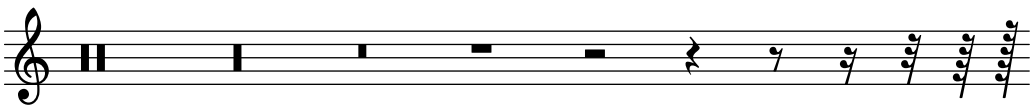
2.2 Écriture des silences

On saisit les silences dans une expression musicale tout comme les notes.

2.2.1 Silences

Les silences sont écrits comme des notes avec le nom de note `r` – premier caractère du mot *rest*. Les durées supérieures à la pause s'indiquent à l'aide de commandes prédéfinies :
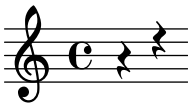
```
\new Staff {
  % Ces deux lignes permettent uniquement une meilleure présentation
  \time 16/1
  \omit Staff.TimeSignature
  % Imprime un silence de maxime, équivalant à quatre brèves
  r\maxima
  % imprime un silence de longue, équivalant à deux brèves
  r\longa
  % Imprime un silence de brève
  r\breve
  r1 r2 r4 r8 r16 r32 r64 r128
}
```



Les pauses d'une mesure complète, qui sont placées au centre de la mesure, doivent être entrées comme des mesures de silence. Elles peuvent être utilisées pour une seule mesure comme pour plusieurs, et leur utilisation est expliquée à la rubrique Section 2.2.3 [Silences valant une mesure], page 73.

Pour spécifier explicitement la position verticale d'un silence, écrivez une note suivie de `\rest`. Un silence de même durée sera placé à la position où serait imprimée la note. Cela rend plus facile la mise en place de musique polyphonique, puisque le formateur automatique de collision des silences laissera ces silences tranquilles.

```
\relative { a'4\rest d4\rest }
```



Morceaux choisis

Styles de silences

Les silences peuvent être gravés selon différents styles.

```
restsA = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 s32
  s64 s128 s256 s512 s1024 s1024
}
restsB = {
  r\maxima r\longa r\breve r1 r2 r4 r8 r16 r32
  r64 r128 r256 r512 r1024 s1024
}

\new Staff \relative c {
  \omit Score.TimeSignature
  \cadenzaOn
```

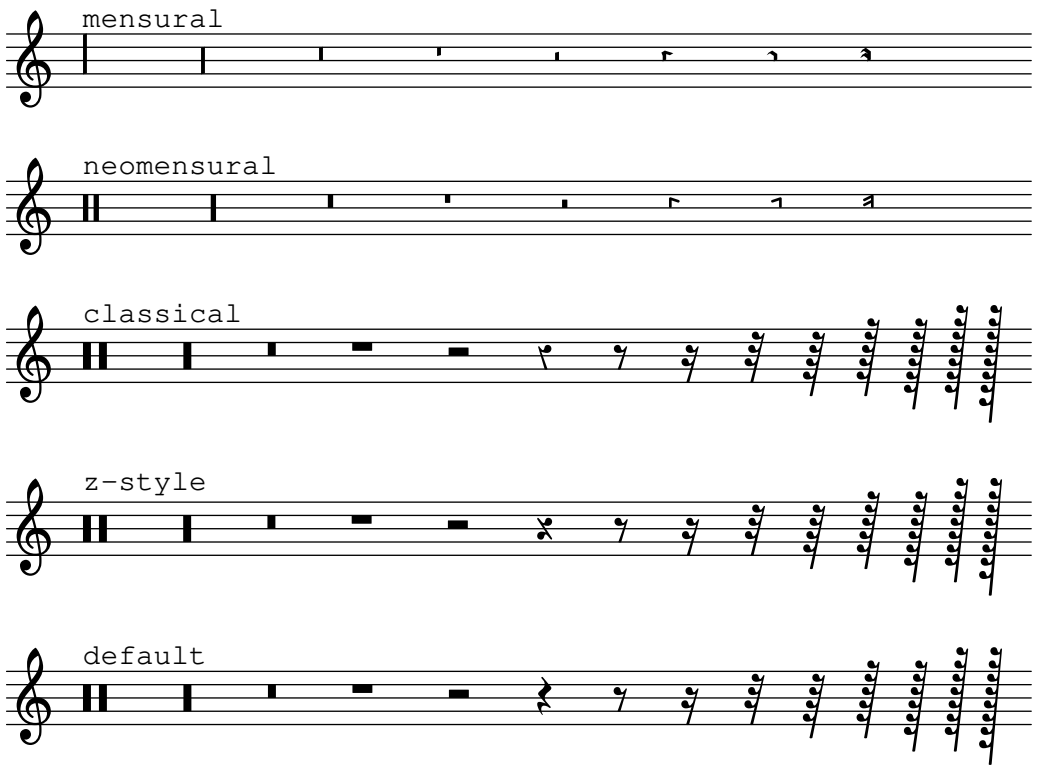
```
  \override Staff.Rest.style = #'mensural
  <>^\markup \typewriter { mensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'neomensural
  <>^\markup \typewriter { neomensural } \restsA \bar "" \break

  \override Staff.Rest.style = #'classical
  <>^\markup \typewriter { classical } \restsB \bar "" \break

  \override Staff.Rest.style = #'z
  <>^\markup \typewriter { z-style } \restsB \bar "" \break

  \override Staff.Rest.style = #'default
  <>^\markup \typewriter { default } \restsB \bar "" \break
}
```



Voir aussi

Glossaire musicologique : Section “breve” dans *Glossaire*, Section “longa” dans *Glossaire*, Section “maxima” dans *Glossaire*.

Manuel de notation : Section 2.2.3 [Silences valant une mesure], page 73.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Rest” dans *Référence des propriétés internes*.
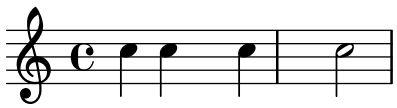
Problèmes connus et avertissements

Il n’existe pas à proprement parler de limite (inférieure ou supérieure) en terme de durée d’un silence. Cependant, le nombre de glyphes disponibles ne couvre que les silences allant du 1 024e de pause (256e de soupir) à la maxime (équivalant à huit pauses).

2.2.2 Silences invisibles

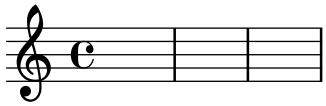
Il existe deux formes de silence invisible – que l'on pourrait appeler « saut » : le *silence d'espacement* qui se note `s`, et la commande `\skip`. Le silence d'espacement est une note qui ne produit pas de résultat. Comme pour toute autre note ou silence, la durée qui lui est affectée définit la durée par défaut des notes qui le suivent.

```
\relative c'' {
  c4 c s c |
  s2 c |
}
```
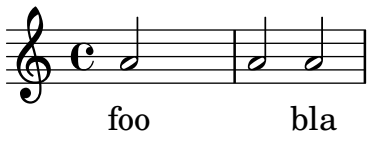


Tout comme les autres notes ou silences, un silence invisible génère implicitement les contextes `Staff` et `Voice` s'il n'y en avait pas.

```
{ s1 s s }
```
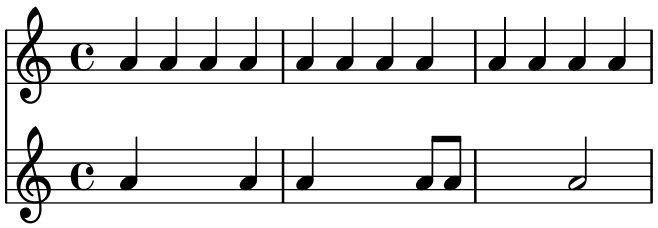


La syntaxe `s` est seulement disponible pour les modes d'entrée de notes et d'accords. Dans les autres situations, pour l'entrée de paroles par exemple, vous devrez utiliser la commande `\skip` pour sauter un moment. La commande `\skip` prend en argument soit une durée explicite, soit un fragment musical, et saute la durée de cet argument. Cette durée ne sera pas prise en considération dès lors que les paroles suivent le rythme des notes de la mélodie à laquelle vous les aurez associées à l'aide des commandes `\addlyrics` ou `\lyricsto`.

```
<<
  {
    a'2 \skip2 a'2 a'2
  }
  \new Lyrics {
    \lyricmode {
      foo2 \skip 1 bla2
    }
  }
>>
```



Lorsque l'argument de `\skip` est de la musique, la durée par défaut de la note qui suit est celle de la dernière note du fragment « sauté ». Toutefois, et pour préserver la compatibilité ascendante avec l'implémentation originelle de `\skip`, la durée explicitée en argument n'affectera en rien la durée des notes qui suivent.

```
<<
  {
    \*12 a'4
  }
  {
    a'4 \skip 2 a' |
    a'4 \skip { a'8 a' a' a' } a' a' |
    s2 a'
  }
>>
```
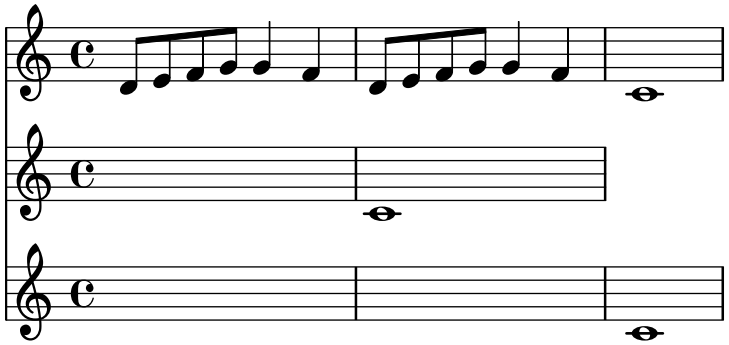


La commande \skip préserve les effets de la commande unfoldRepeats qui la contiendrait, à l'inverse de la fonction Scheme skip-of-length.

```
MyCadenza = \fixed c' {
  \repeat volta 2 {
    d8 e f g g4 f4
  }
}

music = <<
  \new Staff {
    \MyCadenza
    c'1
  }
  \new Staff {
    #(skip-of-length MyCadenza)
    c'1
  }
  \new Staff {
    \skip \MyCadenza
    c'1
  }
>>

\unfoldRepeats \music
```



La commande \skip ne fait que sauter du temps musical ; elle ne produit rien du tout, pas même un symbole transparent.

```
% Saisie valide qui ne donne rien
{ \skip 1 \skip1 \skip 1 }
```

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 7.1.4 [Dictée à trous], page 297, Section 36.7 [Visibilité des objets], page 812.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SkipMusic" dans *Référence des propriétés internes*.

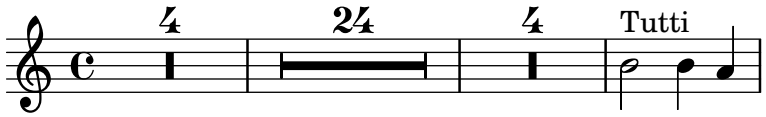
Problèmes connus et avertissements

Dans la mesure où l'argument fourni à la commande \skip – durée ou fragment musical – affecte différemment la durée de la musique qui suit, il est de bon ton de mentionner explicitement une durée à ce qui la suit.

2.2.3 Silences valant une mesure

Un silence valant une ou plusieurs mesures entières s'entre avec un R majuscule. Sa durée s'indique comme pour n'importe quelle note, y compris un éventuel multiplicateur comme indiqué dans Section 2.1.3 [Changement d'échelle des durées], page 62.

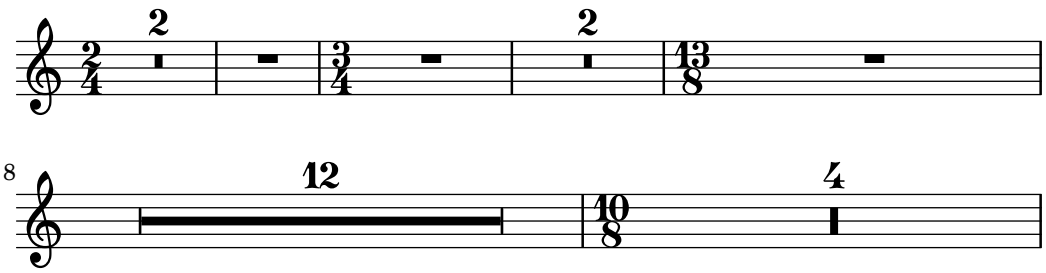
```
% Mesures de silence contractées en une seule
\compressMMRests {
  R1*4
  R1*24
  R1*4
  b'2^"Tutti" b'4 a'4
}
```



Cet exemple illustre aussi la manière de condenser une succession de mesures vides, comme indiqué dans Section 6.3.4 [Compression de mesures vides], page 282.

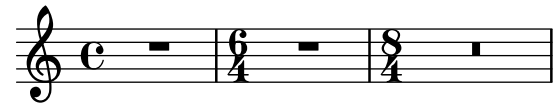
Ceci ne peut être utile que pour une mesure complètement vide. Sa durée doit donc correspondre à la longueur de la mesure telle que définie par la métrique. C'est la raison pour laquelle on utilisera aussi des points d'augmentation ou des fractions :

```
\compressMMRests {
  \time 2/4
  R1 | R2 |
  \time 3/4
  R2. | R2.*2 |
  \time 13/8
  R1*13/8 | R1*13/8*12 |
  \time 10/8
  R4*5*4 |
}
```
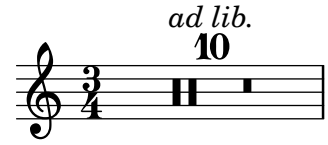


Un R qui s'étend sur une seule mesure s'imprime tantôt comme une pause, tantôt comme une brève – ou « bâton de pause » – qui sera centrée sur la mesure quelle qu'en soit la métrique :

```
\time 4/4
R1 |
\time 6/4
R1*3/2 |
\time 8/4
R1*2 |
```
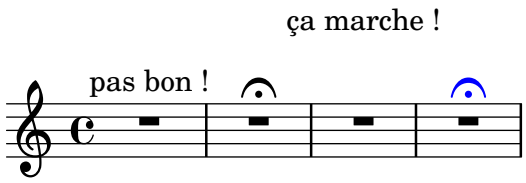


Vous pouvez aussi ajouter du texte à un silence multimesure en utilisant la syntaxe *note*-markup (cf. Section 8.2 [Mise en forme du texte], page 329).

```
\compressMMRests {
  \time 3/4
  R2.*10^\markup { \italic "ad lib." }
}
```



Note : C'est MultiMeasureRestText qui créera le texte, et MultiMeasureRestScript pour les articulations, non TextScript ni Script. Les commandes de dérogation ou de redéfinition doivent s'adresser à l'objet concerné, comme vous pouvez le constater dans l'exemple suivant.

```
% Ceci échouera : il y a erreur quant à l'objet spécifié
\override TextScript.padding = 5
\override Script.color = #blue
R1^"pas bon !"
R1\fermata
% Formulation correcte, qui fonctionnera
\override MultiMeasureRestText.padding = 5
\override MultiMeasureRestScript.color = #blue
R1^"ça marche !"
R1\fermata
```



Un silence multimesure placé directement après une commande `\partial` risque fort de perturber le vérificateur de limites et numéros de mesure.

Commandes prédéfinies

`\textLengthOn`, `\textLengthOff`, `\compressMMRests`.

Morceaux choisis

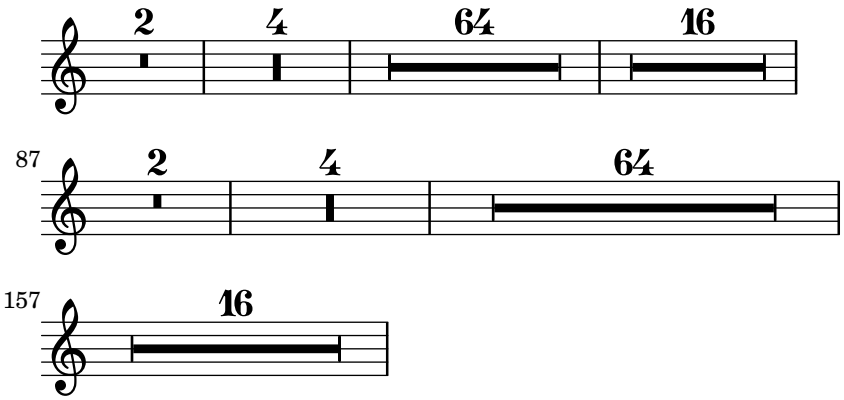
Contrôle de la taille d'un silence multimesure

Les silences multimesure ont une largeur relative à leur durée totale, contrôlée par la propriété `space-increment` de l'objet graphique `MultiMeasureRest`. Sa valeur par défaut est fixée à 2.

```
\relative c' {
  \omit Staff.TimeSignature
  \compressEmptyMeasures

  R1*2 R1*4 R1*64 R1*16 \break
  \override MultiMeasureRest.space-increment = 4
  R1*2 R1*4 R1*64 R1*16
}

\layout {
  ragged-right = ##t
}
```



Positionnement des silences multimesures

Si l'on peut positionner verticalement un silence simple en le rattachant à une note, il n'en va pas de même pour un silence multimesure. Néanmoins, et uniquement dans le cadre de musique polyphonique, les silences multimesures sont positionnés différemment selon qu'ils appartiennent à une voix au numéro pair ou impair.

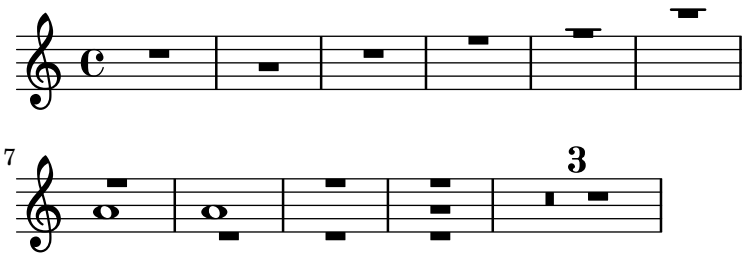
Le positionnement des silences multimesures peut se contrôler comme ici indiqué.

```
\relative c'' {
  % Multi-measure rests by default are set under the fourth line.
  R1
  % They can be moved using an override or tweak.
  \tweak staff-position -2 R1
  \tweak staff-position 0 R1
  \tweak staff-position 2 R1
  \override MultiMeasureRest.staff-position = 3 R1
  \override MultiMeasureRest.staff-position = 6 R1
  \revert MultiMeasureRest.staff-position
  \break

  % Odd-numbered voices are under the top line.
  << { R1 } \\ { a1 } >>
  % Even-numbered voices are under the bottom line.
  << { a1 } \\ { R1 } >>
  % Multi-measure rests in both voices remain separate.
  << { R1 } \\ { R1 } >>

  % Separating multi-measure rests in more than two voices
  % requires an override or tweak.
  << { R1 } \\ { R1 } \\ { \tweak staff-position -2 R1 } >>

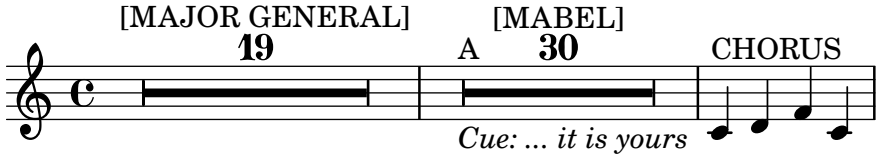
  % Using compressed bars in multiple voices requires another override
  % in all voices to avoid multiple instances being printed.
  \compressMMRests
  <<
    \revert MultiMeasureRest.direction
    { R1*3 } \\
    \revert MultiMeasureRest.direction
    { R1*3 }
  >>
}
```



Ajout de texte à un silence multimesure

Lorsque du texte est attaché à un silence multimesure, il sera centré dans la mesure, au-dessus ou en dessous de la portée. Afin d'étirer la mesure dans le cas où ce texte est relativement long, il suffit d'insérer un accord vide auquel on attache le texte en question, avant le silence multimesure. Le texte attaché à un silence invisible sera aligné sur la gauche de là où serait positionnée la note dans la mesure. Cependant, si la taille de la mesure est déterminée par la longueur du texte, il apparaîtra comme centré.

```
\relative c' {
  \compressMMRests {
    \textLengthOn
    <>^\markup { [MAJOR GENERAL] }
    R1*19
    <>_\markup { \italic { Cue: ... it is yours } }
    <>^\markup { A }
    R1*30^\markup { [MABEL] }
    \textLengthOff
    c4^\markup { CHORUS } d f c
  }
}
```



Voir aussi

Glossaire musicologique : Section “silence multimesures” dans *Glossaire*.

Manuel de notation : Section 2.1.3 [Changement d’échelle des durées], page 62, Section 8.1.2 [Commentaires textuels], page 320, Section 6.3.4 [Compression de mesures vides], page 282, Section 2.1.1 [Durées], page 53, Section 8.2 [Mise en forme du texte], page 329, Chapitre 8 [Texte], page 317.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “MultiMeasureRest” dans *Référence des propriétés internes*, Section “MultiMeasureRestNumber” dans *Référence des propriétés internes*, Section “MultiMeasureRestScript” dans *Référence des propriétés internes*, Section “MultiMeasureRestText” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Vous ne pouvez pas utiliser de doigtés (par ex. `R1*10-4`) pour positionner des nombres au dessus d’un silence multimesure, le numéro de doigt (4) risquant de chevaucher le nombre de mesures à compter (10).

Condenser plusieurs silences en un unique silence multimesure ne peut être automatisé.

Les silences multimesures peuvent générer des collisions avec d’autres silences.

2.2.4 Césures

La commande `\caesura` indique un silence de durée non déterminée. Il s’agit typiquement d’une rupture du flux sonore qui ne raccourcit pas la note qui précède.

```
\fixed c'' { c2. \caesura d4 }
```



En matière d’hymnes et de chants liturgiques, `\caesura` sert plus généralement à diviser les phrases. Pour plus d’information, voir les références à la fin de cette section. Pour les coupures qui amputent la note précédente, voir Section 3.2.3 [Signes de respiration], page 179.
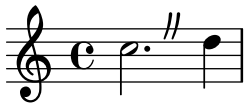
Des articulations peuvent suivre un \caesura afin d'indiquer la durée relative ou la signification de la pause. Seront alors créés des objets CaesuraScript.

```
\fixed c'' { c2. \caesura \fermata d4 }
```



Par défaut, \caesura crée un objet BreathingSign. L'élément breath de la propriété de contexte caesuraType contrôle lequel des signes prédéfinis sera créé par \caesura – voir Section B.14 [Liste des marques de respiration], page 946.
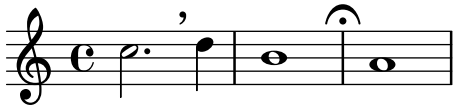
```
\fixed c'' {
  \set Score.caesuraType = #'((breath . curvedcaesura))
  c2. \caesura d4
}
```



Afin de désigner un ou plusieurs objets CaesuraScript devant être créés pour intégrer une césure sans articulation, il faudra définir l'élément scripts de la propriété de contexte caesuraType – des scripts additionnels peuvent toutefois s'attacher en tant qu'articulations. Conjointement à l'élément breath, les scripts listés dans l'élément script s'attachent au BreathingSign ; c'est par contre au barLine qu'ils s'attacheront s'il est présent.

La propriété de contexte caesuraTypeTransform peut consister en une fonction Scheme pour permettre une forme d'adaptation plus ou moins automatisée. Le générateur de fonction at-bar-line-substitute-caesura-type prend en charge les styles pour lesquels la notation diffère lors d'une barre de mesure.

```
\fixed c' {
  \set Score.caesuraType =
    #'((breath . spacer)
       (scripts . (outsidecomma)))
  \set Score.caesuraTypeTransform =
    #(at-bar-line-substitute-caesura-type
      '((scripts . (fermata))))
  c'2. \caesura d'4
  b1 \caesura
  a1
}
```



Commandes prédéfinies

\caesura.

Morceaux choisis

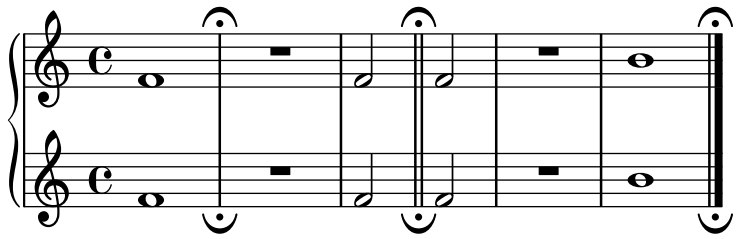
Positionnement de points d'orgue en opposition sur une barre de mesure

Dans l'extrait suivant est construite une commande qui permet d'imprimer un point d'orgue à la fois en surplomb et en dessous d'une barre de mesure. Dans le cas où il n'y aurait normalement pas de barre à cet endroit, sera ajoutée une double barre. Sémantiquement, cette commande constuit une césure plus longue que la normale, ce qui pourrait être considéré comme un usage abusif selon le cas.

```
twoWayFermata = {
  \once \set Staff.caesuraType = #'((underlying-bar-line . "||"))
  \once \set Staff.caesuraTypeTransform = ##f
  \caesura ^\fermata _\fermata
}

music = {
  f'1 \twoWayFermata
  R1
  f'2 \twoWayFermata f'2
  R1
  b'1 \twoWayFermata \fine
}

\new GrandStaff <<
  \new Staff \music
  \new Staff \music
>>
```



Voir aussi

Glossaire musicologique : Section "césure" dans *Glossaire*.

Manuel de notation : Section 9.7.4 [Barre de phrase en musique liturgique], page 421, Section 17.5.5 [Barres de mesure kiéviennes], page 581, Section 17.4.4 [Divisions], page 571, Section 3.2.3 [Signes de respiration], page 179.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "BreathingSign" dans *Référence des propriétés internes*, Section "Caesura_engraver" dans *Référence des propriétés internes*, Section "CaesuraEvent" dans *Référence des propriétés internes*, Section "CaesuraScript" dans *Référence des propriétés internes*, Section "Tunable context properties" dans *Référence des propriétés internes*.

2.3 Gravure du rythme

2.3.1 Mesures

Répartir la musique en mesures numérotées et garder trace du temps au sein des mesures est la tâche du `Timing_translator`. Ce traducteur n'est pas directement responsable de la gravure, mais il définit certaines propriétés de contexte qui guident les graveurs et interprètes.

Le `Timing_translator` réside par défaut dans le contexte `Score`. Un alias, `Timing`, sera ajouté au contexte auquel le `Timing_translator` est rattaché. Déclarer explicitement un contexte `Voice` ou `Staff` assure l'existence de cet alias.

Le `Timing_translator` dispose des propriétés suivantes du contexte `Timing` afin de garder trace du minutage de la partition.

`currentBarNumber`
: Le numéro de la mesure en cours. Un exemple d'utilisation se trouve au chapitre Section 2.5.3 [Numéros de mesure], page 135.
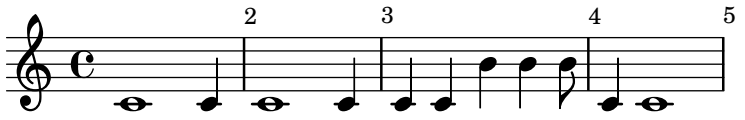
`measureLength`
: La longueur de la mesure, dans la métrique en cours. Pour une mesure à 4/4, elle est de 1, et de 3/4 pour une mesure à 6/8. Il est possible de définir manuellement cette propriété pour créer une mesure dont la longueur sera exceptionnellement différente, mais elle ne devrait en tout cas pas être inférieure ou égale à `measurePosition` puisque l'éventualité d'entamer une nouvelle mesure au bon moment est passée.

`measurePosition`
: Cette propriété interne enregistre le moment où l'on en est dans la mesure en cours. Cette quantité est remise à zéro dès lors qu'on dépasse `measureLength` ; la variable `currentBarNumber` est alors incrémentée. `measurePosition` ne devrait pas être déterminée explicitement ; on peut la modifier par la commande `\partial`.

`timing`
: Lorsqu'on lui assigne la valeur `#t`, les valeurs ci-dessus mentionnées sont mises à jour à chaque pas (note, silence ou saut). Fixée à `#f`, leur valeur restera figée indéfiniment.

Le calage peut être modifié en réglant l'une de ces variables. Dans l'exemple qui suit, nous réglons la métrique à 4/4, tout en fixant `measureLength` à 5/4. Arrivé à 4/8 dans la troisième mesure, la commande `\partial` nous avance pour laisser 5/8, raccourcissant donc cette mesure d'une croche. La barre de mesure suivante tombera donc à 9/8 et non à 5/4.

```
\fixed c' {
  \override Score.BarNumber.break-visibility = #all-visible
  \set Timing.measureLength = #5/4
  c1 c4
  c1 c4
  c4 c \partial 8*5 b4 b b8
  c4 c1
}
```



Voir aussi

Manuel de notation : Section 2.3.5 [Musique sans métrique], page 94, Section 2.5.3 [Numéros de mesure], page 135.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “Timing_translator” dans *Référence des propriétés internes*, Section “Score” dans *Référence des propriétés internes*.
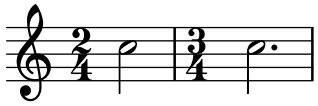
2.3.2 Métrique

Le chiffre de mesure, ou métrique, indique le mètre d'une pièce : une alternance régulière de temps forts et de temps faibles.

Une métrique simple s'obtient en donnant une fraction en argument à la commande `\time`.

```
\time 2/4
c''2
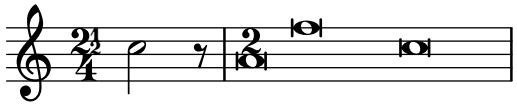
\time 3/4
c''2.
```



Lorsque la métrique est fractionnelle ou que son dénominateur dure plus d'une ronde, il faut recourir à une syntaxe Scheme.
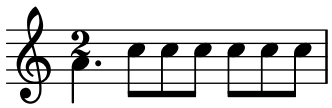
```
\time #'(5/2 . 4)
c''2 r8

\override Timing.TimeSignature.denominator-style = #'note
\time #'(2 . 1/2)
f''\breve c''
```



D'autres dénominateurs inusuels peuvent s'utiliser pour ajouter un point d'augmentation à la note présentée ; toutefois, et afin de bénéficier des fonctionnalités présentées de part et d'autre, l'utilisation d'une fraction conventionnelle pour `\time` associée à une dérogation à `TimeSignature.time-signature` est fortement recommandée en pareil cas.

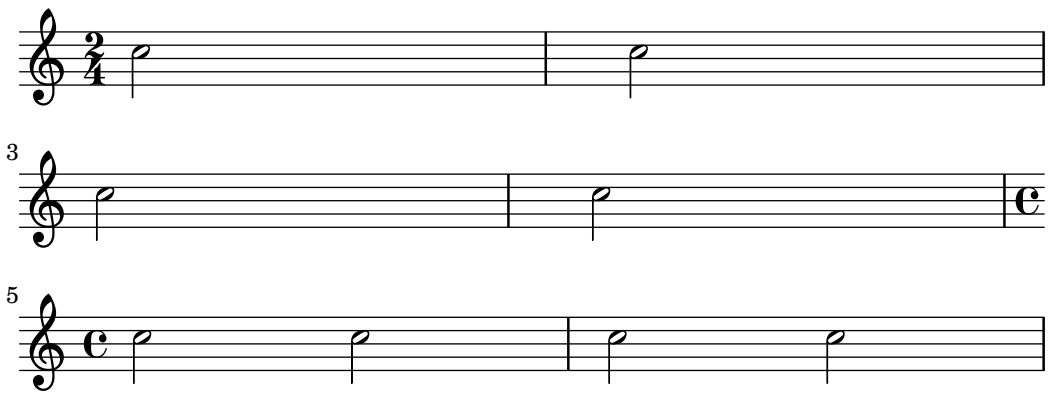
```
\override Timing.TimeSignature.denominator-style = #'note
\once \override Timing.TimeSignature.time-signature = #'(2 . 8/3)
\time 6/8
c''8 8 8 8 8 8
```



Les changements de métrique en cours de mesure sont abordés dans Section 2.3.4 [Levées], page 93.
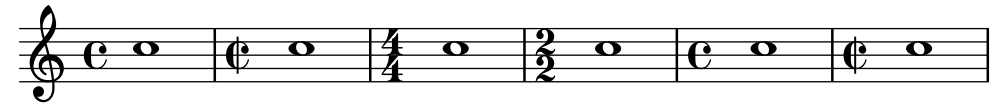
La métrique est imprimée en début de morceau, et à chaque fois qu'elle est modifiée. Si cette modification intervient au niveau d'un saut de ligne, une métrique « de précaution » sera imprimée avant de passer à la ligne suivante. Ce comportement par défaut peut être modifié, comme indiqué au chapitre Section 36.7 [Visibilité des objets], page 812.

```
\relative c'' {
  \time 2/4
  c2 c
  \break
  c c
  \break
  \time 4/4
  c c c c
}
```



Le symbole de métrique utilisé pour les mesures à 2/2 et 4/4 peut être changé pour un style numérique :

```
\relative c'' {
  % Default style
  \time 4/4 c1
  \time 2/2 c1
  % Change to numeric style
  \numericTimeSignature
  \time 4/4 c1
  \time 2/2 c1
  % Revert to default style
  \defaultTimeSignature
  \time 4/4 c1
  \time 2/2 c1
}
```
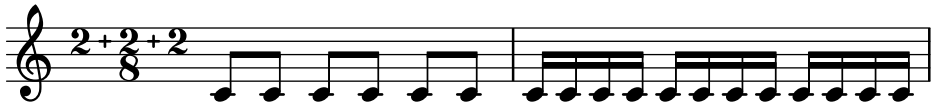


Les métriques anciennes font l'objet d'un Section 17.3.3 [Métriques anciennes], page 558.

Métriques complexes
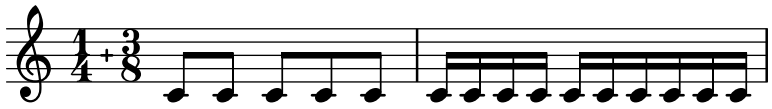
Une métrique composite se crée à l'aide de la fonction `\time`, la métrique étant représentée par une expression Scheme. Les fractions sont représentées par des paires (*numérateur* . *dénominateur*) et où le « dénominateur » est toujours un nombre. Le « numérateur » est un nombre ou une liste de deux ou plusieurs nombres. Une liste représente une concaténation.

```
\fixed c' {
  \time #'((2 2 2) . 8)
  \*6 c8
  \*12 c16
}
```
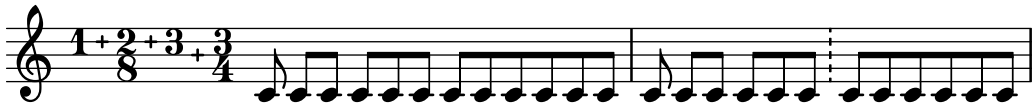


Une métrique constituée de l'alternance stricte de plusieurs fractions s'obtient en regroupant ces fractions dans une liste. Bien entendu, les ligatures automatiques s'ajusteront aux différentes valeurs. Des barres de subdivision de mesures peuvent s'ajouter pour permettre une meilleure lecture des rythmes complexes.

```
\fixed c' {
  \time #'((1 . 4) (3 . 8))
  \*5 c8
  \*10 c16
}
```
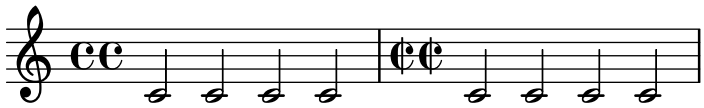


```
\fixed c' {
  \time #'(((1 2 3) . 8) (3 . 4))
  \*12 c8
  \submeasureBarsOn
  \*12 c8
}
```



Par défaut, la stricte alternance de métriques basées sur 4/4 et 2/2 se présente selon la tradition par une suite de « C » ou de « C barré ». Il n'est pas nécessaire de recourir à cette interprétation des métriques pour le mètre sous-jacent : il est aussi possible d'apporter une dérogation à `TimeSignature.time-signature` pour afficher ces symboles alors qu'est défini un autre mètre à l'aide de `\time`.

```
\fixed c' {
  \time #'((4 . 4) (4 . 4))
  \*4 c2
  \once \override Timing.TimeSignature.time-signature
    = #'((2 . 2) (2 . 2))
  \time 4/2
  \*4 c2
}
```
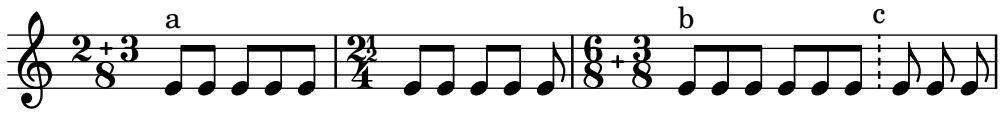


Propriétés de la métrique

En plus de déterminer la métrique qui sera imprimée, la commande `\time` réglera aussi les valeurs par défaut des propriétés de contexte `beatBase`, `beatStructure`, `submeasureStructure` et `beamExceptions` correspondant à la métrique.
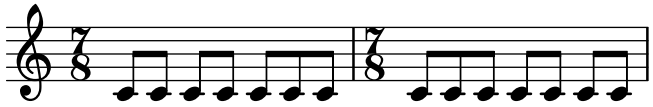
La valeur par défaut de ces propriétés est calculée à partir de la métrique en cours, avec quelques exceptions inscrites dans le fichier `scm/time-signature-settings.scm`. Sans compter les exceptions,

- `beatBase` est de un sur le dénominateur de la métrique. Par exemple, pour une métrique à 3/2, `beatBase` est à 1/2.
- `beatStructure` divise la mesure
  - selon un numérateur structuré (a), ou
  - en temps de 3 × `beatBase` lorsque le numérateur est 6, 9, 12... (b), ou
  - en temps de 1 × `beatBase`.
- `submeasureStructure` divise la mesure avec un alternance stricte de métriques (c).



La valeur par défaut de `beatStructure` peut se voir aménagée dès la commande `\time` à l'aide d'un premier argument :
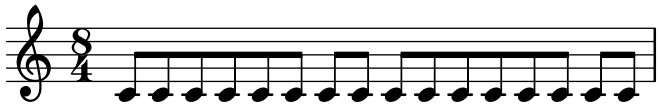
```
\score {
  \new Staff {
    \relative {
      \time 2,2,3 7/8
      \*7 c'8 |
      \time 3,2,2 7/8
      \*7 c8 |
    }
  }
}
```



Les valeurs par défaut de toutes ces variables associées à la métrique peuvent se définir en même temps. Les valeurs adaptées ne seront effectives qu'à partir du moment où interviendra une commande `\time` de la valeur de métrique correspondante :

```
\score {
  \new Staff {
    \relative c' {
      \overrideTimeSignatureSettings
        8/4         % timeSignature
        #1/4        % beatBase
        3,1         % beatStructure
        #'((2 2) (3 1)) % beatStructure, submeasureStructure
        #'()        % beamExceptions
      \time 8/4
      \*16 c8 |
```
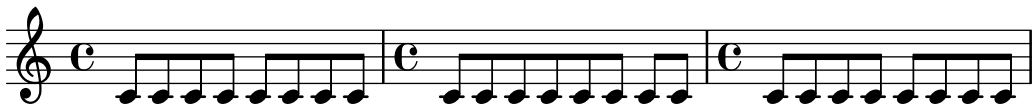
```
    }
  }
}
```



`\overrideTimeSignatureSettings` prend quatre arguments :

1. *`timeSignature`*, une fraction indiquant la métrique pour laquelle ces valeurs doivent s'appliquer,
2. *`beatBase`*, durée musicale correspondant à une unité de `beatStructure`,
3. *`structure`*, une liste Scheme décrivant la structure de cette pulsation, en unité de `beatBase` – autrement dit `beatStructure` – optionnellement couplée avec des listes indiquant les subdivisions de la mesure – c'est à dire `submeasureStructure`.
4. *`beamExceptions`*, une liste associative des règles de ligature pour cette métrique, en dehors de celles basées sur le temps comme indiqué à la rubrique Section 2.4.2 [Définition des règles de ligature automatique], page 107.

Vous pouvez revenir à tout moment aux réglages prédéterminés d'une métrique :

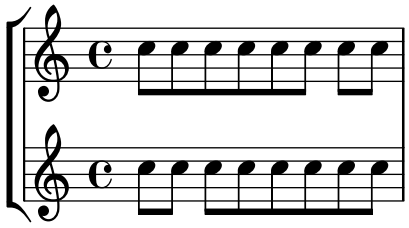
```
\score {
  \relative {
    \*8 c'8 |
    \overrideTimeSignatureSettings
      4/4        % timeSignature
      #1/4       % beatBase
      3,1        % beatStructure
      #'()       % beamExceptions
    \time 4/4
    \*8 c8 |
    \revertTimeSignatureSettings 4/4
    \time 4/4
    \*8 c8  |
  }
}
```



Le fait d'autoriser une présentation polymétrique permet d'obtenir des réglages particuliers pour les différentes portées d'un regroupement – voir Section 2.3.6 [Notation polymétrique], page 96.

```
\score {
  \new StaffGroup <<
    \new Staff {
      \overrideTimeSignatureSettings
        4/4        % timeSignature
        #1/4       % beatBase
        3,1        % beatStructure
        #'()       % beamExceptions
      \time 4/4
```

```
        \*8 c''8
      }
      \new Staff {
        \overrideTimeSignatureSettings
          4/4        % timeSignature
          #1/4       % beatBase
          1,3        % beatStructure
          #'()       % beamExceptions
        \time 4/4
        \*8 c''8
      }
    >>
    \layout {
      \enablePerStaffTiming
    }
}
```



Une autre méthode de modification des propriétés liées à la métrique, et qui évite sa réimpression au moment du changement, est indiquée à la rubrique Section 2.4.2 [Définition des règles de ligature automatique], page 107.

Commandes prédéfinies

`\numericTimeSignature`, `\defaultTimeSignature`.
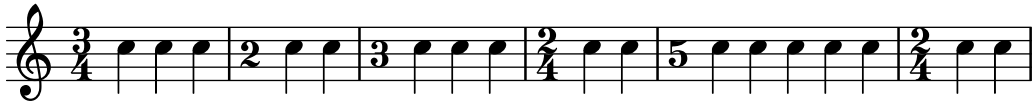
Morceaux choisis

Affichage seulement du numérateur d'une métrique (au lieu d'une fraction)

La métrique est parfois indiquée non pas par une fraction (par ex. 7/4) mais simplement par son numérateur (le chiffre 7 dans ce cas). L'instruction `\override Staff.TimeSignature.style = #'single-number` permet de déroger au style par défaut de manière permanente – un `\revert Staff.TimeSignature.style` annulera ces modifications. Lorsque cette métrique sous la forme d'un seul chiffre ne se présente qu'une seule fois, il suffit de recourir à un `\tweak`.

```
\relative c'' {
  \time 3/4
  c4 c c
  % Change the style permanently
  \override Staff.TimeSignature.style = #'single-number
  \time 2/4
  c4 c
  \time 3/4
  c4 c c
  % Revert to default style:
  \revert Staff.TimeSignature.style
  \time 2/4
```

```
  c4 c
  % single-number style only for the next time signature
  \tweak style #'single-number \time 5/4
  c4 c c c c
  \time 2/4
  c4 c
}
```



Voir aussi

Glossaire musicologique : Section “métrique” dans *Glossaire*.

Manuel de notation : Section 2.4.2 [Définition des règles de ligature automatique], page 107, Section 2.3.1 [Mesures], page 80, Section 17.3.3 [Métriques anciennes], page 558, Section 2.3.6 [Notation polymétrique], page 96.

Fichiers d’initialisation : `scm/time-signature-settings.scm`.
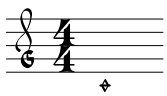
Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “TimeSignature” dans *Référence des propriétés internes*, Section “Timing_translator” dans *Référence des propriétés internes*.

Problèmes connus et avertissements
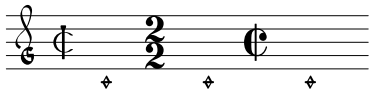
`\numericTimeSignature` et `\defaultTimeSignature` sont sans effet dans le cadre d’un contexte `MensuralStaff`. L’obtention de métriques modernes avec un `MensuralStaff` requiert soit de le stipuler dès la création du contexte :

```
\new MensuralStaff \with { \numericTimeSignature } {
  c'1
}
```



soit de recourir explicitement à un `\override` :
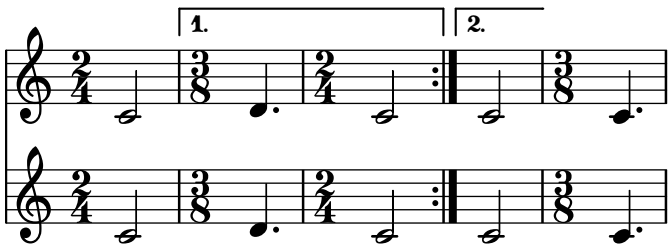
```
\new MensuralStaff {
  \time 2/2
  c'1
  \override MensuralStaff.TimeSignature.style = #'numbered
  \time 2/2
  c'
  \override MensuralStaff.TimeSignature.style = #'default
  \time 2/2
  c'
}
```



Lorsqu’il y a plus d’une portée et qu’un changement de métrique intervient lors d’une première fois, mais pas dans la suivante, il est nécessaire d’aider LilyPond à synchroniser cette situation en ajoutant explicitement une indication de métrique dans la seconde fois tout en la rendant invisible.

```
music = {
  \repeat volta 2 {
    \time 2/4 c'2 |
    \alternative {
      \volta 1 {
        \time 3/8 d'4. |
        \time 2/4 c'2 | }
      \volta 2 {
        \once \omit Staff.TimeSignature
        \time 2/4 c'2 |
      }
    }
  }
  \time 3/8 c'4. |
}
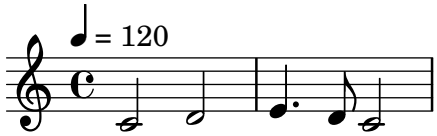
<<
  \new Staff \music
  \new Staff \music
>>
```



2.3.3 Indication métronomique

Une indication métronomique s'insère tout simplement comme ceci :
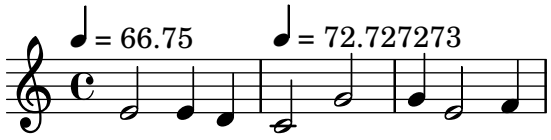
```
\relative {
  \tempo 4 = 120
  c'2 d
  e4. d8 c2
}
```



La vitesse donnée n'est pas forcément un entier, mais elle doit être un nombre exact. Le formateur par défaut arrondit au quart le plus proche ; la propriété de contexte `tempoCountPrecision` permet de le contrôler.
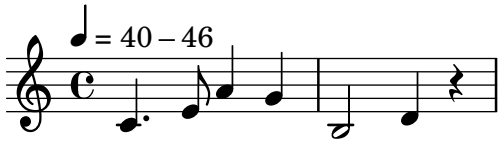
```
tempoI = 100
\fixed c' {
  \tempo 4 = #(* tempoI 2/3)  % 66,66...
  e2 e4 d
  \once \set Timing.tempoCountPrecision = #1/1000000
  \tempo 4 = #(* tempoI 8/11)  % 72.72...
  c2 g2
  g4 e2 f4
```
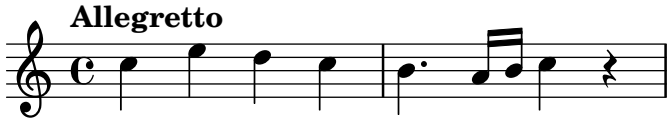
```
}
```



Lorsque le réglage précis du métronome est laissé à l'appréciation de l'exécutant, vous pouvez cependant lui fournir une plage :

```
\relative {
  \tempo 4 = 40 - 46
  c'4. e8 a4 g
  b,2 d4 r
}
```
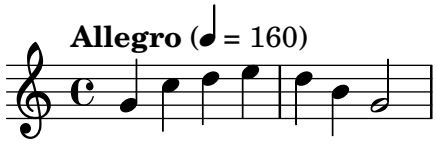


Vous pouvez préférer une indication textuelle :

```
\relative {
  \tempo "Allegretto"
    c''4 e d c
    b4. a16 b c4 r4
}
```
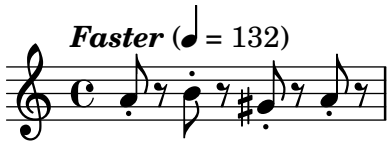


Lorsque vous combinez des indications métronomiques sous forme textuelle et numérique, l'indication numérique sera placée entre parenthèses :

```
\relative {
  \tempo "Allegro" 4 = 160
  g'4 c d e
  d4 b g2
}
```
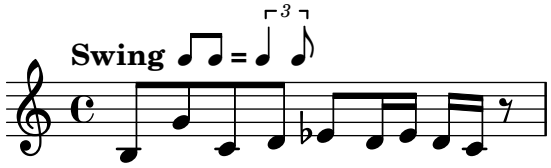


En matière d'indication textuelle, vous pouvez utiliser n'importe quel objet de type *markup*, comme ici :

```
\relative {
  \tempo \markup { \italic Faster } 4 = 132
  a'8-. r8 b-. r gis-. r a-. r
}
```
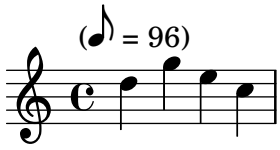


La commande de *markup* \rhythm a ceci de particulier qu'elle permet d'imprimer un motif rythmique – voir l'annexe Section A.1.4 [Markup for music and musical symbols], page 879.

```
\relative {
  \tempo \markup {
    Swing
    \hspace #0.4
    \rhythm { 8[ 8] } = \rhythm { \tuplet 3/2 { 4 8 } }
  }
  b8 g' c, d ees d16 ees d c r8
}
```



Mentionner une indication textuelle vide vous permet de mettre entre parenthèses l'indication numérique :

```
\relative {
  \tempo "" 8 = 96
  d''4 g e c
}
```
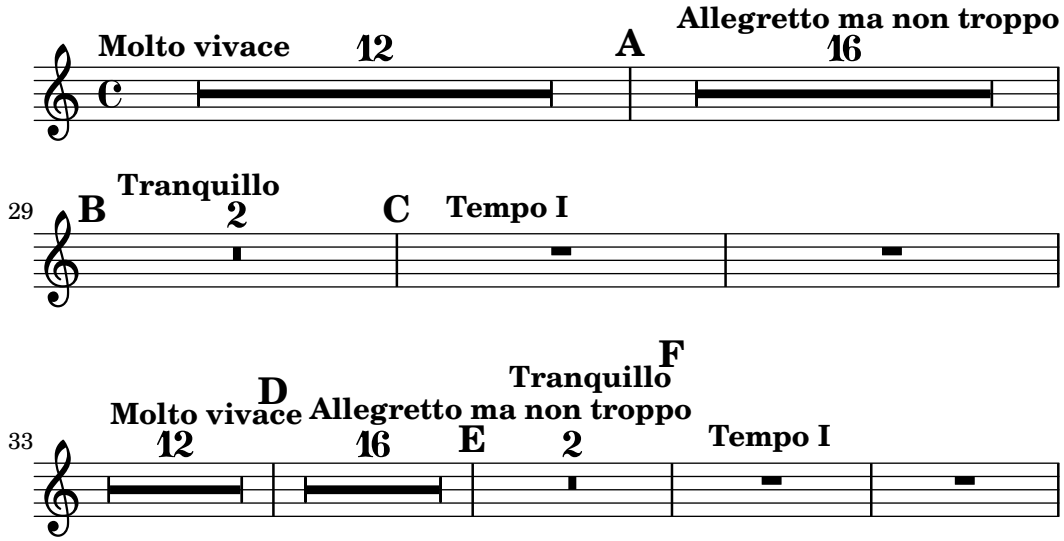


Dans le cas d'une partie où l'instrumentiste a de longs moments de silence (voir Section 2.2.3 [Silences valant une mesure], page 73), les indications de tempo, marques de repère et indications textuelles peuvent être fort rapprochées l'une de l'autre. L'instruction \markLengthOn permet de préserver suffisamment d'espace horizontal (et vertical) de telle sorte que ces indications ne se chevauchent ; l'instruction \markLengthOff restaure le comportement par défaut qui ignore les indications de tempo dans les calculs d'espacement horizontal.

```
\compressMMRests {
  \markLengthOn
  \tempo "Molto vivace"
  R1*12
  \mark \default \tempo "Allegretto ma non troppo"
  R1*16
  \mark \default \tempo "Tranquillo"
  R1*2
```

```
  \mark \default \tempo "Tempo I"
  R1 R1 \break

  \markLengthOff
  \tempo "Molto vivace"
  R1*12
  \mark \default \tempo "Allegretto ma non troppo"
  R1*16
  \mark \default \tempo "Tranquillo"
  R1*2
  \mark \default \tempo "Tempo I"
  R1 R1
}
```



Morceaux choisis

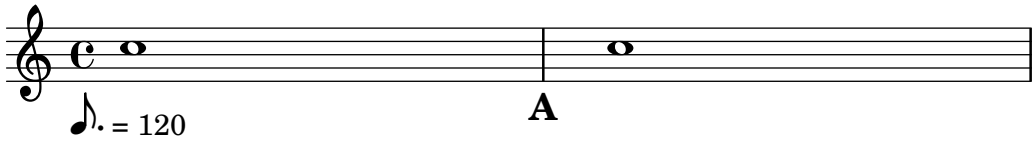
Impression du métronome et des repères sous la portée

Les indications de tempo et les marques de repère s'impriment par défaut au-dessus de la portée. Le fait de régler en conséquence la propriété `direction` des objets `MetronomeMark` et `RehearsalMark` les placera au-dessous de la portée.

```
\layout {
  ragged-right = ##f
}

{
  % Metronome marks below the staff
  \override Score.MetronomeMark.direction = #DOWN
  \tempo 8. = 120
  c''1
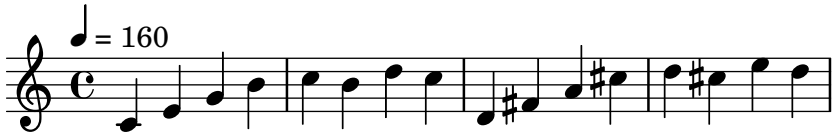
  % Rehearsal marks below the staff
  \override Score.RehearsalMark.direction = #DOWN
  \mark \default
  c''1
}
```



Changement de tempo sans indication sur la partition

Vous pouvez indiquer un changement de tempo pour le fichier MIDI sans pour autant l'imprimer. Il suffit alors de le rendre invisible aux musiciens.
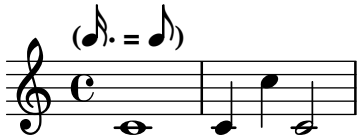
```
\score {
  \new Staff \relative c' {
    \tempo 4 = 160
    c4 e g b
    c4 b d c
    \set Score.tempoHideNote = ##t
    \tempo 4 = 96
    d,4 fis a cis
    d4 cis e d
  }
  \layout { }
  \midi { }
}
```



Création d'une indication métronomique sous forme d'étiquette

Vous pouvez créer des indications de tempo sous la forme d'étiquettes textuelles – des objets *markup* –, notamment des équivalences. Cependant, elles n'apparaîtront pas dans le fichier MIDI.

```
\relative c' {
  \tempo \markup {
    \concat {
      (
      \smaller \general-align #Y #DOWN \note { 16. } #UP
      " = "
      \smaller \general-align #Y #DOWN \note { 8 } #UP
      )
    }
  }
  c1
  c4 c' c,2
}
```



Pour de plus amples détails, veuillez consulter Section 8.2 [Mise en forme du texte], page 329.

Voir aussi

Glossaire musicologique : Section “métronome” dans *Glossaire*, Section “indication métronomique” dans *Glossaire*, Section “indication du tempo” dans *Glossaire*, Section “marque de métronome” dans *Glossaire*.

Manuel de notation : Chapitre 24 [Génération de fichiers MIDI], page 672, Section 8.2 [Mise en forme du texte], page 329, Section 2.2.3 [Silences valant une mesure], page 73.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “MetronomeMark” dans *Référence des propriétés internes*.
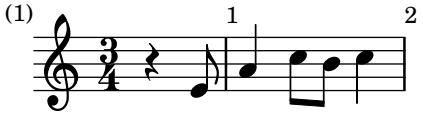
2.3.4 Levées

Les mesures incomplètes, telles que les anacrouses ou levées, doivent être entrées avec la commande `\partial`. La syntaxe de cette commande est

```
\partial durée
```

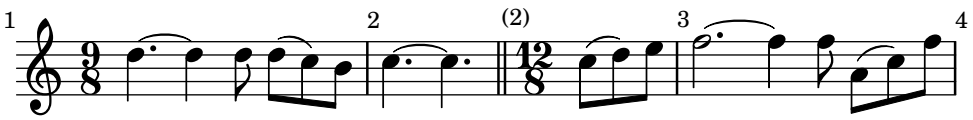
Lorsque `\partial` est utilisé en début de pièce, *durée* égale la valeur rythmique précédant la première mesure.

```
\relative {
  \verboseBarNumbers
  \time 3/4
  \partial 4.
  r4 e'8 | a4 c8 b c4 |
}
```
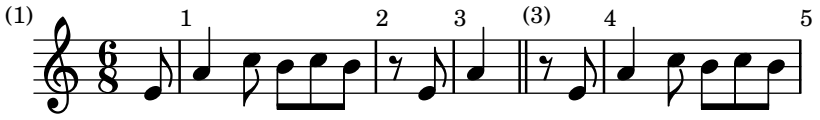


Lorsque `\partial` est utilisé après le début du morceau, *durée* égale la valeur rythmique **restant** dans la mesure en cours. Ceci ne crée pas de nouveau numéro de mesure.

```
\relative {
  \verboseBarNumbers
  \time 9/8
  d''4.~ 4 d8 d( c) b | c4.~ 4. \bar "||"
  \time 12/8
  \partial 4.
  c8( d) e | f2.~ 4 f8 a,( c) f |
}
```



La commande `\partial` est **obligatoire** lorsque la métrique change en cours de mesure, mais peut aussi s’utiliser isolément.

```
\relative {
  \verboseBarNumbers
  \time 6/8
  \partial 8
  e'8 | a4 c8 b[ c b] |
  \partial 4
  r8 e,8 | a4 \bar "||"
  \partial 4
  r8 e8 | a4
  c8 b[ c b] |
}
```



Pour des raisons techniques, l'argument fourni à `\partial` ne saurait être une durée nulle (comme `\partial 4*0`).

Voir aussi

Glossaire musicologique : Section "anacrouse" dans *Glossaire*.

Manuel de notation : Section 2.6.1 [Notes d'ornement], page 150.
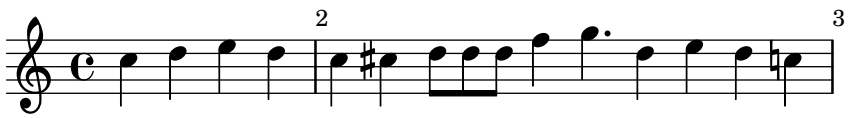
Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "Timing_translator" dans *Référence des propriétés internes*.

2.3.5 Musique sans métrique

Dans un passage tel qu'une cadence, il peut être souhaitable de désactiver la temporisation et tout ce qu'elle entraîne : positionnement des barres et numérotation des mesures, réinitialisation des altérations accidentelles, etc. La musique placée entre `\cadenzaOn` et `\cadenzaOff` ne tient pas compte de la longueur de la mesure.
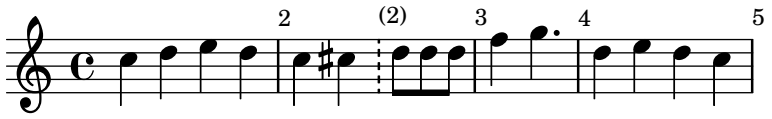
```
\relative c'' {
  % Affiche tous les numéros de mesure
  \override Score.BarNumber.break-visibility = #all-visible
  c4 d e d
  \cadenzaOn
  c4 cis d8[ d d] f4 g4.
  \cadenzaOff
  d4 e d c
}
```



Afin de diviser un passage non mesuré, il suffit de réactiver temporairement la temporisation grâce à l'utilisation de `\partial` pour créer une mesure réduite. La commande `\bar` n'entame pas de nouvelle mesure, même si une barre est imprimée.
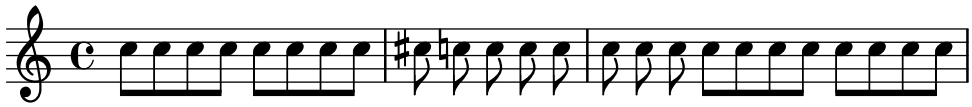
```
cadenzaMeasure = {
  \cadenzaOff
  \partial 1024 s1024
  \cadenzaOn
}

\relative c'' {
  % Affiche tous les numéros de mesure
  \override Score.BarNumber.break-visibility = #all-visible
  c4 d e d
  \cadenzaOn
  c4 cis \bar "!" d8[ d d] \cadenzaMeasure f4 g4.
  \cadenzaMeasure
  \cadenzaOff
  d4 e d c
}
```



L'instruction `\cadenzaOn` désactive les ligatures automatiques ; elles seront réactivées après un `\cadenzaOff`. Toutes les ligatures devront donc être indiquées de manière explicite tout au long de la cadence – voir Section 2.4.3 [Barres de ligature manuelles], page 117.
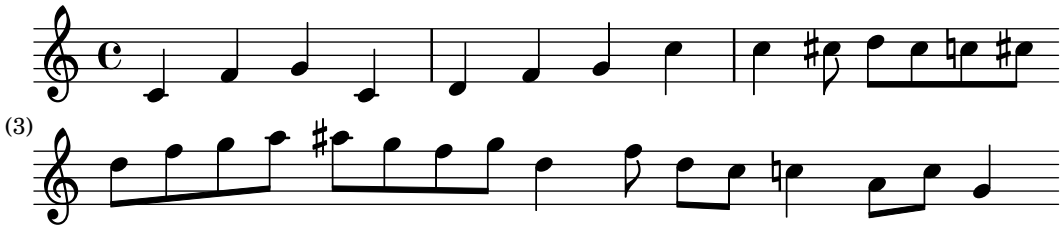
```
\relative {
  \*8 c''8
  \cadenzaOn
  cis8 c c c c
  \bar"|"
  c8 c c
  \cadenzaOff
  \*8 c8
}
```



Ces commandes prédéfinies affecteront toutes les portées de la partition, même si vous ne les placez que dans un simple contexte `Voice`. Pour éviter ce désagrément, transférez le `Timing_translator` du contexte `Score` au contexte `Staff`, comme indiqué au chapitre Section 2.3.6 [Notation polymétrique], page 96.

Les sauts de ligne ou de page sont désactivés dans le cadre d'une cadence : puisqu'il n'y a pas de métrique, il n'est pas possible de déterminer automatiquement où ils seraient appropriés. Il est donc nécessaire, lorsque la cadence est relativement longue, d'insérer des points de rupture potentiels aux endroits appropriés à l'aide de la commande `\allowBreak` ou d'envisager d'autres solutions – voir Section 28.1 [Sauts de ligne], page 711.

```
\relative {
  c'4 f g c, d f g c
  \cadenzaOn
  c4 cis8
  \allowBreak
  d[ cis c cis]
  \allowBreak
  d[ f g a]
  \allowBreak
  ais[ g f g]
  \allowBreak
  d4 f8
  \allowBreak
  d[ cis] c4
  \allowBreak
  a8[ c] g4
}
```



Commandes prédéfinies

`\cadenzaOn`, `\cadenzaOff`.

Voir aussi

Glossaire musicologique : Section “cadence” dans *Glossaire*.

Manuel de notation : Section 1.1.3 [Altérations], page 8, Section 2.4.3 [Barres de ligature manuelles], page 117, Section 2.3.6 [Notation polymétrique], page 96, Section 36.7 [Visibilité des objets], page 812.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

2.3.6 Notation polymétrique

LilyPond gère les métriques composites, aussi bien de manière explicite que de manière détournée – modification de l’indicateur de métrique et échelonnement de la durée des notes.

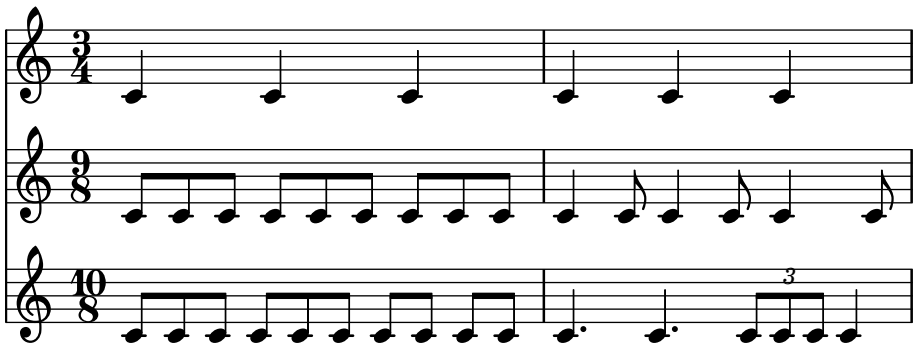
Métriques différentes et mesures d’égale longueur

Il suffit, pour obtenir cette forme de notation, d’utiliser la commande `\polymetric \time` pour définir la métrique locale, et `\scaleDurations` pour que la longueur de mesure corresponde à `Timing.measureLength`. Ceci réglera le même jeu de propriétés que la commande `\time`, à l’exception de `measureLength`.

Désactiver les propriétés locales et retrouver les valeurs du contexte `Timing` s’obtient par un `\polymetric \default`.

L’exemple suivant utilise parallèlement des mesures à 3/4, 9/8 et 10/8, mais 3/4 est choisi comme longueur de référence des mesures. La musique de la première ligne est espacée normalement. Pour la deuxième portée, les durées sont multipliées par 2/3 de telle sorte que $2/3 \times 9/8 = 3/4$; pour la troisième elles sont multipliées par 3/5, de telle sorte que $3/5 \times 10/8 = 3/4$.

```
\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \scaleDurations 2/3 {
      \context Staff \polymetric \time 9/8
      \*3 { c8 c c }
      \*3 { c4 c8 }
    }
  }
  \new Staff {
    \scaleDurations 3/5 {
      \context Staff \polymetric \time 3,3,2,2 10/8
      \*2 { c8 c c }
      \*2 { c8 c } |
      c4. c \tuplet 3/2 { c8 c c } c4
    }
  }
>>
```



Métriques différentes et mesures de longueur inégale

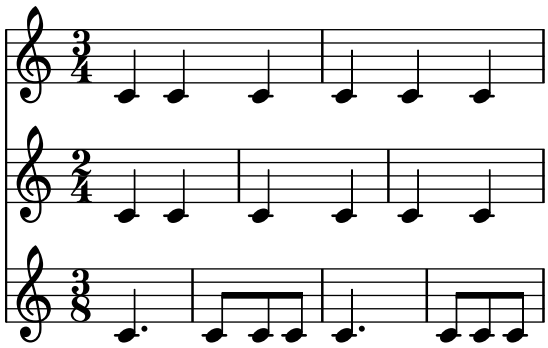
Il arrive aussi que chaque portée ait sa propre métrique. Vous y parviendrez en insérant l'instruction `\enablePerStaffTiming` dans le bloc `\layout`.

```
\layout {
  \enablePerStaffTiming
}

% Chaque portée à désormais sa propre métrique.

\relative <<
  \new Staff {
    \time 3/4
    c'4 c c |
    c4 c c |
  }
  \new Staff {
    \time 2/4
    c4 c |
```

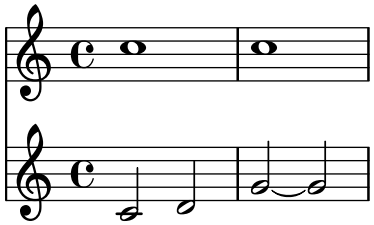
```
    c4 c |
    c4 c |
  }
  \new Staff {
    \time 3/8
    c4. |
    c8 c c |
    c4. |
    c8 c c |
  }
>>
```

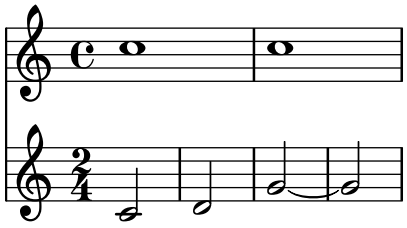


Lorsqu'un seul morceau est polymétrique, l'instruction `\enablePerStaffTiming` sera placée dans la section `\layout` du bloc `\score` considéré.

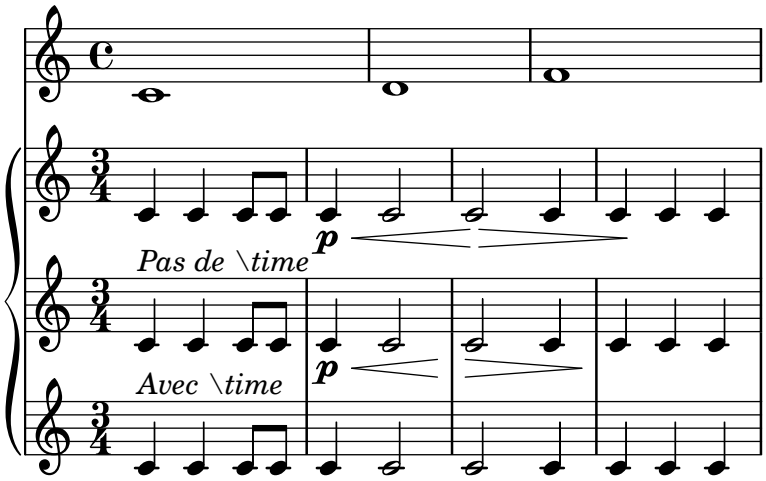
```
\score {
  <<
    \new Staff { c''1 1 }
    \new Staff { c'2 d' g'2~ 2 }
  >>
}

\score {
  \layout {
    \enablePerStaffTiming
  }
  <<
    \new Staff { \time 4/4 c''1 1 }
    \new Staff { \time 2/4 c'2 d' g'2~ 2 }
  >>
}
```





Dans le cadre d'une pièce polymétrique, toutes les portées doivent comporter une instruction `\time` dès lors qu'elles n'ont pas la métrique par défaut de 4/4. Ceci s'applique aussi aux portées spéciales dépourvues de lignes de portée, tels les contextes `Dynamics`, puisque le placement de certains bandeaux comme les soufflets est synchronisé sur les barres de mesure.



Cette fonctionnalité est opérationnelle dans une sortie MIDI dès lors qu'elle est libellée dans le bloc `\midi`.

```
\layout {
  \enablePerStaffTiming
}

\midi {
  \enablePerStaffTiming
}
```

Voir aussi

Glossaire musicologique : Section "polymétrie" dans *Glossaire*, Section "métrique composite" dans *Glossaire*, Section "métrique" dans *Glossaire*.

Manuel de notation : Section 2.1.3 [Changement d'échelle des durées], page 62, Section 2.4.1 [Barres de ligature automatiques], page 104, Section 2.4.3 [Barres de ligature manuelles], page 117, Section 2.3.2 [Métrique], page 81.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "TimeSignature" dans *Référence des propriétés internes*, Section "Timing_translator" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*.
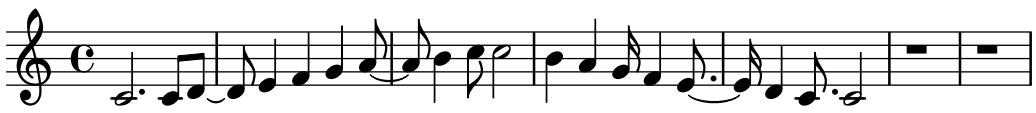
Problèmes connus et avertissements

Bien que les notes de différentes portées tombant au même moment seront positionnées horizontalement au même endroit, les barres de mesure (dans chacune des portées) peuvent causer un espacement incohérent pour chacune des différentes métriques.

2.3.7 Découpage automatique des notes

On peut convertir automatiquement les notes longues en notes liées. Il suffit pour cela de remplacer le graveur `Note_heads_engraver` par le graveur `Completion_heads_engraver`. Il en va de même pour des silences ; le `Completion_rest_engraver` devra alors remplacer le `Rest_engraver`. Dans l'exemple suivant, les notes ou silences qui dépassent de la mesure sont divisés et au besoin liés.
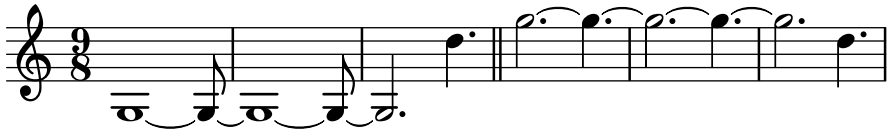
```
\new Voice \with {
  \remove Note_heads_engraver
  \consists Completion_heads_engraver
  \remove Rest_engraver
  \consists Completion_rest_engraver
}
\relative {
  c'2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2 r1*2
}
```



Ces graveurs divisent toutes les notes et silences qui débordent de la mesure, et insèrent des liaisons de prolongation. Dans la pratique, cette fonctionnalité permet de déboguer des partitions complexes : si les mesures ne sont pas entièrement remplies, alors les liaisons de prolongation montrent exactement la durée des décalages de mesure.
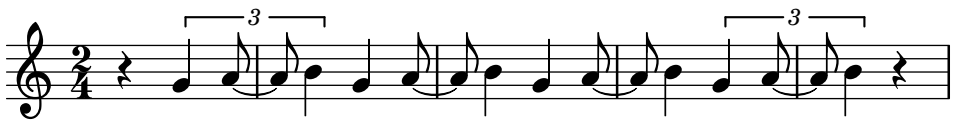
La propriété `completionUnit` détermine la durée de référence pour diviser les notes.

```
\new Voice \with {
  \remove Note_heads_engraver
  \consists Completion_heads_engraver
} \relative {
  \time 9/8 g\breve. d''4. \bar "||"
  \set completionUnit = #3/8
  g\breve. d4.
}
```



Ces graveurs découperont les notes de durée altérée, telles celles d'un triolet, en notes ayant le même facteur d'échelle que les notes saisies.

```
\new Voice \with {
  \remove Note_heads_engraver
  \consists Completion_heads_engraver
} \relative {
  \time 2/4 r4
  \tuplet 3/2 {g'4 a b}
  \scaleDurations 2/3 {g a b}
  g4*2/3 a b
  \tuplet 3/2 {g4 a b}
  r4
}
```



Voir aussi

Glossaire musicologique : Section “liaison de tenue” dans *Glossaire*.

Manuel d’initiation : Section “Ajout et suppression de graveurs” dans *Manuel d’initiation*, Section “Tout savoir sur les graveurs” dans *Manuel d’initiation*.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Références des propriétés internes : Section “Note_heads_engraver” dans *Référence des propriétés internes*, Section “Completion_heads_engraver” dans *Référence des propriétés internes*, Section “Rest_engraver” dans *Référence des propriétés internes*, Section “Completion_rest_engraver” dans *Référence des propriétés internes*, Section “Forbid_line_break_engraver” dans *Référence des propriétés internes*.

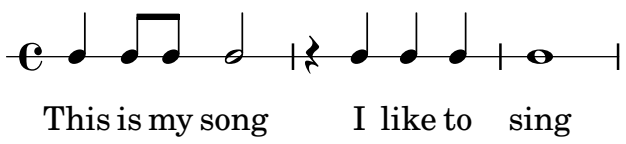
Problèmes connus et avertissements

Pour rester cohérent avec le comportement précédent, les notes ou silences d’une durée supérieure à la mesure, comme un `c1*2`, seront divisés sans être mis à l’échelle – on aura donc `{ c1 c1 }`. La propriété `completionFactor`, qui contrôle ce comportement, peut être désactivée (valorisée à `#f`) pour autoriser les notes ou silences divisés à adopter le facteur d’échelle des durées saisies.

2.3.8 Gravure de lignes rythmiques

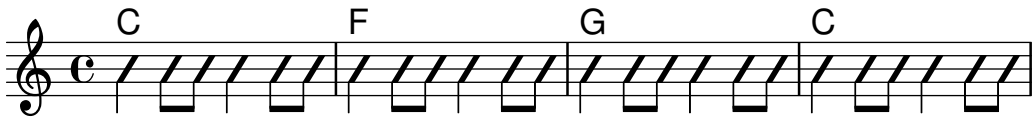
Au moyen d’une portée rythmique – *rhythmic staff* en anglais – on peut montrer seulement le rythme d’une mélodie : toutes les notes sont ramenées à la même hauteur, sur une portée d’une seule ligne.

```
<<
  \new RhythmicStaff {
    \new Voice = "myRhythm" \relative {
      \time 4/4
      c'4 e8 f g2
      r4 g g f
      g1
    }
  }
  \new Lyrics {
    \lyricsto "myRhythm" {
      This is my song
      I like to sing
    }
  }
>>
```



L’utilisation combinée du `Pitch_squash_engraver` et de `\improvisationOn` permet d’afficher la structure rythmique d’une grille d’accords :

```
<<
  \new ChordNames {
    \chordmode {
      c1 f g c
    }
  }
  \new Voice \with {
    \consists Pitch_squash_engraver
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
  }
>>
```



Une musique comportant des accords peut s'utiliser dans un `RhythmicStaff`, et avec le `Pitch_squash_engraver`, dès lors que les accords sont auparavant réduits en notes uniques à l'aide de la fonction musicale `\reduceChords` :

```
\new RhythmicStaff {
  \time 4/4
  \reduceChords {
    <c>2
    <e>2
    <c e g>2
    <c e g>4
    <c e g>4
  }
}
```



Commandes prédéfinies

`\improvisationOn`, `\improvisationOff`.

Morceaux choisis

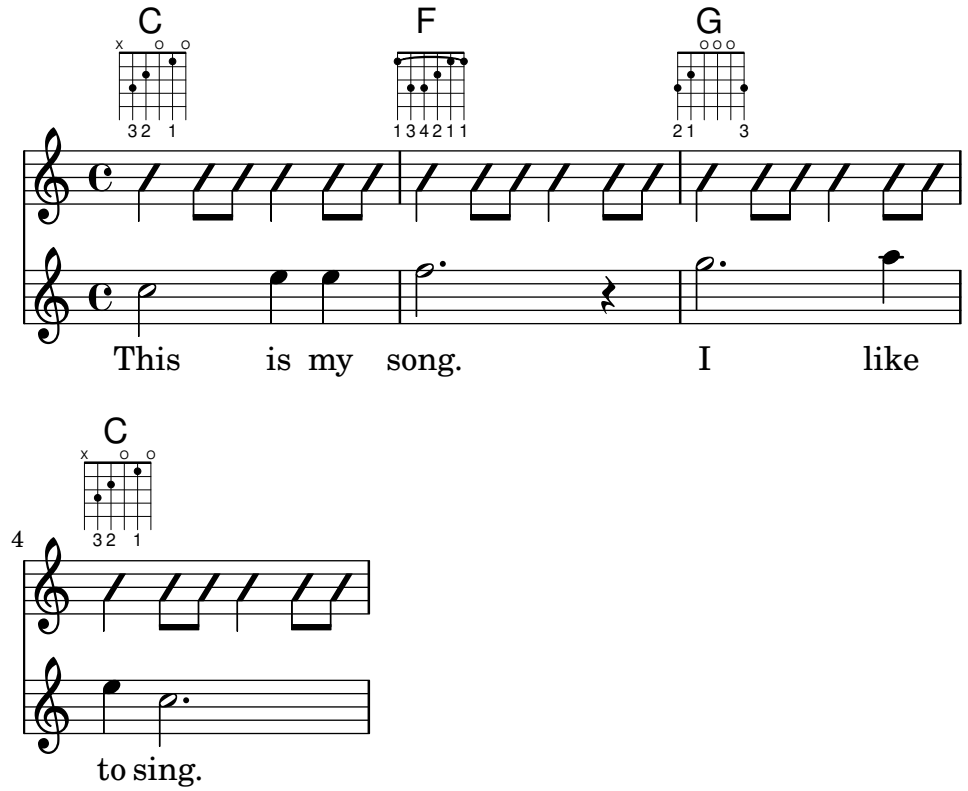
Rythmique et guitare

En matière de notation pour guitare, il arrive que soient indiqués les « coups de gratte » en plus de la mélodie, grilles d'accords et diagrammes de tablature.

```
\include "predefined-guitar-fretboards.ly"

<<
  \new ChordNames \chordmode {
    c1 | f | g | c
  }
```

```
  \new FretBoards \chordmode {
    c1 | f | g | c
  }
  \new Voice \with {
    \consists "Pitch_squash_engraver"
  } \relative c'' {
    \improvisationOn
    c4 c8 c c4 c8 c
    f4 f8 f f4 f8 f
    g4 g8 g g4 g8 g
    c4 c8 c c4 c8 c
  }
  \new Voice = "melody" \relative c'' {
    c2 e4 e4
    f2. r4
    g2. a4
    e4 c2.
  }
  \new Lyrics \lyricsto "melody" {
    This is my song.
    I like to sing.
  }
>>
```



Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “RhythmicStaff” dans *Référence des propriétés internes*, Section “Pitch_squash_engraver” dans *Référence des propriétés internes*.

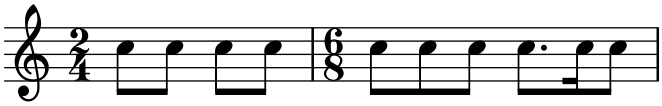
2.4 Barres de ligature

LilyPond dispose de deux façons d'indiquer des ligatures, l'une automatisée, l'autre par saisie manuelle, les deux pouvant se combiner.

2.4.1 Barres de ligature automatiques

LilyPond décide automatiquement de la manière de grouper les notes et d'imprimer les ligatures.
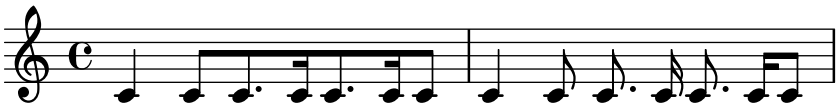
```
\relative c'' {
  \time 2/4 c8 c c c
  \time 6/8 c8 c c c8. c16 c8
}
```



Lorsque ce comportement automatisé n'est pas satisfaisant, on peut définir des groupements manuellement – voir Section 2.4.3 [Barres de ligature manuelles], page 117. Dans le cas où le groupe de notes en question contient un silence, il est **impératif** d'indiquer explicitement les début et fin de la ligature.

Lorsque les ligatures automatiques ne sont pas nécessaires, il suffit de désactiver la fonctionnalité par un `\autoBeamOff` – réactivation par `\autoBeamOn` :

```
\relative c' {
  c4 c8 c8. c16 c8. c16 c8
  \autoBeamOff
  c4 c8 c8. c16 c8.
  \autoBeamOn
  c16 c8
}
```



> **Note :** Si des ligatures sont utilisées dans la musique d'une chanson (pour indiquer des mélismes), les ligatures automatiques doivent être désactivées avec `\autoBeamOff`, et indiquées manuellement.
> L'utilisation conjointe de `\partCombine` et de `\autoBeamOff` peut produire des résultats quelque peu surprenants ; ceci fait l'objet d'un exemple particulier à la rubrique Morceaux choisis.

Des règles de dérogation au comportement automatique par défaut sont possibles ; voir Section 2.4.2 [Définition des règles de ligature automatique], page 107.

Commandes prédéfinies

`\autoBeamOff`, `\autoBeamOn`.

Morceaux choisis

Ligature au moment d'un saut de ligne

Il est normalement impensable qu'un saut de ligne tombe au milieu d'une ligature. LilyPond le permet néanmoins en activant la propriété `Beam.breakable`.

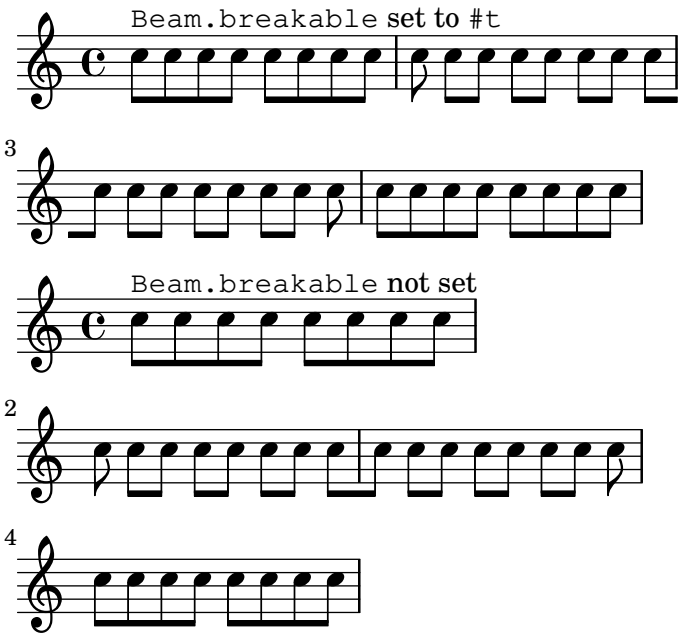
Cette propriété n'affecte pas les sauts manuels insérés par les commandes comme `\break`.

```
music = {
  \*8 c8
  c8 \*7 { c[ c] }  c
  \*8 c8
}

\relative c'' {
  <>^\markup { \typewriter Beam.breakable set to \typewriter "#t" }
  \override Beam.breakable = ##t
  \music
}

\relative c'' {
  <>^\markup { \typewriter Beam.breakable not set }
  \music
}
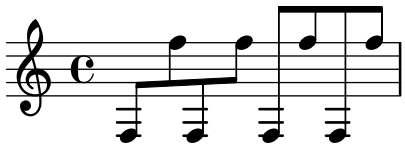
\paper {
  line-width = 100\mm
}
```



Ligature et directions de hampe inversées

LilyPond insère automatiquement des ligatures coudées – certaines hampes vers le haut, d'autres vers le bas – lorsqu'il détecte un intervalle important entre des têtes de notes. Ce comportement peut être changé par l'intermédiaire de l'objet `auto-knee-gap` – défini par défaut à 5,5 espaces, plus la largeur et la pente de la ligature en question.

```
{
  f8 f''8 f8 f''8
  \override Beam.auto-knee-gap = #6
  f8 f''8 f8 f''8
}
```



PartCombine et autoBeamOff

La fonction \autoBeamOff dans le cadre d'un \partCombine agit de façon bien particulière ; c'est pourquoi il vaut mieux tout d'abord recourir à

```
\set Staff.autobeaming = ##f
```

pour désactiver les ligatures automatiques pour l'ensemble de la portée concernée.
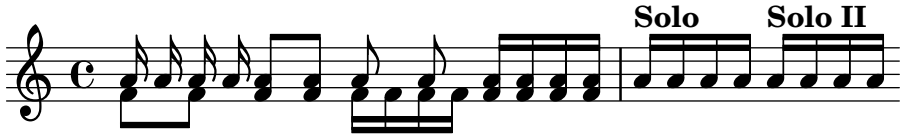
L'instruction \partCombine fonctionne de manière interne sur la base de quatre voix : solo hampes montantes, solo hampes descendantes, ensemble et solo.

Lorsque \autoBeamOff apparaît dans le premier argument de la combinaison, il s'applique à la voix active à ce moment précis, qu'il s'agisse du solo hampes montantes ou du combiné hampes montantes. Lorsqu'elle est introduite dans le second argument, la commande \autoBeamOff s'appliquera au solo hampes descendantes.

Vous devrez donc, afin que \autoBeamOff soit pleinement opérationnel dans le cadre d'un \partCombine, l'introduire aux **quatre** niveaux.

```
{
  % \set Staff.autoBeaming = ##f % turns off all auto-beaming

  \partCombine {
    \autoBeamOff   % applies to split up-stems
    \*4 a'16
    % \autoBeamOff % applies to combined stems
    \*4 a'8
    \*4 a'16
    % \autoBeamOff % applies to solo
    \*4 a'16
    r4
  } {
    % \autoBeamOff % applies to split down-stems
    \*4 f'8
    \*8 f'16 |
    r4
    \*4 a'16
  }
}
```



Voir aussi

Manuel de notation : Section 2.4.3 [Barres de ligature manuelles], page 117, Section 2.4.2 [Définition des règles de ligature automatique], page 107.

Fichiers d'initialisation : `scm/auto-beam.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Références des propriétés internes : Section "Auto_beam_engraver" dans *Référence des propriétés internes*, Section "Beam_engraver" dans *Référence des propriétés internes*, Section "Beam" dans *Référence des propriétés internes*, Section "BeamEvent" dans *Référence des propriétés internes*, Section "BeamForbidEvent" dans *Référence des propriétés internes*, Section "beam-interface" dans *Référence des propriétés internes*, Section "unbreakable-spanner-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les propriétés d'une ligature sont déterminées **dès le début** de sa construction ; toute adaptation qui interviendrait avant sa terminaison ne sera prise en compte qu'à l'occasion de la **prochaine** ligature.

2.4.2 Définition des règles de ligature automatique

Lorsque la fonction de ligature automatique est active, le positionnement des ligatures dépend des trois propriétés `beatBase`, `beatStructure` et `beamExceptions`. Chacune de ces propriétés peut être adaptée, jusqu'à la prochaine commande `\time`, comme indiqué ci-après, ou bien carrément être modifiées – voir Section 2.3.2 [Métrique], page 81.

Dès lors qu'une règle affectant `beamExceptions` est applicable, elle prendra le pas sur `beatBase` et `beatStructure`.

Ligature basée sur `beatBase` et `beatStructure`
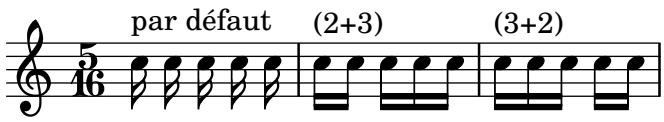
`beamExceptions` dispose par défaut de règles pour les métriques les plus courantes ; il est donc impératif de les invalider pour gérer les ligatures automatiques à l'aide de `beatBase` et `beatStructure`. Les règles de `beamExceptions` se désactivent par un

```
\set Timing.beamExceptions = #'()
```
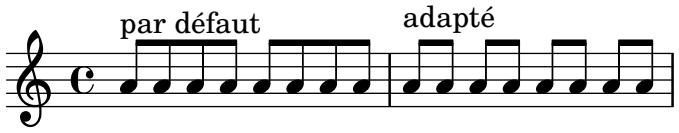
Lorsque `beamExceptions` est défini à `#'()`, que ce soit explicitement ou en raison de l'absence de règles par défaut de `beamExceptions` pour la métrique en vigueur, la terminaison des ligatures est directement liée à la pulsation telle que spécifiée par les propriétés `beatBase` et `beatStructure`.

La commande `\time` réinitialise `beatBase`, `beatStructure` et `beamExceptions`. C'est pour cette raison que les adaptations doivent se placer à la suite de la commande `\time`, non au préalable.

```
\relative c'' {
  \time 5/16
  c16^"par défaut" c c c c |
  % beamExceptions n'est probablement pas définie pour 5/16,
  % mais désactivons les par sûreté
  \set Timing.beamExceptions = #'()
  \set Timing.beatStructure = 2,3
  c16^"(2+3)" c c c c |
  \set Timing.beatStructure = 3,2
  c16^"(3+2)" c c c c |
}
```

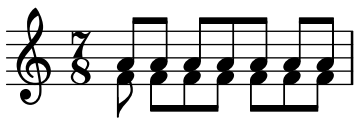



```
\relative {
  \time 4/4
  a'8^"par défaut" a a a a a a a
  % Désactivons beamExceptions parce sûrement
  % définie pour 4/4
  \set Timing.beamExceptions = #'()
  \set Timing.beatBase = #1/4
  \set Timing.beatStructure = 1,1,1,1
  a8^"adapté" a a a a a a a
}
```



Les effets de règles de ligature peuvent être restreints à un contexte particulier. En l'absence de règle particulière déterminée dans un contexte de niveau inférieur, les règles définies au niveau directement supérieur s'appliqueront.
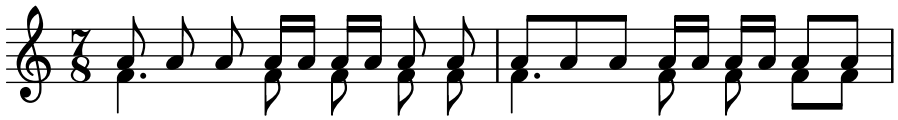
```
\new Staff {
  \time 7/8
  % Pas besoin de désactiver beamExceptions
  % car il n'y en a pas de définie en 7/8
  \set Staff.beatStructure = 2,3,2
  <<
    \new Voice = one {
      \relative {
        a'8 a a a a a a
      }
    }
    \new Voice = two {
      \relative {
        \voiceTwo
        \set Voice.beatStructure = 1,3,3
        f'8 f f f f f f
      }
    }
  >>
}
```



Lorsque plusieurs voix cohabitent sur une même portée et que les règles de ligature doivent s'appliquer sans distinction, il faut spécifier que ces règles affectent le contexte `Staff` :
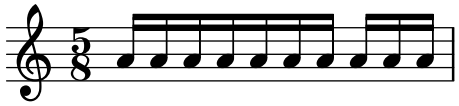
```
\time 7/8
% rythme 3-1-1-2
% Changement appliqué à Voice par défaut -- ne marche pas bien.
% Parce que voix auto-générées, toutes les pulsations seront à
% beatBase #1/8
\set beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>

% Fonctionne correctement après spécification du contexte Staff
\set Staff.beatStructure = 3,1,1,2
<< \relative {a'8 a a a16 a a a a8 a} \\ \relative {f'4. f8 f f f} >>
```



Vous pouvez ajuster la valeur de beatBase afin d'obtenir des ligatures selon vos besoin. Notez cependant que la valeur de beatStructure devra être en corrélation avec cette nouvelle valeur de beatBase.

```
\time 5/8
% Pas besoin de désactiver beamExceptions
% car pas définie pour 5/8
\set Timing.beatBase = #1/16
\set Timing.beatStructure = 7,3
\*10 a'16
```
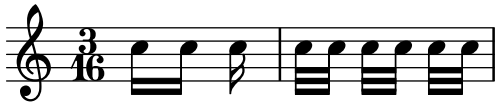


Ligature basée sur beamExceptions

Les règles spécifiques autres que celles concernant la terminaison des ligatures sont gérées par la propriété beamExceptions.

La valeur de beamExceptions, structure de données Scheme plutôt complexe, est plus facilement générée par la fonction \beamExceptions. Cette fonction reçoit au moins un schéma de ligature manuelle sur la durée d'une mesure (les mesures doivent être séparées par un test de mesure | puisque cette fonction n'a pas d'autre moyen de déterminer la longueur de la mesure). En voici un simple exemple :

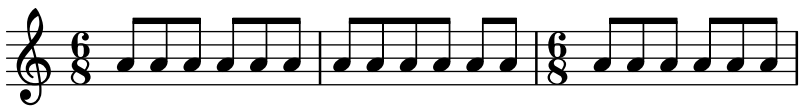
```
\relative c'' {
  \time 3/16
  \set Timing.beatStructure = 2,1
  \set Timing.beamExceptions =
    \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }
  c16 c c |
  \*6 c32 |
}
```



Note : La propriété `beamExceptions` doit répertorier absolument **toutes** les exceptions. Il n'est en effet pas possible d'en ajouter, modifier ou supprimer *a posteriori*. Cela peut paraître fastidieux, mais toutes les règles de ligature devraient être appréciées avant de les spécifier.

Lorsqu'intervient un changement de métrique, les valeurs par défaut de `Timing.beatBase`, `Timing.beatStructure` et `Timing.beamExceptions` sont réinitialisées. Il suffit donc, pour revenir aux règles de ligature par défaut d'un contexte `Timing`, de spécifier à nouveau la métrique.
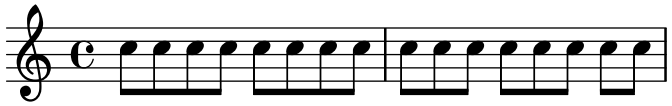
```
\relative a' {
  \time 6/8
  \*6  a8
  % groupe (4 + 2)
  \set Timing.beatStructure = 4,2
  \*6 a8
  % retour au réglage par défaut
  \time 6/8
  \*6 a8
}
```



Les règles de ligature automatique par défaut sont répertoriées, par métrique, dans le fichier `scm/time-signature-settings.scm`. Les manières de déroger à ce comportement sont abordées au chapitre Section 2.3.2 [Métrique], page 81.

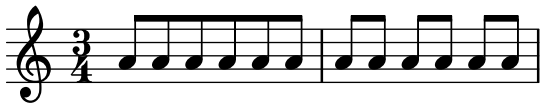
De nombreuses règles de ligature automatique comportent une clé `beamExceptions`. Par exemple, s'il n'y a que des croches dans une mesure à 4/4, celles-ci seront réparties en deux groupes. Le fait de ne pas réinitialiser `beamExceptions` lors d'un aménagement de la pulsation – l'élément `beatStructure` – empêchera l'application de cette dérogation.

```
\time 4/4
\set Timing.beatBase = #1/8
\set Timing.beatStructure = 3,3,2
% Pas de ligature en (3 3 2) à cause de beamExceptions
\*8 c''8 |
% On ligature en (3 3 2) car beamExceptions vidée
\set Timing.beamExceptions = #'()
\*8  c''8
```
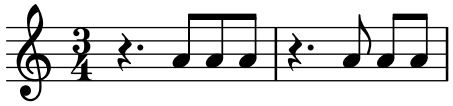


De la même manière, les croches d'une mesure à 3/4 sont ligaturées sur la mesure par défaut. Ligaturer sur le temps requiert un appel à `beamExceptions`.

```
\time 3/4
% Ligaturé en (6) à cause de beamExceptions
\repeat unfold 6 {a'8} |
% On ligature en (1 1 1) grâce à beatBase et beatStructure par défaut
\set Timing.beamExceptions = #'()
\repeat unfold 6 {a'8}
```



Certaines partitions des périodes romantique ou classique font apparaître des ligatures sur la moitié d'une mesure à 3/4 (ou à 3/8), ce qui va à l'encontre de la règle établie – comme le fait remarquer Gould à la page 153 de son ouvrage – puisque donne l'impression d'une mesure à 6/8. Il en va de même pour une mesure à 3/8. La reproduction d'un tel comportement se contrôle à l'aide de la propriété de contexte `beamHalfMeasure`, qui d'ailleurs ne sera effective que lorsque le numérateur de la métrique est un 3.

```
\relative a' {
  \time 3/4
  r4. a8 a a |
  \set Timing.beamHalfMeasure = ##f
  r4. a8 a a |
}
```



Principes de la ligature automatique

Lorsqu'elle est activée, la gestion automatisée des ligatures est directement liée aux propriétés `beatBase`, `beatStructure` et `beamExceptions`.

Les règles déterminant le positionnement des ligatures automatiques s'appliquent dans l'ordre suivant de priorité :

- une ligature explicite – indiquée par `[...]` – sera toujours respectée ; sinon
- si une règle explicite de terminaison a été définie grâce à la propriété `beamExceptions` pour un type de ligature spécifique dans la métrique en cours, c'est elle qui s'appliquera ; sinon
- si une règle explicite de terminaison a été définie grâce à la propriété `beamExceptions` pour un type de ligature plus large, c'est elle qui s'appliquera aux emplacements valides ; sinon
- utilisation des valeurs de `beatBase` et `beatStructure` pour regrouper les notes par des ligatures, selon la pulsation interne de la mesure.

Le *type de ligature* correspond à la durée la plus courte dans le groupe.

Les règles de ligature par défaut sont répertoriées dans le fichier `scm/time-signature-settings.scm`.

Morceaux choisis

Subdivision des ligatures

Les ligatures d'une succession de notes de durée inférieure ou égale à la double croche ne sont pas subdivisées par défaut. Autrement dit, tous les traits de ligature (deux ou plus) seront continus. Ce comportement peut être modifié afin de diviser la ligature en sous-groupes grâce à la propriété `subdivideBeams`. Lorsqu'elle est activée, un certain nombre de traits de ligature entre deux hampes est supprimé à des intervalles correspondant à la durée de sous-groupe souhaitée.

Les propriétés `beamMinimumSubdivision` et `beamMaximumSubdivision` permettent de configurer les limites de la subdivision automatique des ligatures : l'intervalle rythmique minimum auquel subdiviser et le nombre de tronçons supprimés selon cet intervalle. Leur valeur par défaut est respectivement de `0` et `+inf.0`, ce qui demandera à LilyPond de subdiviser le plus possible.

Certains cas particulier doivent être pris en considération.

- Lorsque le nombre fourni en numérateur à `beamMaximumSubdivision` n'est pas une puissance de 2, la plus petite durée pour subdiviser sera de `beamMaximumSubdivision` divisé par une puissance de 2 tout en restant supérieur ou égal à `beamMinimumSubdivision`.
- Dès lors que `beamMaximumSubdivision` est inférieur à `beamMinimumSubdivision`, la profondeur de subdivision se limite à `beamMaximumSubdivision` mais pas la fréquence ou l'intervalle, ce qui peut dévier de la valeur métrique à laquelle on est en droit de s'attendre.

Lorsque la propriété `respectIncompletBeams` est activée, les subdivisions incomplètes à plus de deux hampes sont considérées comme « extension » du sous-groupe précédent, autrement dit, le groupe de subdivision précédente sera étendu pour couvrir aussi la subdivision incomplète. Lorsqu'elle est désactivée, ce qui est le cas par défaut, une nouvelle subdivision est alors entamée.

```
\relative c'' {
  \time 1/4

  <>^"default"
  c32 c c c c c c c

  <>^"with subdivision"
  \set subdivideBeams = ##t
  c32 c c c c c c c

  <>^"min 1/8"
  \once \set beamMinimumSubdivision = #1/8
  c32 c c c c c c c

  <>^"max 1/16"
  \once \set beamMaximumSubdivision = #1/16
  c32 c c c c c c c

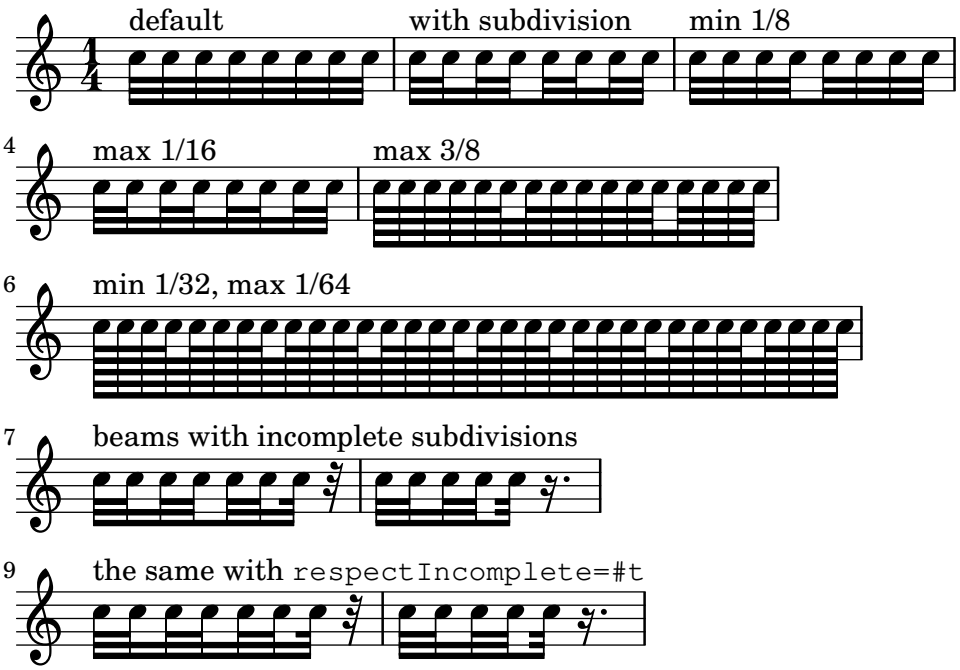
  <>^"max 3/8"
  \once \set beamMaximumSubdivision = #3/8
  \*16 c64

  <>^"min 1/32, max 1/64"
  % Set maximum beam subdivision interval to 1/64 to limit
  % subdivision depth, despite not being metrically correct.
  \once \set beamMinimumSubdivision = #1/32
```

```
  \once \set beamMaximumSubdivision = #1/64
  \*32 c128
  \break

  <>^"beams with incomplete subdivisions"
  c32 c c c c c c r32
  c32 c c c c r16.
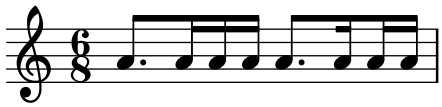
  <>^\markup { "the same with"
               \typewriter { "respectIncomplete=#t" } }
  \set respectIncompleteBeams = ##t
  % The incomplete subgroup extends the completed subgroup.
  c32 c c c c c c r32
  % No visual change since we have only two stems in the
  % incomplete subgroup.
  c32 c c c c r16.
}
```



Ligature à la pulsation

Une sous-ligature tronquée peut pointer en direction de la pulsation à laquelle elle se rattache. Dans l'exemple suivant, la première ligature évite toute troncature (comportement par défaut), alors que la deuxième respecte rigoureusement la pulsation.

```
\relative c'' {
  \time 6/8
  a8. a16 a a
  \set strictBeatBeaming = ##t
  a8. a16 a a
}
```



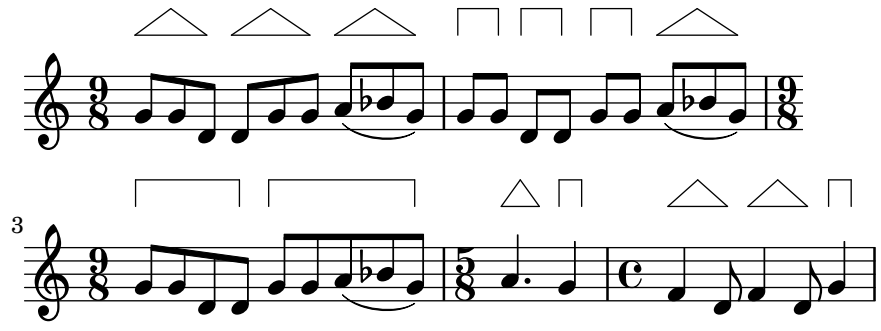
Signes de direction, signes de sous-groupe

Les regroupement par temps au sein d'une mesure sont contrôlés par des propriétés de contexte : `beatStructure` liste la longueur de chaque temps, en unités de `beatBase`. Les valeurs par défaut sont répertoriées dans le fichier `scm/time-signature-settings.scm`. Ces propriétés sont modifiables grâce à la commande `\set`.

Par ailleurs, l'instruction `\time` accepte des règles de pulsation différentes des valeurs par défaut. Dans la mesure où `\time` s'applique au contexte `Timing`, elle ne redéfinira pas les valeurs de `beatStructure` ou `beatBase` lorsqu'elles sont modifiées dans un contexte de niveau inférieur comme `Voice` par exemple.

Si l'on fait appel au `Measure_grouping_engraver`, la fonction `set-time-signature` créera aussi des symboles `MeasureGrouping`. Ces symboles aident à la lecture des œuvres modernes à la rythmique complexe. Dans l'exemple qui suit, la mesure à 9/8 est divisée en 2, 2, 2 et 3, alors que la mesure à 5/8 répond aux règles par défaut contenues dans le fichier `scm/time-signature-settings.scm`. Pour une mesure à 4/4, il faudra explicitement définir `beatBase` en croches afin que le motif irrégulier de la mesure soit correctement rendu.
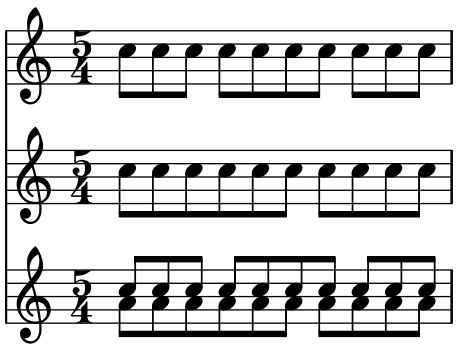
```
\score {
  \new Voice \relative c'' {
    \time 9/8
    g8 g d d g g a( bes g) |
    \set Timing.beatStructure = 2,2,2,3
    g8 g d d g g a( bes g) |
    \time 4,5 9/8
    g8 g d d g g a( bes g) |
    \time 5/8
    a4. g4 |
    \time 3,3,2 4/4
    \set Timing.beatBase = #1/8
    f4 d8 f4 d8 g4
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```



Définition de règles de ligature pour la partition

Les règles de ligature définies au niveau du contexte `Score` s'appliqueront à toutes les portées. Il est toutefois possible de moduler au niveau `Staff` ou `Voice` :

```
\relative c'' {
  \time 5/4
  % Set default beaming for all staves
  \set Score.beatBase = #1/8
  \set Score.beatStructure = 3,4,3
  <<
    \new Staff {
      c8 c c c c c c c c c
    }
    \new Staff {
      % Modify beaming for just this staff
      \set Staff.beatStructure = 6,4
      c8 c c c c c c c c c
    }
    \new Staff {
      % Inherit beaming from Score context
      <<
        {
          \voiceOne
          c8 c c c c c c c c c
        }
        % Modify beaming for this voice only
        \new Voice {
          \voiceTwo
          \set Voice.beatStructure = 6,4
          a8 a a a a a a a a a
        }
      >>
    }
  >>
}
```



Voir aussi

Manuel de notation : Section 2.3.2 [Métrique], page 81.

Fichiers d'initialisation : `scm/time-signature-settings.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

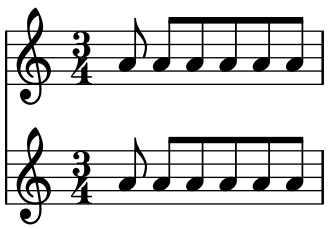
Référence des propriétés internes : Section "Auto_beam_engraver" dans *Référence des propriétés internes*, Section "Beam" dans *Référence des propriétés internes*, Section "BeamForbidEvent" dans *Référence des propriétés internes*, Section "beam-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Si une partition se termine alors qu'une ligature automatique est restée inachevée, cette dernière ligature ne sera pas imprimée du tout. C'est également valable dans le cas d'une musique polyphonique saisie avec la syntaxe << ... \\ ... >>, où une voix se terminerait sans que la dernière ligature ne soit achevée. Le plus simple, en pareil cas, est de spécifier manuellement les dernières ligatures.

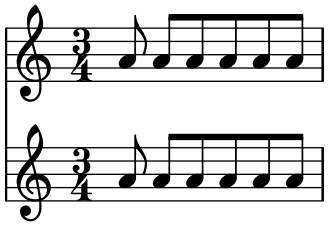
Le traducteur `Timing` est par défaut affecté au contexte `Score`. Définir la métrique dans une portée aura donc des effets sur les ligatures de toutes les autres. Par voie de conséquence, la définition de la métrique apparaissant dans une autre portée annulera les aménagements précédemment apportés aux règles de ligature. Il est donc préférable, pour éviter tout désagrément, de ne spécifier la métrique que dans une seule portée.

```
<<
  \new Staff {
    \time 3/4
    \set Timing.beatBase = #1/8
    \set Timing.beatStructure = 1,5
    \set Timing.beamExceptions = #'()
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \*6 a'8
  }
>>
```



Vous pouvez adapter les règles de ligature par défaut pour une métrique particulière de telle sorte que ces règles que vous aurez définies soient toujours prises en compte. La modification des règles de ligature automatiques est abordée au chapitre Section 2.3.2 [Métrique], page 81.
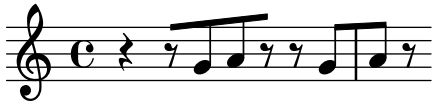
```
<<
  \new Staff {
    \overrideTimeSignatureSettings
      3/4               % timeSignature
      #1/8              % beatBase
      1,5               % beatStructure
      #'()              % beamExceptions
    \time 3/4
    \repeat unfold 6 { a'8 }
  }
  \new Staff {
    \time 3/4
    \*6 a'8
  }
>>
```



2.4.3 Barres de ligature manuelles
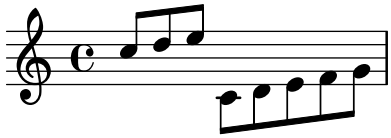
Dans certaines situations, il peut s'avérer nécessaire de supplanter l'algorithme de regroupement automatique des notes, par exemple pour prolonger une ligature par-dessus un silence ou une barre de mesure, ou bien pour suivre le rythme des paroles plutôt que celui des notes. Le début et la fin de la ligature sont alors indiqués respectivement par `[` et `]`.

```
r4 r8[ g' a r] r8 g[ | a] r
```
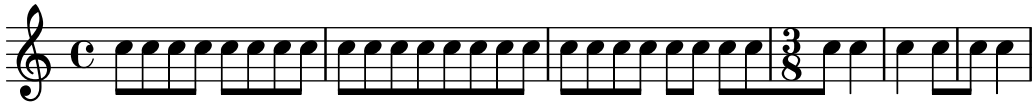


Le positionnement des ligatures manuelles se détermine comme pour toute indication attachée à une note :

```
\relative { c''8^[ d e] c,_[ d e f g] }
```
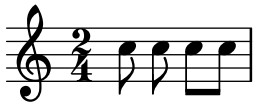


Au lieu de prendre totalement le contrôle des ligatures, il suffit souvent de transmettre à l'algorithme d'automatisation quelques suggestions. Un `\beamBreak` requiert de l'algorithme qu'il cesse automatiquement la ligature créée. `\noBeamBreak`, , à l'inverse, force le regroupement de deux notes adjacentes par une ligature. Ceci peut même causer la création d'une nouvelle ligature.

```
{
  c''8 8 8 8 8 8 8 8 |
  c''8 8 8 8 \noBeamBreak 8 8 8 8 |
  8 8 8 8 8 8 \beamBreak 8 8 \noBeamBreak |
  \time 3/8 |
  8 4 |
  4 8 \noBeamBreak |
  8 4 |
}
```



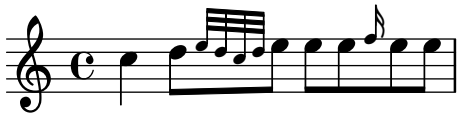
Le fait d'affubler une note particulière d'un `\noBeam` aura pour effet de l'empêcher d'être ligaturée :

```
\relative {
  \time 2/4
  c''8 c\noBeam c c
}
```
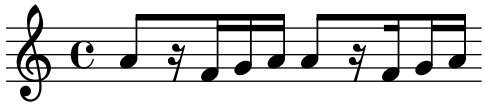


Notes d'ornement et normales font l'objet d'un traitement distinct. Il est dont possible de ligaturer ou non des notes d'ornement sans gêner ce qui est en place au niveau de la notation normale.

```
\relative {
  c''4 d8[
  \grace { e32 d c d }
  e8] e[ e
  \grace { f16 }
  e8 e]
}
```



LilyPond peut déterminer automatiquement les sous-groupes à l'intérieur d'un groupement de notes, bien que le résultat ne soit pas toujours optimal. Les propriétés `stemLeftBeamCount` et `stemRightBeamCount` permettent alors d'ajuster ce comportement pour la note qui vient. Lorsque l'une ou l'autre de ces propriétés est définie, elle ne s'applique qu'une seule fois, après quoi sa définition est effacée. Dans l'exemple qui suit, le dernier fa n'a de ligature supplémentaire que sur sa gauche ; autrement dit, c'est la ligature à la croche qui est importante.

```
\relative a' {
  a8[ r16 f g a]
  a8[ r16
  \set stemLeftBeamCount = 2
  \set stemRightBeamCount = 1
  f16
  \set stemLeftBeamCount = 1
  g16 a]
}
```



Commandes prédéfinies

`\noBeam`.

Morceaux choisis

Débordement de ligature

En combinant `stemLeftBeamCount`, `stemRightBeamCount` et des paires de `[]` attachées à des notes isolées, vous pourrez obtenir des crochets rectilignes aux allures de ligatures.

Pour des crochets rectilignes à droite sur des notes isolées, il suffit d'ajouter une paire d'indicateurs de ligature `[]` et de déterminer `stemLeftBeamCount` à zéro, comme dans l'exemple 1.

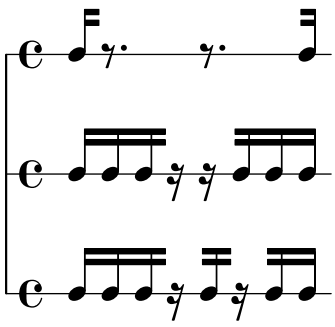
Pour des crochets rectiligne à gauche, c'est `stemRightBeamCount` qu'il faudra déterminer (exemple 2).

Pour que les barres de ligature débordent sur la droite, `stemRightBeamCount` doit avoir une valeur positive ; pour un débordement à gauche, c'est sur `stemLeftBeamCount` qu'il faut jouer. Tout ceci est illustré par l'exemple 3.

Il est parfois judicieux, lorsqu'une note est encadrée de silences, de l'affubler de crochets rectilignes de part et d'autre. L'exemple 4 montre qu'il suffit d'adjoindre à cette note un `[]`.

Notez bien que `\set stemLeftBeamCount` sera toujours synonyme de `\once \set`. Autrement dit, la détermination des ligatures n'est pas « permanente » ; c'est la raison pour laquelle les crochets du `c'16[]` isolé du dernier exemple n'ont rien à voir avec le `\set` indiqué deux notes auparavant.

```
\score {
  <<
    \new RhythmicStaff {
      \set stemLeftBeamCount = 0
      c16[] r8.
      r8.
      \set stemRightBeamCount = 0
      16[]
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r r
      \set stemLeftBeamCount = 2
      16 16 16
    }
    \new RhythmicStaff {
      16 16
      \set stemRightBeamCount = 2
      16 r16
      16[] r16
      \set stemLeftBeamCount = 2
      16 16
    }
  >>
}
```



Crochets de style alternatif

Une dérogation à la propriété `stencil` de l'objet `Flag` permet aux croches et notes de durée inférieure d'adopter une autre forme de crochet. Sont disponibles les variantes `modern-straight-flag`, `old-straight-flag` et `flat-flag`. Un `\revert` permet de retrouver l'allure par défaut.

Des crochets empilés, autrement dit à l'espacement resserré, s'obtiennent à l'aide de l'instruction `\flagStyleStacked`, qui s'annule par un `\flagStyleDefault`.

Une dérogation au stencil de `Flag` ne modifie en rien le positionnement vertical individuel des crochets. Ceci s'observe avec des crochets rectilignes : LilyPond n'ajuste pas dynamiquement l'écart entre les crochets les uns par rapport aux autres de la même manière que pour les ligatures. L'une des solutions pour harmoniser l'apparence consiste à remplacer les crochets par des demi-ligatures comme indiqué dans la deuxième portée . ceci ne peut toutefois se faire automatiquement. Dans le code de cet extrait, ces demi-ligatures se mentionnent à l'aide du préfixe `@`, comme par exemple `@c8`.

Il est important de noter que des demi-ligatures *ne sont pas* des objets `Flag`, ce qui signifie que des modifications apportées aux objets `Flag` n'auront aucun effet sur elles (il faut utiliser des propriétés de `Beam`), et les propriétés de leur objet `Stem` associé adoptera le même comportement qu'avec des ligatures.

```
"@" =
#(define-music-function (music) (ly:music?)
  #{ \set stemLeftBeamCount = 0 $music [] #})

testnotes = {
  \autoBeamOff
  c8 d16 e''32 f64 \acciaccatura { g,,,8 } a128 b
}

\relative c' {
  \override TextScript.staff-padding = 6
  \time 1/4
    <>^"default" \testnotes
  \override Flag.stencil = #modern-straight-flag
    <>_"modern straight" \testnotes
  \override Flag.stencil = #old-straight-flag
    <>^"old straight" \testnotes
  \override Flag.stencil = #flat-flag
    <>_"flat" \testnotes
  \revert Flag.stencil

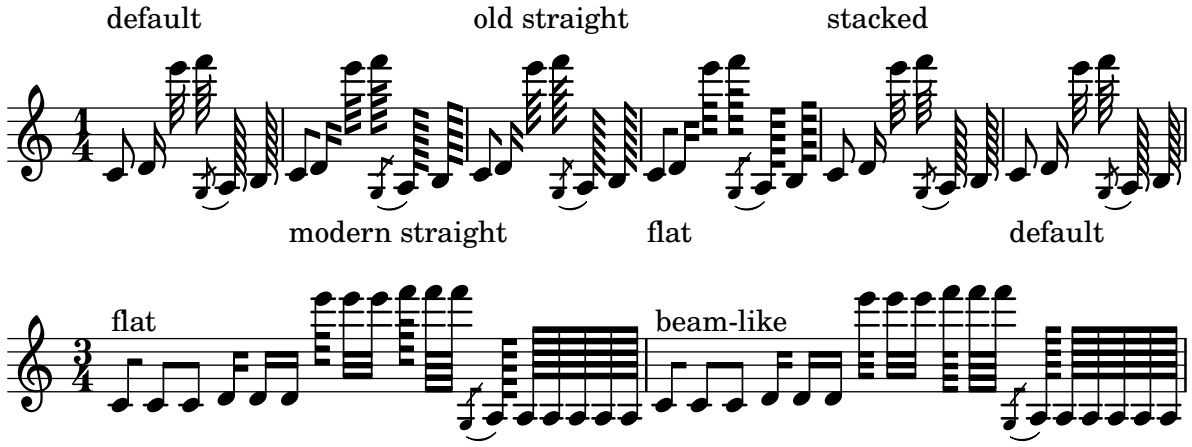
  \flagStyleStacked
    <>^"stacked" \testnotes
```

```
  \flagStyleDefault
    <>_"default" \testnotes
}

\relative c' {
  \time 3/4
  \override Flag.stencil = #flat-flag

  <>^"flat" c8 c[ c] d16 d[ d] e''32 e[ e] f64 f[ f]
    \acciaccatura { g,,,8 } a128 a[ a a a]
  <>^"beam-like" @c8 c[ c] @d16 d[ d] @e''32 e[ e] @f64 f[ f]
    \acciaccatura { g,,,8 } @a128 a[ a a a]
}

\layout {
  indent = 0
  \context {
    \Score
    \override NonMusicalPaperColumn.line-break-permission = ##f
  }
}
```



Voir aussi

Manuel de notation : Section 36.1 [Direction et positionnement], page 801, Section 2.6.1 [Notes d'ornement], page 150.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Beam” dans *Référence des propriétés internes*, Section “BeamEvent” dans *Référence des propriétés internes*, Section “Beam_engraver” dans *Référence des propriétés internes*, Section “beam-interface” dans *Référence des propriétés internes*, Section “Stem_engraver” dans *Référence des propriétés internes*.

2.4.4 Liens de croches en soufflet

Les ligatures en soufflet permettent d'indiquer qu'un petit groupe de notes se joue en accélérant ou en ralentissant sans pour autant modifier le tempo du morceau. L'étendue du soufflet s'indique par `[` et `]` ; son orientation est déterminée par la propriété `grow-direction` de l'objet `Beam`.
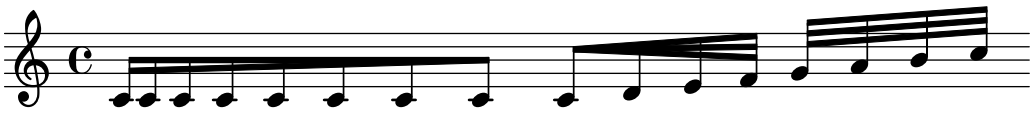
Lorsque la sortie MIDI doit refléter les *ritardando* ou *accelerando* indiqués par une ligature en soufflet, les notes qui la composent doivent être regroupées dans une expression musicale

délimitée par des accolades, précédée de la commande \featherDurations. Cette commande détermine le ratio entre les durées des premières et dernières notes du groupe en question.

Les crochets indiquent l'étendue de la ligature et les accolades les notes concernées par une modification de leur durée. Il s'agit en général du même groupe de notes, mais les deux commandes sont indépendantes l'une de l'autre.

Dans l'exemple ci-après, les huit doubles croches occupent exactement le même espace qu'une blanche, mais la première est moitié moins longue que la dernière et celles qui les séparent s'allongent peu à peu. Les quatre triples croches qui suivent vont s'accélérant, alors que les quatre dernières gardent un tempo régulier.

```
\relative c' {
  \override Beam.grow-direction = #LEFT
  \featherDurations 2/1
  { c16[ c c c c c c c] }
  \override Beam.grow-direction = #RIGHT
  \featherDurations 2/3
  { c32[ d e f] }
  % revert to non-feathered beams
  \override Beam.grow-direction = #'()
  { g32[ a b c] }
}
```



Si le résultat imprimable ne reflète les durées que de manière approximative, la sortie MIDI sera quant à elle parfaitement « ponctuelle ».

Commandes prédéfinies

\featherDurations.

Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Problèmes connus et avertissements

La commande \featherDurations ne permet de traiter que de très courts extraits, avec une faible amplitude.
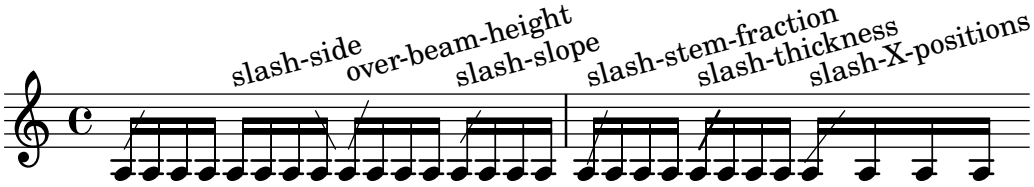
2.4.5 Ligatures barrées

Les ligatures peuvent apparaître barrées à l'aide de la procédure de stencil spécifique beam::slashed-stencil. La barre peut s'imprimer sur la gauche ou sur la droite de la ligature, et peut s'ajuster grâce aux sous-propriétés de details over-beam-height, slash-slope, slash-side, slash-stem-fraction, slash-thickness et slash-X-positions. Il est à noter que ces sous-propriétés s'accommodent entre elles afin d'obtenir un meilleur rendu ; autrement dit, modifier l'une peut avoir un impact sur les autres.

```
mus = \*4 a16

{
  \override TextScript.rotation = #'(15 1 0)
  \override Beam.stencil = #beam::slashed-stencil
  \mus
```

```
  <>^"slash-side" %% par défaut: LEFT
  \once \override Beam.details.slash-side = #RIGHT
  \mus
  <>^"over-beam-height" %% par défaut: 0.75
  \once \override Beam.details.over-beam-height = #1.5
  \mus
  <>^"slash-slope" %% par défaut: 2
  \once \override Beam.details.slash-slope = #1.0
  \mus
  <>^"slash-stem-fraction" %% par défaut: 0.3
  \once \override Beam.details.slash-stem-fraction = #0.6
  \mus
  <>^"slash-thickness" %% par défaut: 0.1
  \once \override Beam.details.slash-thickness = #0.2
  \mus
  <>^"slash-X-positions" %% par défaut: (-0.5 . 1)
  \once \override Beam.details.slash-X-positions = #'(-1 . 2)
  \mus
}
```
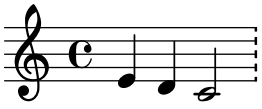


2.5 Mesures

2.5.1 Barres de mesure

Les barres de mesure délimitent les mesures ou sections, mais peuvent aussi indiquer une reprise. En principe, de simples barres sont insérées automatiquement en respectant la métrique en vigueur. Diverses commandes permettent l'insertion automatique de différents types de barre selon l'effet attendu – voir Section 2.5.2 [Barres de mesure automatiques], page 133.

Il est possible de forcer l'impression d'une barre de mesure spécifique à l'aide de commande `\bar` :

```
\relative { e'4 d c2 \bar "!" }
```



Rien ne s'oppose à ce que la dernière note d'une mesure ne s'arrête avant la barre de mesure ; on considère simplement qu'elle se prolonge sur la mesure suivante. Des débordements à répétition finissent par générer une musique comprimée ou qui sort de la page, pour la simple et bonne raison que les sauts de ligne automatiques ne peuvent intervenir qu'à la fin d'une mesure complète, autrement dit lorsque toutes les notes sont terminées avant la fin de la mesure.

> **Note :** Une durée erronée peut empêcher les sauts de ligne, ce qui conduit à une musique compressée, voire à un débordement de la page.
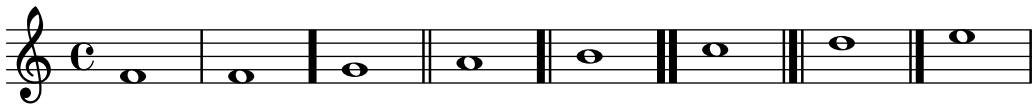
Les sauts de ligne sont aussi possibles lorsqu'est insérée manuellement une barre de mesure, même si cette mesure est incomplète. L'utilisation de la commande `\allowBreak` permet de passer à la ligne suivante sans pour autant imprimer de barre de mesure – voir Section 28.1 [Sauts de ligne], page 711.

Cette barre invisible, ainsi que d'autres barres spéciales, peuvent être insérées manuellement n'importe où. Lorsqu'elles coïncident avec la fin d'une mesure, elles remplacent la simple barre que LilyPond aurait insérée automatiquement. Dans le cas contraire, la barre spécifiée s'insérera là où vous l'aurez positionnée.

Ces insertions n'affectent en rien le calcul du positionnement automatique des barres de mesure à suivre ni les propriétés y afférentes – numérotation, altérations accidentelles, sauts de ligne. . .
Lorsqu'une barre manuelle est insérée à l'endroit où viendrait se placer une barre normale, seul l'effet visuel en sera modifié.

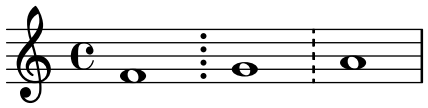
Vous disposez de plusieurs types de barres simples et de doubles barres :

```
\relative {
  f'1 \bar "|"
  f1 \bar "."
  g1 \bar "||"     % voir \section
  a1 \bar ".|"
  b1 \bar ".."
  c1 \bar "|.|"
  d1 \bar "|."     % voir \fine
  e1
}
```



ainsi que d'une barre en pointillé et d'une discontinue :

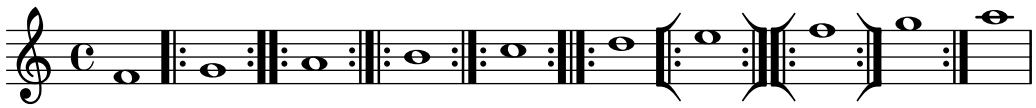
```
\relative {
  f'1 \bar ";"
  g1 \bar "!"
  a1
}
```



et de différents types de barre de reprise :

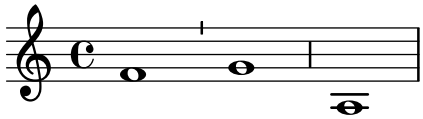
```
\relative {
  f'1 \bar ".|:"
  g1 \bar ":..:"
  a1 \bar ":|.|:"
  b1 \bar ":|.:"
  c1 \bar ":.|.:"
  d1 \bar "[|:"
  e1 \bar ":|][|:"
  f1 \bar ":|]"
  g1 \bar ":|."
  a1
```

```
}
```



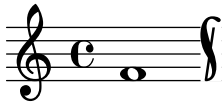
De plus, une barre de mesure peut s'imprimer raccourcie ou sous la forme d'une coche. Dans le cadre du chant grégorien, nous vous invitons à plutôt utiliser `\divisioMinima` et `\divisioMaior` comme indiqué au paragraphe Section 17.4.4 [Divisions], page 571.

```
f'1 \bar "'"
g'1 \bar ","
a1
```



LilyPond prend en charge la notation kiévienne, qui dispose d'une barre de mesure spécifique :

```
f'1 \bar "k"
```
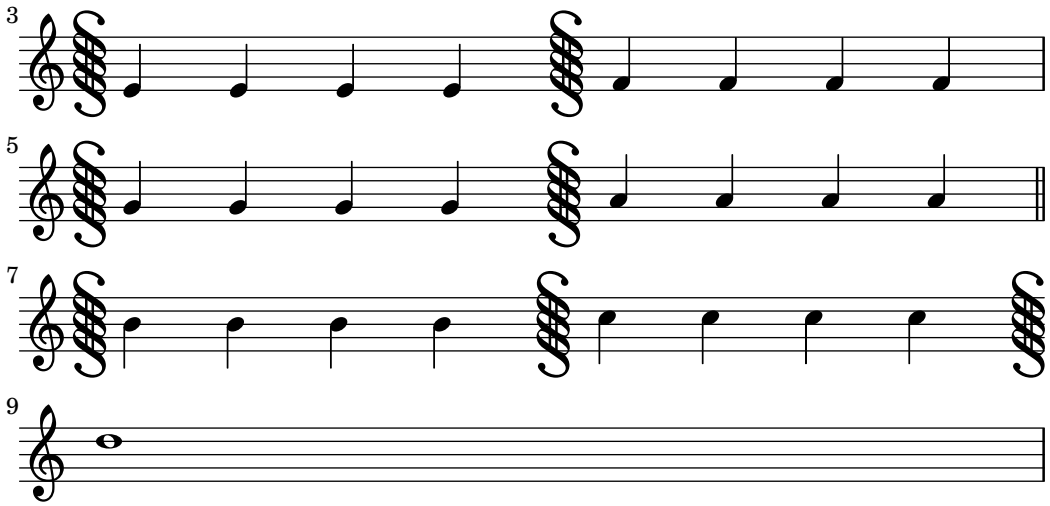


De plus amples détails quant à cette forme de notation sont fournis à la rubrique Section 17.5 [Typographie de notation kiévienne], page 579.

L'insertion d'un *segno* directement sur la portée s'obtient à l'aide de plusieurs types de barre de mesure, dont les effets diffèrent lorsqu'en fin de ligne :

```
\fixed c' {
  c4 4 4 4
  \bar "S"
  d4 4 4 4 \break
  \bar "S"
  e4 4 4 4
  \bar "S-|"
  f4 4 4 4 \break
  \bar "S-|"
  g4 4 4 4
  \bar "S-||"
  a4 4 4 4 \break
  \bar "S-||"
  b4 4 4 4
  \bar "S-S"
  c'4 4 4 4 \break
  \bar "S-S"
  d'1
}
```

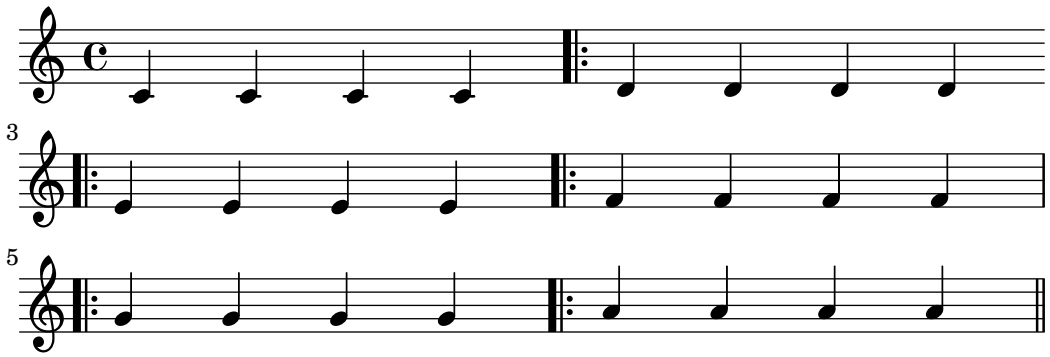


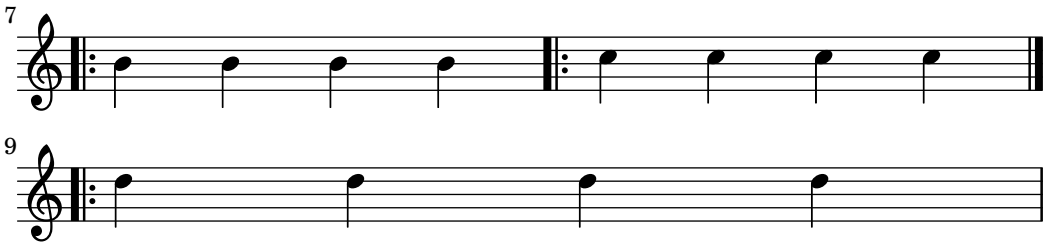


Bien que l'on puisse insérer des barres de reprise manuellement, LilyPond n'en déduira pas pour autant qu'il s'agit d'un passage à répéter. Il est préférable d'indiquer les passages répétés à l'aide des différentes commandes de reprise (voir Chapitre 4 [Répétitions et reprises], page 193) qui se chargeront d'imprimer le type de barre approprié qui peut d'ailleurs être personnalisé – voir Section 2.5.2 [Barres de mesure automatiques], page 133.

Dans les faits, un `".|:-||"` équivaut à un `".|:"` sauf s'il intervient à un saut de ligne : une double barre sera alors imprimée en fin de portée, et la barre de reprise au début de la nouvelle.

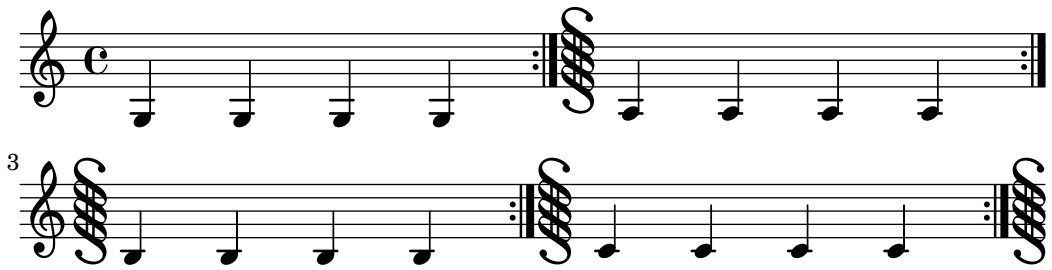
```
\fixed c' {
  c4 4 4 4
  \bar ".|:"
  d4 4 4 4 \break
  \bar ".|:"
  e4 4 4 4
  \bar ".|:-|"
  f4 4 4 4 \break
  \bar ".|:-|"
  g4 4 4 4
  \bar ".|:-||"
  a4 4 4 4 \break
  \bar ".|:-||"
  b4 4 4 4
  \bar ".|:-|."
  c'4 4 4 4 \break
  \bar ".|:-|."
  d'4 4 4 4
}
```
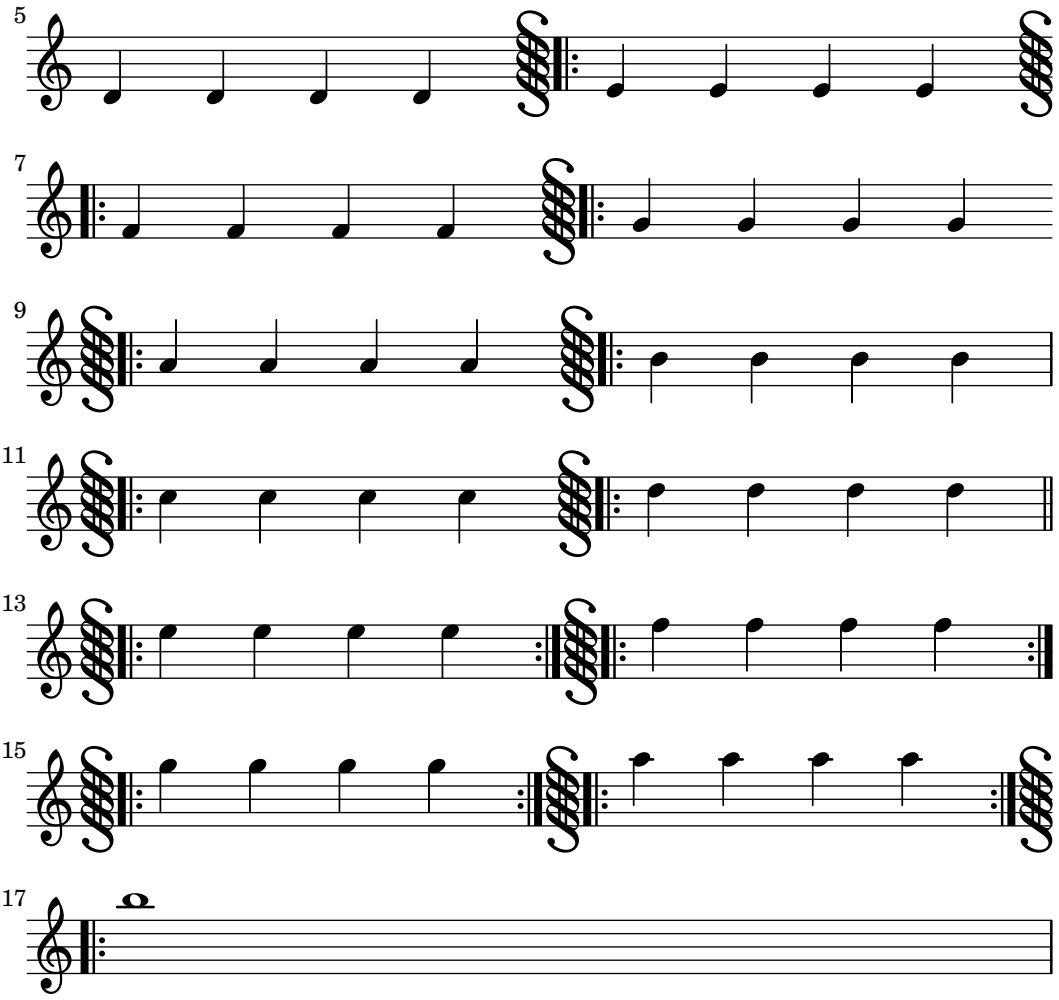




LilyPond dispose de différents moyens de combiner une barre de reprise avec un *segno* :

```
\fixed c' {
  g,4 4 4 4
  \bar ":|.S"
  a,4 4 4 4 \break
  \bar ":|.S"
  b,4 4 4 4
  \bar ":|.S-S"
  c4 4 4 4 \break
  \bar ":|.S-S"
  d4 4 4 4
  \bar "S.|:-S"
  e4 4 4 4 \break
  \bar "S.|:-S"
  f4 4 4 4
  \bar "S.|:"
  g4 4 4 4 \break
  \bar "S.|:"
  a4 4 4 4
  \bar "S.|:-|"
  b4 4 4 4 \break
  \bar "S.|:-|"
  c'4 4 4 4
  \bar "S.|:-||"
  d'4 4 4 4 \break
  \bar "S.|:-||"
  e'4 4 4 4
  \bar ":|.S.|:"
  f'4 4 4 4 \break
  \bar ":|.S.|:"
  g'4 4 4 4
  \bar ":|.S.|:-S"
  a'4 4 4 4 \break
  \bar ":|.S.|:-S"
  b'1
}
```





Nombre de ces indications peuvent s'insérer automatiquement à l'aide des commandes `\repeat` – voir Chapitre 4 [Répétitions et reprises], page 193.

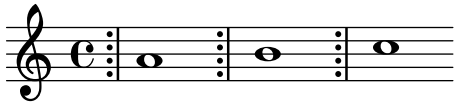
L'instruction `\defineBarLine` permet de définir ses propres types de barre de mesure, en respectant la syntaxe :

```
\defineBarLine type_de_barre #'(fin début extension)
```

En résumé, l'argument *type_de_barre* détermine à la fois le glyphe qui sera utilisé en cours de portée et l'identifiant de cette barre de mesure. Les autres arguments déterminent les glyphes à utiliser aux extrémités de la portée et dans l'espace entres les portées d'un système. Le fait de régler *fin*, *début* ou *extension* à `#t` revient à utiliser le glyphe de *type_de_barre* à la position correspondante. Un réglage à `#f` se traduira par l'absence de barre à cette même position.

Dans le détail, l'argument *type_de_barre* est une chaîne de caractères qui a une double fonction : il détermine d'une part le glyphe qui sera imprimé lorsqu'une barre de mesure intervient en cours de portée et, d'autre part, il sert d'identifiant pour l'objet barre de mesure appelé par `\bar type_de_barre`. Il doit être de la forme *dessin* ou *dessin-annotation* (avec un trait d'union littéral), où *annotation* est une chaîne quelconque et *dessin* une chaîne dont les caractères sont les noms de l'un des éléments de barre prédéfinis listés ci-après. La représentation qui résulte de la concaténation de ces éléments sera utilisée en cours de portée. Par exemple, un *type_de_barre* égal à `";|"` ou `";|-autre"` spécifie une barre composite constituée d'une ligne pointillée (';') accouplée à une ligne pleine ('|') :

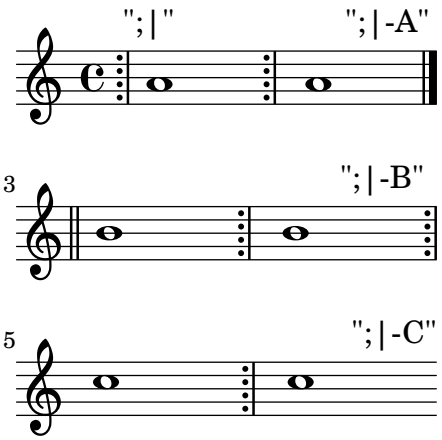
```
\defineBarLine ";|" #'(#t #t #t)
\defineBarLine ";|-autre" #'(#f #f #f)
\fixed c' {
   \bar ";|" a1 \bar ";|" b1 \bar ";|-autre" c'1 \bar ";|-autre"
}
```

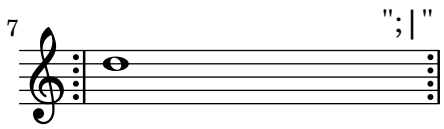


L'*annotation* (*autre* dans le second exemple ci-dessus) permet de distinguer ce type de barre des autres ayant le même *dessin* mais adoptant des comportements différents à l'occasion d'un saut de ligne ou en cas de portées multiples. Par convention, la chaîne spécifiée pour *fin* est souvent reprise comme annotation, aussi aurions-nous pu avoir nommé le second exemple '`";|-f"`'.

Les arguments *fin* et *début* spécifient la barre à afficher à la fin d'une ligne et au début de la suivante lorsque `\bar` *type_de_barre* intervient sur un saut de ligne. *début* s'appliquera aussi lorsque `\bar` *type_de_barre* est placé au commencement de la pièce. Le format de ces arguments est identique à *type_de_barre*. La chaîne d'éléments de barre détermine le glyphe qui apparaîtra selon son positionnement dans la ligne. De plus, chacun de ces éléments peut prendre la valeur `#t` pour reproduire la valeur de *type_de_barre*, ou `#f` pour ne rien afficher. Dans l'exemple suivant, tous les types de barre impriment un trait pointillé suivi d'un trait plein lorsqu'en cours de ligne, mais le comportement change en début ou en fin de ligne.

```
%   pointillé-plein partout
\defineBarLine  ";|"  #'( #t #t #t)
%   plein-gras en fin, plein-plein en début
\defineBarLine  ";|-A" #'( "|."  "||"  #f )
%   pointillé-plein en fin, rien en début
\defineBarLine  ";|-B" #'( #t  #f  #f )
%   rien en fin, pointillé-plein en début
\defineBarLine  ";|-C" #'( #f  #t  #f )
\relative c'' {
  \bar ";|" \textMark "\";|\""
  a1  \bar ";|-A"
  a1  \bar ";|-A" \textEndMark "\";|-A\"" \break
  b1  \bar ";|-B" b \bar ";|-B" \textEndMark "\";|-B\"" \break
  c1  \bar ";|-C" c \bar ";|-C" \textEndMark "\";|-C\"" \break
  d1  \bar ";|" \textEndMark "\";|\""
}
```

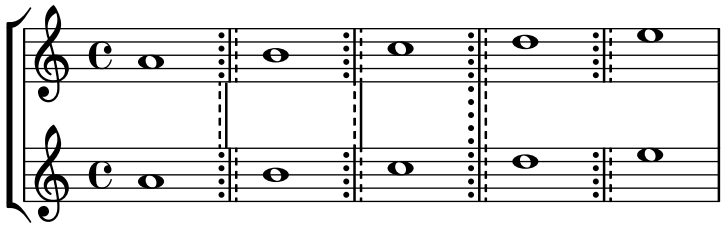




Il est à noter que les chaînes de *début* ou de *fin* peuvent être des noms de type de barre précédemment définis. Dans le cadre d'un contexte de portée unique, ceci n'a pas d'importance puisque les éléments de barre dans la chaîne spécifiée seront utilisés sans les personnalisations de la barre à laquelle il est fait référence par cette chaîne. Il est néanmoins important, dans le cadre d'un système à plusieurs portées, que *fin* et *début* fassent référence à des types de barre déjà définis (y compris *type_de_barre*) ou soient réglés à `#t` ou `#f`.

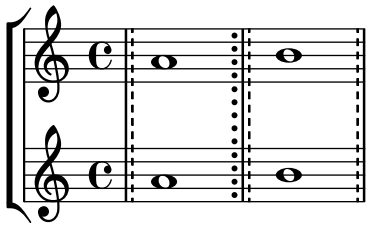
L'argument *extension* sera effectif dans les systèmes à plusieurs portées (voir Section 6.1.2 [Regroupement de portées], page 250) puisqu'il détermine ce qui apparaîtra entre les portées regroupées. Cet argument est composé d'une chaîne d'éléments de barre, de même longueur que *type_de_barre* ou plus court – tout élément supplémentaire sera ignoré. Chaque élément sera imprimé dans le prolongement de l'élément correspondant de *type_de_barre*. Le caractère espace `" "` permet de préserver de l'espace et ainsi aligner correctement les différents tronçons d'une barre d'un seul tenant entre les portées d'un système. Régler *extension* à `#t` revient à sélectionner le glyphe utilisé en cours de ligne. Le régler à `#f` omet la barre interportées et lui affecter `""` (une chaîne vide) générera une extension de largeur zéro. Voici quelques exemples :

```
\defineBarLine  ";|!-A"  #'(#t #t "!|")
\defineBarLine  ";|!-B"  #'(#t #t " !|")
\defineBarLine  ";|!-C"  #'(#t #t #t)
\defineBarLine  ";|!-D"  #'(#t #t #f)
\fixed c' {
  \new StaffGroup <<
    \new Staff {
      a1 \bar ";|!-A"
      b  \bar ";|!-B"
      c' \bar ";|!-C"
      d' \bar ";|!-D"
      e'
    }
    \new Staff {
      a1 b c' d' e'
    }
  >>
}
```



Lorsqu'intervient un saut de ligne seront utilisés, en lieu et place de *extension*, *fin* et *début* pour déterminer la barre interportées à afficher. Dans le cas d'un regroupement de portées, LilyPond émettra un avertissement si *fin* et *début* ne font pas référence à des types de barre définis, et aucune prolongation ne sera imprimée. Il est toujours possible d'affecter `#f` à *fin* et *début*, auquel cas aucune prolongation n'apparaîtra à la position correspondante. Ces arguments peuvent aussi être affectés d'un `#t` ou *type_de_barre*, auquel cas *extension* déterminera le type de prolongation à utiliser à l'endroit correspondant.

```
\defineBarLine  "!|-t"  #'(#t #t #t)
\defineBarLine  "|!-t"  #'(#t #t #t)
\defineBarLine  ";|!-bad"  #'("|!" "!|" #t)  % échec en fin de ligne
\defineBarLine  ";|!-good" #'("!|-t" "|!-t" #t)
\relative c'' {
  \new StaffGroup <<
    \new Staff {
      \bar ";|!-good"
      a1 \bar ";|!-good"
      % \bar ";|!-bad"     % "WARNING: No span bar glyph defined..."
      b1 \bar ";|!-good"
    }
    \new Staff {
      a1 b1
    }
  >>
}
```



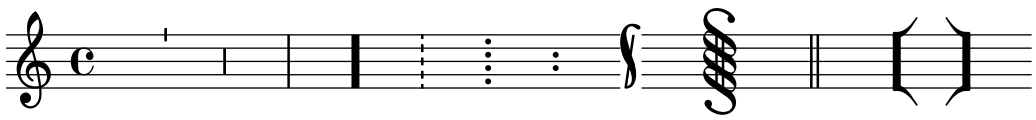
Notez bien que dans le cas où *extension* est constitué d'une chaîne, celle-ci devrait comporter **uniquement** des éléments de barre de mesure ou une espace (‘ ’), et sans annotation comme c'est possible pour les autres arguments – LilyPond émettra un avertissement lorsque *extension* contient une annotation. La seule exception à cette règle est une valeur égale à *type_de_barre* qui, lui, peut comporter une annotation.

Les différents éléments de barre de mesure sont indiqués ci-dessous. La plupart dispose par ailleurs de types de barre prédéfinis (utilisables en argument à `\bar`) qui les référencent individuellement. Certains éléments sont premièrement destinés à être combinés à d'autres et, de ce fait, n'ont pas de type de barre prédéfinis individuellement.

```
\defineBarLine ":" #'(#f #t #f)
\defineBarLine "=" #'(#t #f #t)
\defineBarLine "[" #'(#f #t #f)
\defineBarLine "]" #'(#t #f #f)

\new Staff {
  s1 \bar "'"
  s1 \bar ","
  s1 \bar "|"
  s1 \bar "."
  s1 \bar "!"
  s1 \bar ";"
  s1 \bar ":"
  s1 \bar "k"
  s1 \bar "S"
  s1 \bar "="
  s1 \bar "["
  s1 \bar "]"
```

```
  s1 \bar ""
}
```
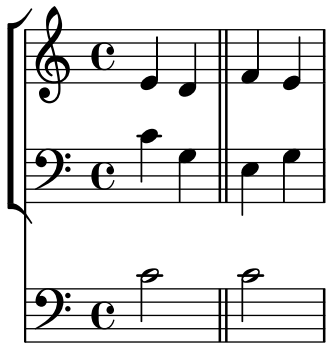


Le type "=" fournit un double trait destiné à être utilisé en combinaison avec un *segno*. Nous vous recommandons de lui préférer \bar "||" pour imprimer une simple double barre fine.

Si d'autres éléments étaient nécessaires, LilyPond dispose de moyens aisés pour les définir. Pour de plus amples informations quant à la manière de modifier ou ajouter des barres de mesure, consultez le fichier scm/bar-line.scm.

Dans une partition comprenant plusieurs portées, la commande \bar placée sur une portée s'applique automatiquement à toutes les portées. Les barres de mesure que l'on obtient alors sont d'un seul tenant sur les portées d'un StaffGroup, d'un PianoStaff ou d'un GrandStaff.

```
<<
  \new StaffGroup <<
    \new Staff \relative {
      e'4 d
      \bar "||"
      f4 e
    }
    \new Staff \relative { \clef bass c'4 g e g }
  >>
  \new Staff \relative { \clef bass c'2 c2 }
>>
```



Le type de barre de mesure utilisé pour l'insertion automatique de barres de mesure est "|". Vous pouvez en changer à tout moment grâce à '\set Timing .measureBarType = *type_de_barre*'.

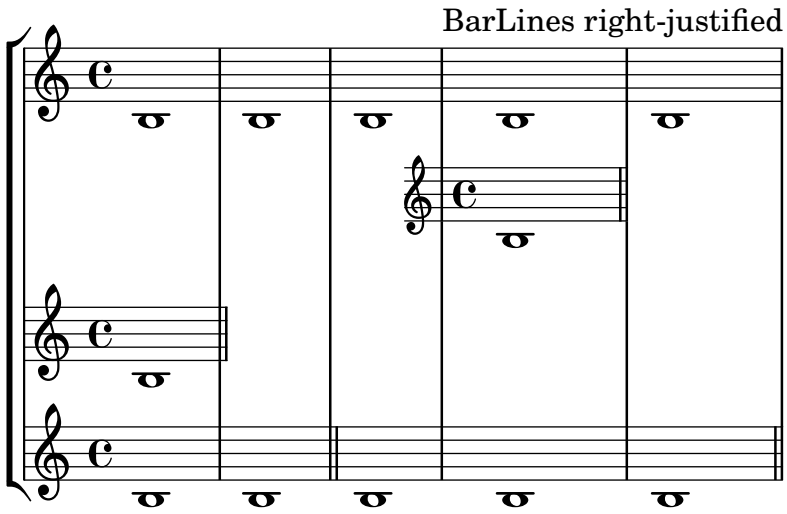
Il est aussi possible de définir plusieurs types de barre de mesure dans une partition à portées multiples, à partir des commandes ou propriétés mentionnées ci-après (voir Section 2.5.2 [Barres de mesure automatiques], page 133). Bien entendu, ceci peut amener à des incohérences entre portion de barre inscrite dans la portée et portion interportées en raison de leurs largeurs différentes.

Les barres de mesure sont généralement alignées par la gauche, sans tenir compte des doubles points des barres de reprise. Pour les aligner par la droite, l'instruction

```
[\once] \override Contexte.BarLine.right-justified = ##t
```

doit s'appliquer, avec *Contexte* correspondant à un contexte spécifique aux portées multiples comme*Score*, *StaffGroup*, *Grandstaff*, etc.

```
\new StaffGroup
  <<
    \new Staff = "a" {
      b1  b b
      <<
        { \textMark "BarLines right-justified" b b }
        \new Staff \with { alignAboveContext = "b" }
        {
          \override StaffGroup.BarLine.right-justified = ##t
          b
          \section
        }
      >>
    }
    \new Staff = "b" { b \section }
    \new Staff = "c" { b b \section b b b \section }
  >>
```



Les barres de mesures ne sont jamais alignées par la droite à la suite d'un saut de ligne. En ce qui concerne les barres en cours de portée et celles alignées par la droite, le point d'ancrage – utilisé pour aligner les `BarNumber`, `RehearsalMark`, etc. – est repositionné en conséquence.

Voir aussi

Manuel de notation : Section 6.1.2 [Regroupement de portées], page 250, Chapitre 4 [Répétitions et reprises], page 193, Section 28.1 [Sauts de ligne], page 711.

Fichiers d'initialisation : `scm/bar-line.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

Référence des propriétés internes : Section "BarLine" dans *Référence des propriétés internes* (faisant partie du contexte `Staff`), Section "SpanBar" dans *Référence des propriétés internes* (sur plusieurs portées), Section "Timing_translator" dans *Référence des propriétés internes* (pour les propriétés liées au temps).

2.5.2 Barres de mesure automatiques

Diverses commandes, autres que `\bar`, peuvent créer des barres de mesure en raison de leurs effets. Les barres générées en pareil cas peuvent se modifier en réglant des propriétés de contexte. Lorsqu'une des propriétés est réglée sur `'()` ou n'est pas définie, elle est ignorée ; la valeur doit

être l'un des types prédéfinis ou bien l'un de ceux précédemment définis à l'aide de la commande `\defineBarLine` – voir Section 2.5.1 [Barres de mesure], page 123.

De multiples raisons peuvent expliquer la création de barres de mesure automatiques différentes en un même moment. Les conflits se résoudront en partie par la fourniture de propriétés pour des combinaisons prédéterminées et en partie par un jeu de priorités. La table suivante présente, par ordre de priorité croissant, les propriétés disponibles.

`underlyingRepeatBarType`
: S'utilise en des points de reprise ou de départ pour lesquels aucune barre n'apparaîtrait normalement. Tel est le cas lorsque des sections répétées ne s'alignent pas avec les mesures. Certaines commandes emploient ce type de barre : `\codaMark`, `\inStaffSegno`, `\repeat segno`, `\repeat volta` et `\segnoMark`.

`caesuraType underlying-bar-line`
: S'utilise avec `\caesura` – voir Section 9.7.4 [Barre de phrase en musique liturgique], page 421.

`submeasureBarType`
: S'utilise pour borner les subdivisions de mesure spécifiées par `submeasureStructure`, lorsque l'autorise un `\submeasureBarsOn` ; voir Section 2.3.2 [Métrique], page 81.

`measureBarType`
: S'utilise en limite de mesure.

`caesuraType bar-line`
: S'utilise avec `\caesura` – voir Section 9.7.4 [Barre de phrase en musique liturgique], page 421.

`sectionBarType`
: S'utilise à un saut de section créé par `\section`.

`fineBarType`
: S'utilise avec l'instruction `\fine`.

`doubleRepeatBarType`
`doubleRepeatSegnoBarType`
`endRepeatBarType`
`endRepeatSegnoBarType`
`fineSegnoBarType`
`fineStartRepeatSegnoBarType`
`segnoBarType`
`startRepeatBarType`
`startRepeatSegnoBarType`
: Seul l'un de ces types s'utilisera, selon la structure de la pièce.

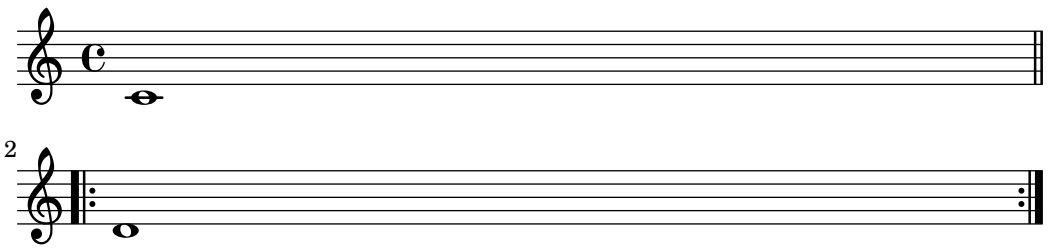
  Les propriétés dont le nom comporte `startRepeat` ou `endRepeat` s'utilisent en début ou en fin de reprise créée par `\repeat volta`. Celles comportant `doubleRepeat` s'utilisent lorsque la fin d'une reprise coïncide avec le début d'une autre.

  Les propriétés dont le nom comporte `segno` s'utilisent pour un *segno* sur la portée ; ceux-ci peuvent être créés par `\repeat segno` ou `\segnoMark`, dès lors que la propriété `segnoStyle` est réglée sur `bar-line`, ou encore lorsqu'ils sont créés par `\inStaffSegno`.

  Les propriétés dont le nom comporte `fine` répondent à l'instruction `\fine`.

La priorité s'applique indépendamment du fait que la barre soit en début, en cours ou en fin de ligne, ce qui permet à des barres de mesure de priorité inférieure d'apparaître là où des types de barre de priorité supérieure n'auraient pas de glyphe défini – voir Section 2.5.1 [Barres de mesure], page 123.

```
\fixed c' {
  c1 \section \break
  \repeat volta 2 d1
}
```

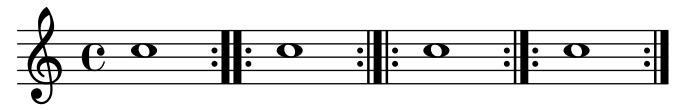


Morceaux choisis

Succession de reprises et style de barre par défaut

LilyPond dispose de trois différents styles de barre pour indiquer une succession de reprises. Vous devez opter pour un style par défaut, à l'aide de la propriété de contexte `doubleRepeatBarType`.

```
\relative c'' {
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":..:"
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":|.|:"
  \repeat volta 2 { c1 }
  \set Score.doubleRepeatBarType = ":|.:"
  \repeat volta 2 { c1 }
}
```



Voir aussi

Manuel de notation : Section 2.5.1 [Barres de mesure], page 123, Chapitre 4 [Répétitions et reprises], page 193.

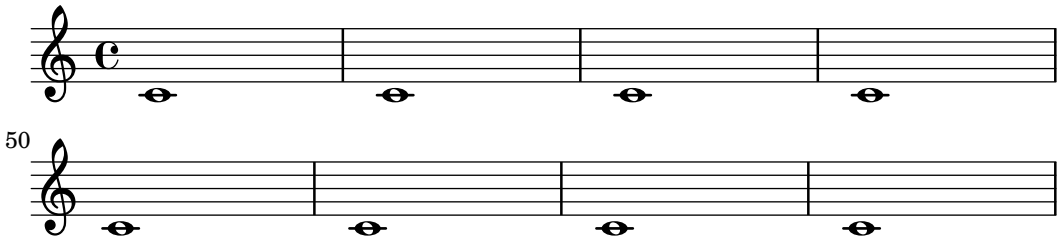
Morceaux choisis : Section "Rythmes" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Repeat_acknowledge_engraver" dans *Référence des propriétés internes*.

2.5.3 Numéros de mesure

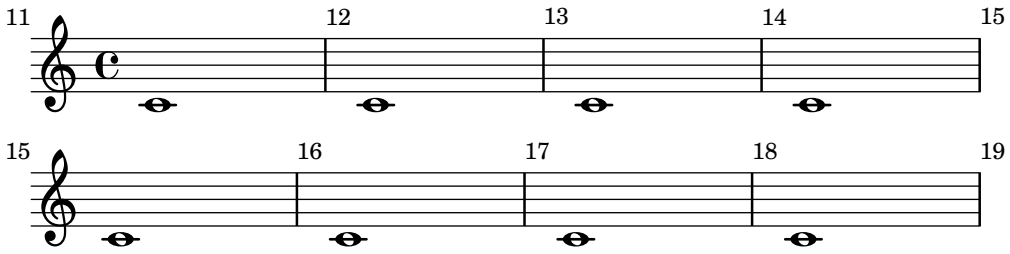
Les numéros de mesure sont imprimés par défaut à chaque début de ligne, sauf la première. Ce nombre est stocké par la propriété `currentBarNumber` qui sera mise à jour à chaque mesure. Vous pouvez aussi le définir de manière arbitraire :

```
\relative c' {
  c1 c c c
  \break
  \set Score.currentBarNumber = 50
  c1 c c c
}
```



Le comportement par défaut consiste à imprimer un numéro de mesure uniquement en tête de chaque ligne. Il est possible d'y déroger à l'aide de la propriété `break-visibility` du `BarNumber`. Elle se compose de trois commutateurs – définis à « vrai » (`#t`) ou « faux » (`#f`) – pour spécifier si le numéro de mesure est visible ou non. Les valeurs sont rangées dans l'ordre suivant : `visible en fin de ligne`, `visible en cours de ligne` et `visible en début de ligne`. Voici comment imprimer partout les numéros de mesure :

```
\relative c' {
  \override Score.BarNumber.break-visibility = ##(#t #t #t)
  \set Score.currentBarNumber = 11
  c1 | c | c | c |
  \break
  c1 | c | c | c |
}
```



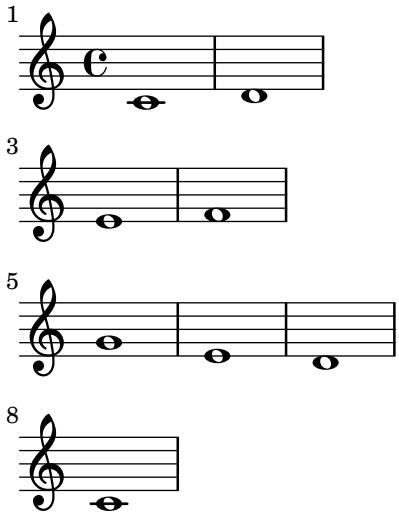
Morceaux choisis

Afficher le numéro de la première mesure

Par défaut, LilyPond n'affiche pas le premier numéro de mesure s'il est inférieur à 2. Le fait de définir `barNumberVisibility` à `all-bar-numbers-visible` permettra d'imprimer n'importe quel numéro pour la première mesure.

```
\paper {
  line-width = 50\mm
}

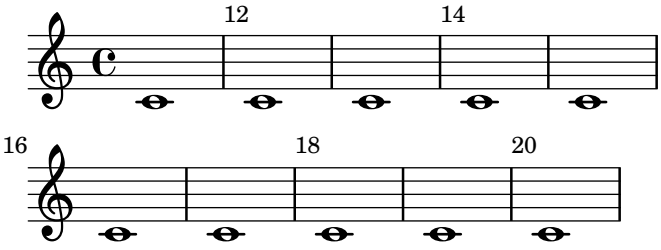
\relative c' {
  \set Score.barNumberVisibility = #all-bar-numbers-visible
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Imprimer les numéros de mesure à intervalle régulier

Vous pouvez imprimer un numéro de mesure à intervalle régulier plutôt qu'en tête de chaque ligne seulement, en recourant à la propriété `barNumberVisibility`. Voici comment afficher le numéro toutes les deux mesures sauf en fin de ligne.
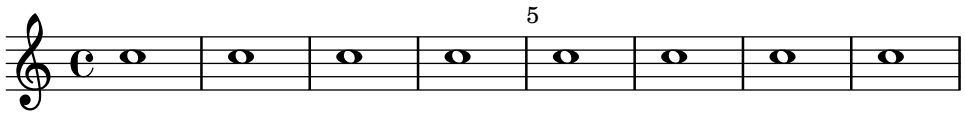
```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \set Score.currentBarNumber = 11
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
  c1 | c | c | c | c
  \break
  c1 | c | c | c | c
}
```

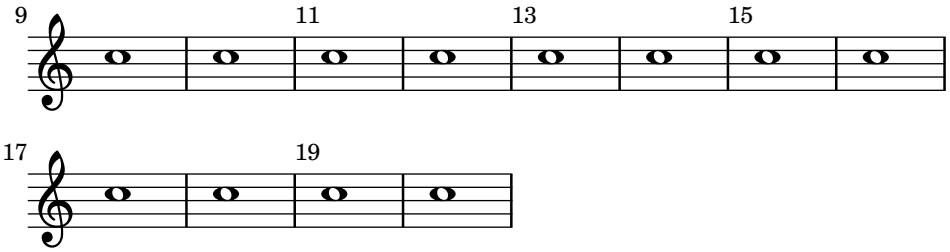


Changement de la fréquence d'impression du numéro de mesure

La fonction de contexte `set-bar-number-visibility` permet de modifier la fréquence à laquelle les numéros de mesures s'impriment.

```
\relative c' {
  \override Score.BarNumber.break-visibility = #end-of-line-invisible
  \context Score \applyContext #(set-bar-number-visibility 4)
  \*10 c'1
  \context Score \applyContext #(set-bar-number-visibility 2)
  \*10 c
}
```



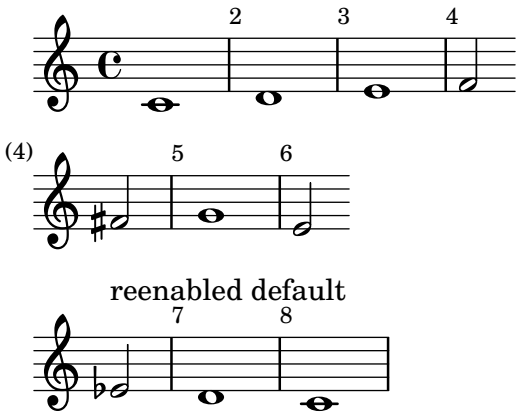


Impression du numéro des mesures tronquées

Le numéro de mesure n'est pas répété en début de ligne pour une mesure tronquée. L'objet `BarNumber` apparaîtra, entre parenthèses, dès lors que la propriété `barNumberVisibility` sera affublée de `first-bar-number-invisible-save-broken-bars`.

```
\layout {
  \context {
    \Score
    barNumberVisibility = #first-bar-number-invisible-save-broken-bars
    \override BarNumber.break-visibility = ##(#f #t #t)
  }
}

\relative c' {
  c1 | d | e | f2 \break
  fis2 | g1 | e2 \break
  <>^"reenabled default"
  % back to default -
  % \unset Score.barNumberVisibility would do so as well
  \set Score.barNumberVisibility =
    #first-bar-number-invisible-and-no-parenthesized-bar-numbers
  es2 | d1 | c
}
```



Impression du numéro de mesure selon `modulo-bar-number-visible`

Lorsque le reste de la division du numéro de la mesure courante par le premier argument de `modulo-bar-number-visible` égale le deuxième argument, LilyPond imprime un objet `BarNumber`.

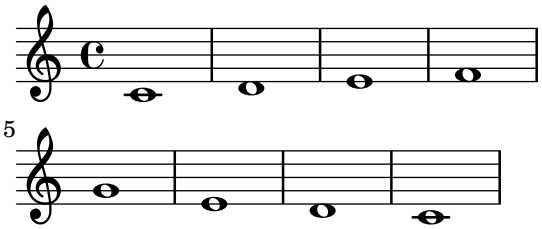
Ceci permet d'imprimer le numéro de mesure à un intervalle donné, par exemple :

- `(modulo-bar-number-visible 3 2)` → affichage à 2, 5, 8...
- `(modulo-bar-number-visible 4 2)` → affichage à 2, 6, 10...

- (modulo-bar-number-visible 3 1) → affichage à 3, 5, 7...
- (modulo-bar-number-visible 5 2) → affichage à 2, 7, 12...

```
\layout {
  \context {
    \Score
    \override BarNumber.break-visibility = ##(#f #t #t)
    barNumberVisibility = #(modulo-bar-number-visible 5 0)
  }
}

\relative c' {
  c1 | d | e | f \break
  g1 | e | d | c
}
```



Inscrire le numéro de mesure dans un cadre ou un cercle

Les numéros de mesure peuvent être encadrés ou entourés d'un cercle.

```
\relative c' {
  % Center bar numbers except at the beginning of a staff.
  \override Score.BarNumber.self-alignment-X =
    #(break-alignment-list CENTER CENTER 0.3)

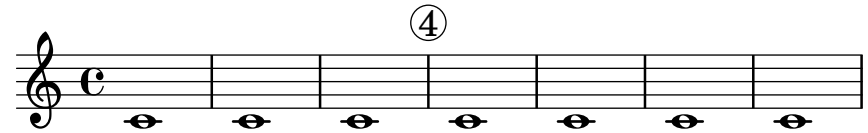
  % Prevent bar numbers at the end of a line and permit them elsewhere.
  \override Score.BarNumber.break-visibility = #end-of-line-invisible

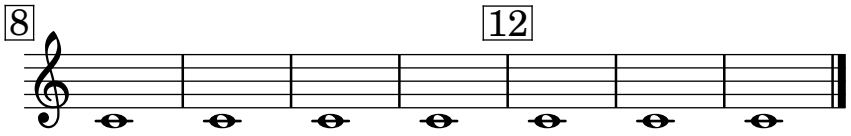
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 4)

  % Increase the size of the bar number by 2.
  \override Score.BarNumber.font-size = 2

  % Draw a circle round the following bar number(s).
  \override Score.BarNumber.stencil
    = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
  \*7 c1 \break

  % Draw a box round the following bar number(s).
  \override Score.BarNumber.stencil
    = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
  \*7 c1 \bar "|."
}
```





Numérotation des mesures et alternatives

Un réglage de la propriété de contexte `alternativeNumberingStyle` permet d'accéder à deux méthodes alternatives de gestion de la numérotation des mesures en cas de reprises.

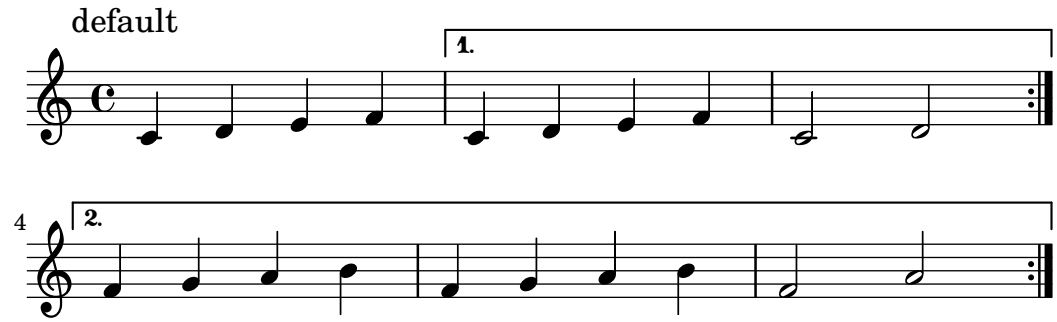
```
music = \relative c' {
  \repeat volta 3 {
    c4 d e f |
    \alternative {
      \volta 1 { c4 d e f | c2 d \break }
      \volta 2 { f4 g a b | f4 g a b | f2 a | \break }
      \volta 3 { c4 d e f | c2 d } } }
  c1 \bar "|."
}
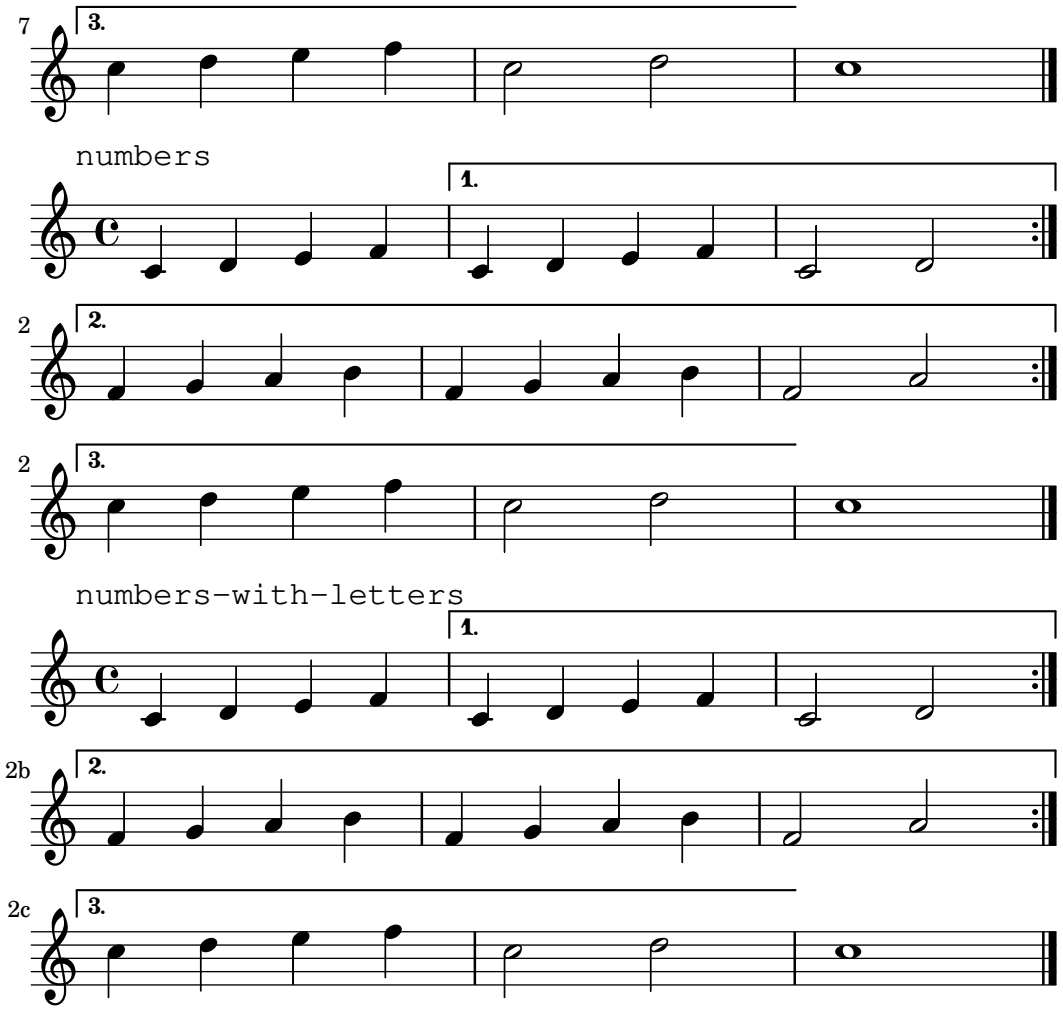
{
  \textMark \markup \large "default"
  \music
}

{
  \textMark \markup \large \typewriter "numbers"
  \set Score.alternativeNumberingStyle = #'numbers
  \music
}

{
  \textMark \markup \large \typewriter "numbers-with-letters"
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \music
}

\layout {
  \context {
    \Score
    \override TextMark.Y-offset = #5
  }
}
```





Alignement des numéros de mesure

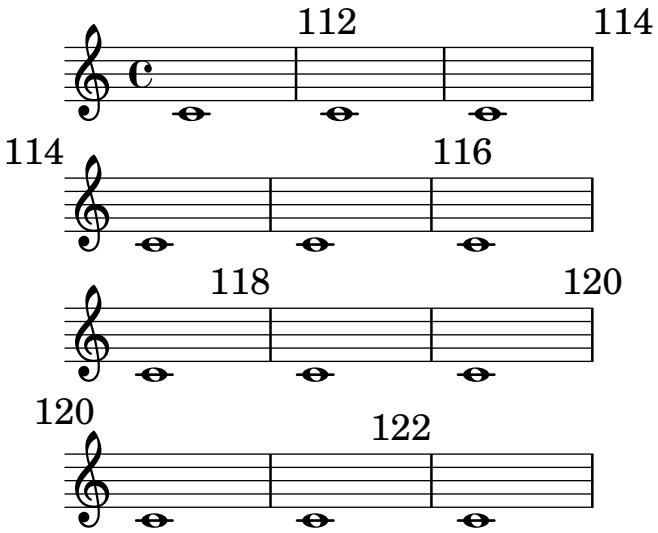
L'alignement des numéros de mesure dépend de l'endroit où ils se produisent : en début de ligne, ils seront alignés par la droite ; ils seront alignés sur la gauche de la barre de mesure dans les autres cas. Ceci peut se modifier à l'aide de la fonction Scheme `break-alignment-list`. Les trois arguments à cette fonction déterminent respectivement l'alignement en fin de ligne, en cours de ligne, et en début de ligne.

```
\relative c' {
  \set Score.currentBarNumber = 111
  \override Score.BarNumber.break-visibility = #all-visible
  % Increase the size of the bar number by 2
  \override Score.BarNumber.font-size = 2
  % Print a bar number every second measure
  \set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)

  c1 | c1 | c1 | \break
  c1 | c1 | c1 | \break

  \override Score.BarNumber.self-alignment-X =
    #(break-alignment-list CENTER RIGHT CENTER)
  c1 | c1 | c1 | \break
  c1 | c1 | c1 |
}
```

```
\paper {
  line-width = 70\mm
}
```

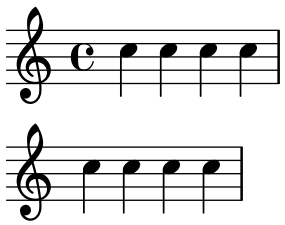


Suppression des numéros de mesure d'une partition

Désactiver le graveur concerné – `Bar_number_engraver` – donnera une partition – contexte `Score` – sans numéros de mesure.

```
\layout {
  \context {
    \Score
    \omit BarNumber
    % or:
    % \remove "Bar_number_engraver"
  }
}

\relative c'' {
  c4 c c c \break
  c4 c c c
}
```



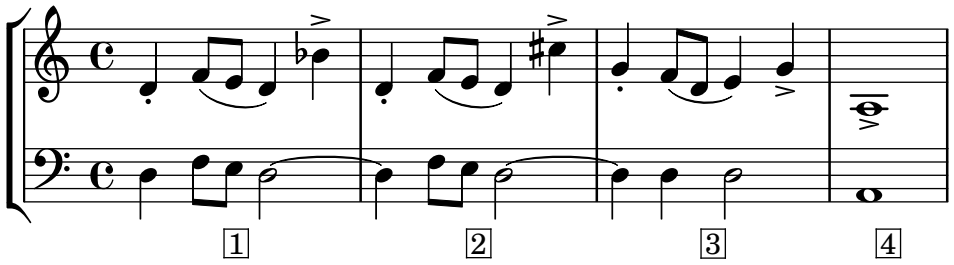
Centrage des numéros de mesure

Il est d'usage, dans les partitions de musique de film, de trouver les numéros de mesure centrés sur leur mesure. Ceci s'obtient en activant la propriété de contexte `centerBarNumbers`. Lorsque cette propriété est utilisée, le type de *grob* (objet graphique) `BarNumber` est remplacé par `CenteredBarNumber` .

L'exemple ci-dessous illustre plusieurs réglages : les numéros de mesure sont à la fois centrés, encadrés, et disposés sous les portées.

```
\layout {
  \context {
    \Score
    centerBarNumbers = ##t
    barNumberVisibility = #all-bar-numbers-visible
    \override CenteredBarNumber.stencil
      = #(make-stencil-boxer 0.1 0.25 ly:text-interface::print)
    \override CenteredBarNumberLineSpanner.direction = #DOWN
  }
}

\new StaffGroup <<
  \new Staff \relative c' {
    d4-. f8( e d4) bes'-> |
    d,-. f8( e d4) cis'-> |
    g-. f8( d e4) g-> |
    a,1-> |
  }
  \new Staff \relative c {
    \clef bass
    d4 f8 e d2~ |
    4 f8 e d2~ |
    4 4 2 |
    a1 |
  }
>>
```



Voir aussi

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Bar_number_engraver” dans *Référence des propriétés internes*, Section “BarNumber” dans *Référence des propriétés internes*, Section “Centered_bar_number_align_engraver” dans *Référence des propriétés internes*, Section “CenteredBarNumber” dans *Référence des propriétés internes*, Section “CenteredBarNumberLineSpanner” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les numéros de mesure peuvent entrer en collision avec les crochets d’un `StaffGroup`. La propriété `padding` – décalage – de l’objet `BarNumber` permet alors d’ajuster leur positionnement. Voir Section “StaffGroup” dans *Référence des propriétés internes* et Section “BarNumber” dans *Référence des propriétés internes* pour plus d’information.

2.5.4 Vérification des limites et numéros de mesure

Les tests de limite de mesure (ou tests de mesure) aident à détecter les erreurs dans les durées. Un test de mesure s'écrit avec une barre verticale `|`. Lors du traitement, elle doit correspondre à une barre de mesure. Sinon, un avertissement est émis qui indique le numéro de ligne où est détectée l'erreur. Dans l'exemple suivant, le deuxième test de mesure signale une erreur.

```
\time 3/4 c2 e4 | g2 |
```

Des durées incorrectes font échouer les tests de mesure et peuvent souvent mettre la partition sens dessus dessous, particulièrement s'il s'agit de musique polyphonique. Vérifier les tests de mesure qui ont échoué et les durées incorrectes est un bon moyen de commencer à corriger sa partition.

Lorsque plusieurs tests successifs présentent un même décalage, seul le message d'avertissement concernant la première occurrence est affiché. L'origine du problème est de fait plus évidente.
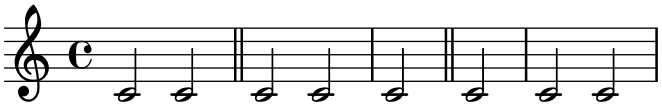
Le test de mesure peut être aussi utilisé dans les paroles, par exemple :

```
\lyricmode {
 \time 2/4
 Twin -- kle | Twin -- kle
}
```

Notez bien qu'en matière de paroles, le test est effectué à l'instant musical où la syllabe **suivant** la marque est traitée. Par voie de conséquence, lorsqu'une mesure débute par un silence, il n'y a pas moyen de positionner une syllabe à cet endroit, et LilyPond émettra un avertissement.

Il est aussi possible d'attribuer une autre fonction au symbole `|`, en assignant une expression musicale à `"|"`. Dans l'exemple suivant, le `|` servira à insérer une double barre là où il apparaît, au lieu de simplement vérifier que la fin de la mesure est atteinte.

```
"|" = \bar "||"
{
  c'2 c' |
  c'2 c'
  c'2 | c'
  c'2 c'
}
```



Lorsque l'on recopie de longues pièces, il peut être utile de vérifier que les numéros de mesure de LilyPond correspondent à l'original que l'on recopie. Cela se fait avec `\barNumberCheck`. Par exemple,

```
\barNumberCheck 123
```

affiche un avertissement lors du traitement si le numéro de mesure à ce point (variable `currentBarNumber`) n'est pas égal à 123.

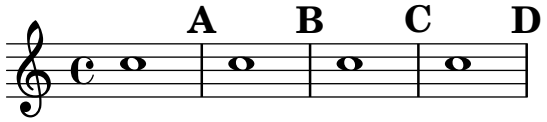
Voir aussi

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.

2.5.5 Indications de repère

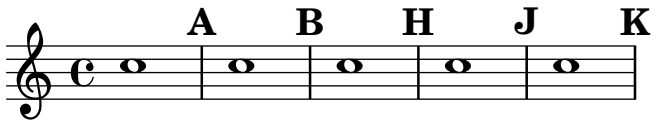
Indiquer un repère s'obtient grâce à la commande `\mark`.

```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
  c1 \mark \default
}
```



Lorsque vous utilisez `\mark \default`, le repère s'incrémente automatiquement. Toutefois, donner un nombre en argument permet de spécifier manuellement le repère en question. La valeur à utiliser est enregistrée dans la propriété `rehearsalMark`.

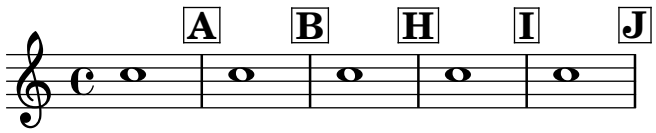
```
\relative c'' {
  c1 \mark \default
  c1 \mark \default
  c1 \mark 8
  c1 \mark \default
  c1 \mark \default
}
```



La lettre **I** n'est pas utilisée, conformément aux usages de la gravure. Cependant, vous pourrez intégrer la lettre **I** en utilisant l'une des commandes suivantes selon que ce repère doive être simple, inclus dans un rectangle ou dans un cercle :

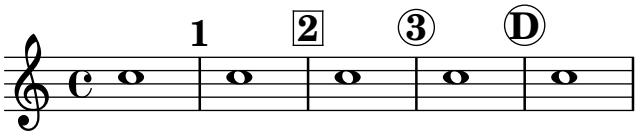
```
\set Score.rehearsalMarkFormatter = #format-mark-alphabet
\set Score.rehearsalMarkFormatter = #format-mark-box-alphabet
\set Score.rehearsalMarkFormatter = #format-mark-circle-alphabet
```

```
\relative c'' {
  \set Score.rehearsalMarkFormatter = #format-mark-box-alphabet
  c1 \mark \default
  c1 \mark \default
  c1 \mark 8
  c1 \mark \default
  c1 \mark \default
}
```



Le style de repère est déterminé par la propriété `rehearsalMarkFormatter`. Il s'agit d'une fonction qui prend en arguments le repère en cours (un entier) ainsi que le contexte en cours, et retournera un objet de type *markup*. Dans l'exemple qui suit, `rehearsalMarkFormatter` est réglé pour une procédure type. Quelques mesures plus loin, son comportement est modifié pour imprimer un repère encadré.

```
\relative c'' {
  \set Score.rehearsalMarkFormatter = #format-mark-numbers
  c1 \mark \default
  c1 \mark \default
  \set Score.rehearsalMarkFormatter = #format-mark-box-numbers
  c1 \mark \default
  \set Score.rehearsalMarkFormatter = #format-mark-circle-numbers
  c1 \mark \default
  \set Score.rehearsalMarkFormatter = #format-mark-circle-letters
  c1
}
```



Le fichier scm/translation-functions.scm comporte les définitions de format-mark-letters (comportement par défaut), format-mark-box-letters, format-mark-numbers et format-mark-box-numbers. Vous pouvez vous en inspirer pour d'autres fonctions de formatage.

format-mark-barnumbers, format-mark-box-barnumbers et format-mark-circle-barnumbers permettent d'imprimer le numéro de mesure au lieu des compteurs alphabétique ou numérique.

Pour affiner le positionnement des repères, veuillez vous référer à Section 8.2 [Mise en forme du texte], page 329, et tout particulièrement ce qui concerne la break-alignable-interface au chapitre Section 36.9 [Alignement des objets], page 820.

Les définitions format-mark-numbers et format-mark-letters sont inscrites dan le fichier scm/translation-functions.scm. Elles seront une source d'inspiration en matière de fonctions de formatage.

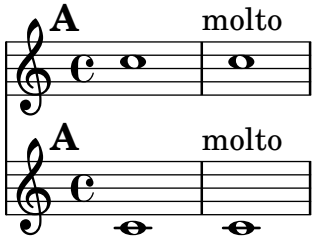
Morceaux choisis

Impression des indications sur toutes les portées d'un système

Bien que ces indications textuelles et repères ne soient habituellement imprimées qu'au niveau de la portée supérieure d'un système, leur affectation peut être répercutée à chacune des portées.

```
\score {
  <<
    \new Staff { \mark \default c''1 \textMark "molto" c'' }
    \new Staff { \mark \default c'1 \textMark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove Mark_engraver
      \remove Text_mark_engraver
      \remove Staff_collecting_engraver
    }
    \context {
      \Staff
      \consists Mark_engraver
      \consists Text_mark_engraver
      \consists Staff_collecting_engraver
```

```
    }
  }
}
```



Voir aussi

Manuel de notation : Section 36.9 [Alignement des objets], page 820, Section 2.3.3 [Indication métronomique], page 88, Section B.8 [La fonte Emmentaler], page 935, Section 8.1.4 [Marque de section], page 323, Section 8.2 [Mise en forme du texte], page 329, Section 8.2.5 [Notation musicale dans du texte formaté], page 345.

Fichiers d'initialisation : `scm/translation-functions.scm`.

Morceaux choisis : Section "Rythme" dans *Morceaux choisis*.
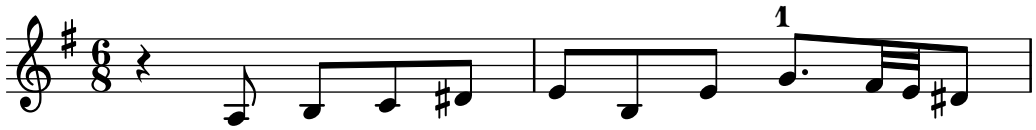
Référence des propriétés internes : Section "AdHocMarkEvent" dans *Référence des propriétés internes*, Section "RehearsalMark" dans *Référence des propriétés internes*, Section "RehearsalMarkEvent" dans *Référence des propriétés internes*.
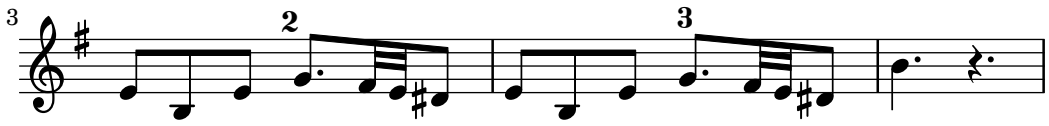
2.5.6 Compteurs de mesures

Les compteurs de mesure constituent un moyen de numéroter des mesures consécutives pour, par exemple, assister l'interprète dans le décompte de mesures lors des reprises. Cette fonctionnalité requiert l'adjonction du `Measure_counter_engraver` à un contexte du type `Staff` ou `Score`.

```
\layout {
  \context {
    \Staff
    \consists Measure_counter_engraver
  }
}

\relative c' {
  \time 6/8
  \key e \minor
  r4 a8 b c dis
  \startMeasureCount
  \*3 {
    e8 b e g8. fis32 e dis8
  }
  \stopMeasureCount
  b'4. r
}
```

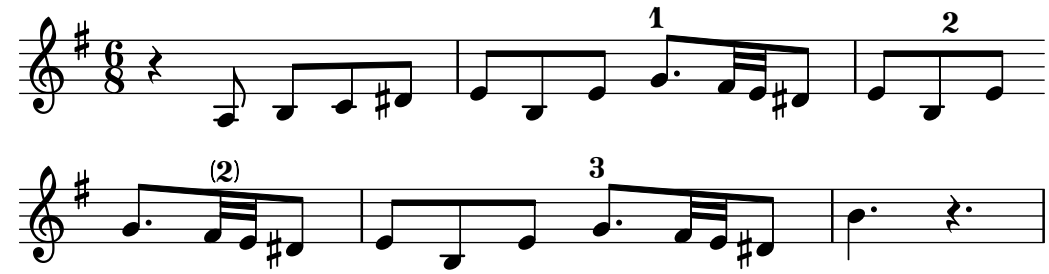




Le numéro des mesures tronquées se présente entre parenthèses.

```
\layout {
  \context {
    \Staff
    \consists Measure_counter_engraver
  }
}

\relative c' {
  \time 6/8
  \key e \minor
  r4 a8 b c dis
  \startMeasureCount
  e8 b e g8. fis32 e dis8
  e8 b e \break g8. fis32 e dis8
  e8 b e g8. fis32 e dis8
  \stopMeasureCount
  b'4. r
}
```
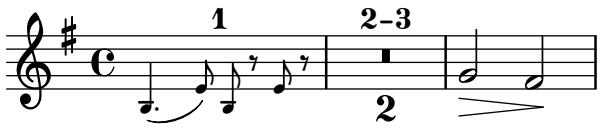


La compression de silences multimesure reçoit un traitement particulier : est présenté l'intervalle de mesures considéré.

```
\layout {
  \context {
    \Staff
    \consists Measure_counter_engraver
  }
  \context {
    \Voice
    \override MultiMeasureRestNumber.direction = #DOWN
  }
}

\compressMMRests {
  \key e \minor
  \startMeasureCount
  \new CueVoice {
    b4.( e'8) b8 r e' r
  }
```

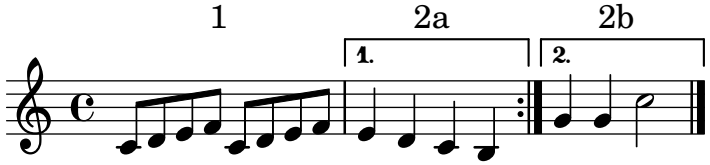
```
  R1*2
  \stopMeasureCount
  g'2\> fis'2\!
}
```



Les compteurs de mesures prennent en considération le style de numérotation des alternatives. Lorsque le style est réglé sur `numbers-with-letters`, leur rendu est meilleur avec une fonte textuelle.

```
\layout {
  \context {
    \Score
    alternativeNumberingStyle = #'numbers-with-letters
  }
  \context {
    \Staff
    \consists Measure_counter_engraver
    \override MeasureCounter.Y-offset = 6
    \override MeasureCounter.font-encoding = #'latin1
    \override MeasureCounter.font-size = 1
  }
}

\relative c' {
  \startMeasureCount
  \repeat volta 2 {
    c8 d e f c d e f
  }
  \alternative {
    { e4 d c b }
    { g'4 g c2 }
  }
  \bar "|."
  \stopMeasureCount
}
```



Commandes prédéfinies

`\startMeasureCount`, `\stopMeasureCount`.

Voir aussi

Manuel de notation : Section 6.3.4 [Compression de mesures vides], page 282, Section 33.4 [Modification des greffons de contexte], page 769, Section 2.5.3 [Numéros de mesure], page 135, Chapitre 28 [Sauts], page 711.

Référence des propriétés internes : Section “Measure_counter_engraver” dans *Référence des propriétés internes*, Section “MeasureCounter” dans *Référence des propriétés internes*, Section “measure-counter-interface” dans *Référence des propriétés internes*.

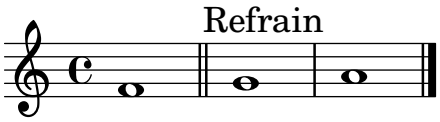
2.5.7 Division en sections

La commande `\section` marque un point dans la musique où une section se termine et une autre commence. Il n’y a pas nécessairement de musique à la suite : cela peut aussi indiquer que cette terminaison apparente n’est pas la fin du morceau comme lorsqu’apparaît l’instruction *D.C.* ou lorsqu’un mouvement s’enchaîne directement avec le suivant. En règle générale, `\section` crée une double barre de mesure. Ses effets peuvent aussi dépendre d’autres considérations, comme par exemple une barre de reprise.

La commande `\fine` indique la fin de la pièce, normalement par une barre finale. Son utilisation ne se limite toutefois pas à la dernière note : elle peut apparaître au sein d’un bloc `\repeat` – voir Section 4.1.5 [Reprises *Al-fine*], page 199.

Une section peut être nommée à l’aide de `\sectionLabel` – voir Section 8.1.4 [Marque de section], page 323).

```
\fixed c' {
  f1
  \section
  \sectionLabel "Refrain"
  g1
  a1
  \fine
}
```



De plus amples détails sur les interactions des barres de mesure `\fine` et `\section` avec les autres types de barre de mesure, ainsi que sur les manières d’en changer l’apparence sont répertoriées dans Section 2.5.2 [Barres de mesure automatiques], page 133.

Voir aussi

Glossaire musicologique : Section “fine” dans *Glossaire*.

Manuel de notation : Section 2.5.2 [Barres de mesure automatiques], page 133, Section 17.4.4 [Divisions], page 571, Section 8.1.4 [Marque de section], page 323, Section 4.1.5 [Reprises *Al-fine*], page 199.
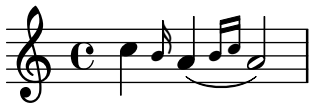
Référence des propriétés internes : Section “FineEvent” dans *Référence des propriétés internes*, Section “SectionEvent” dans *Référence des propriétés internes*.

2.6 Fonctionnalités rythmiques particulières

2.6.1 Notes d’ornement
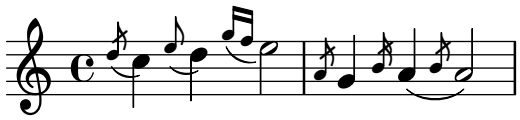
Les petites notes sont des ornements entièrement écrits. Leur taille est un peu plus petite que celle des notes normales et elles n’occupent pas de temps dans la mesure.

```
\relative {
  c''4 \grace b16 a4(
  \grace { b16 c16 } a2)
}
```
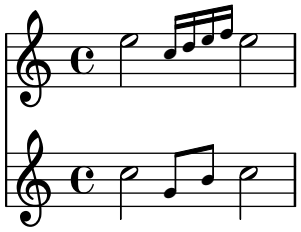


Les plus courantes sont les acciaccatures, qui doivent se jouer très vite, et qui s'écrivent sous forme d'une petite note barrée (sur la hampe) et liée. L'appoggiature est une petite note non barrée qui vole une fraction à la durée de la note réelle qui la suit. LilyPond dispose aussi, grâce à la fonction `\slashedGrace`, d'une petite note barrée et dépourvue de liaison, qui viendra s'insérer entre deux notes déjà liées.

```
\relative {
  \acciaccatura d''8 c4
  \appoggiatura e8 d4
  \acciaccatura { g16 f } e2
  \slashedGrace a,8 g4
  \slashedGrace b16 a4(
  \slashedGrace b8 a2)
}
```
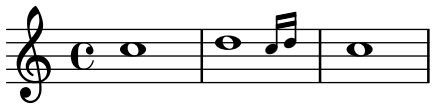


Les petites notes se placent de façon synchrone entre les différentes portées. Dans l'exemple suivant, il y a deux petites double-croches pour chaque petite croche.

```
<<
  \new Staff \relative { e''2 \grace { c16 d e f } e2 }
  \new Staff \relative { c''2 \grace { g8 b } c2 }
>>
```



La commande `\afterGrace` sert à placer une petite note après une note réelle – et non avant comme d'ordinaire. Cette commande requiert deux arguments : la note réelle, et la ou les petites notes qui s'y rattachent.

```
\relative { c''1 \afterGrace d1 { c16[ d] } c1 }
```
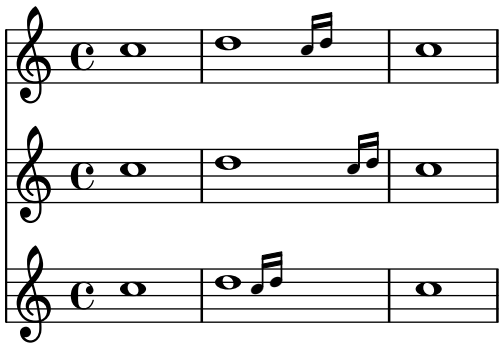


Les petites notes se placent alors **après** la note réelle. Leur positionnement est déterminé par une fraction de la durée de la note principale. Cette fraction, fixée par défaut à

```
afterGraceFraction = 3/4
```

peut être changée en début de fichier. Elle peut aussi se définir à la suite de la commande `afterGrace`.
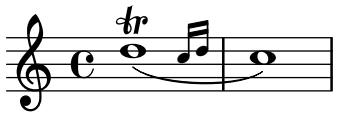
Dans l'exemple suivant, vous pouvez observer la différence entre le comportement par défaut, à 15/16 et enfin à la moitié de la durée de base.

```
<<
  \new Staff \relative {
    c''1 \afterGrace d1 { c16[ d] } c1
  }
  \new Staff \relative {
    c''1 \afterGrace 15/16 d1 { c16[ d] } c1
  }
  \new Staff \relative {
    c''1 \afterGrace 1/2 d1 { c16[ d] } c1
  }
>>
```



Les effets d'une commande `\afterGrace` peuvent aussi s'obtenir à l'aide de silences invisibles. Nous pourrions positionner ces petites notes à sept huitièmes de la durée de la note de base :
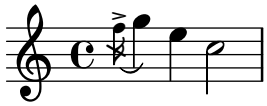
```
\new Voice \relative {
  <<
    { d''1^\trill_( }
    { s2 s4. \grace { c16 d } }
  >>
  c1)
}
```



Les expressions `\grace` obéissent à des règles typographiques particulières, notamment en matière d'orientation et de taille des objets. De ce fait, toute subtilité de mise en forme devra être indiquée **à l'intérieur** de l'expression introduite par `\grace` ; ces réglages additionnels doivent également être désactivés dans cette même expression.

```
\new Voice \relative {
  \acciaccatura {
    \stemDown
    f''16->
    \stemNeutral
  }
  g4 e c2
```
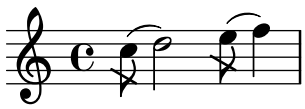
```
}
```



Morceaux choisis

Utilisation de hampe barrée pour une note normale

Le trait que l'on trouve sur les hampes des acciaccatures peut être appliqué dans d'autres situations.

```
\relative c'' {
  \override Flag.stroke-style = "grace"
  c8( d2) e8( f4)
}
```
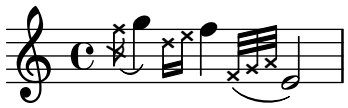


Mise en forme des notes d'ornement

Il est possible de changer globalement l'apparence des notes d'ornement dans un morceau, au moyen des fonctions `add-grace-property` et `remove-grace-property`.

Ici, par exemple, on ôte la définition de l'orientation (propriété `direction`) des objets `Stem` pour toutes les petites notes, afin que les hampes ne soient pas toujours orientées vers le haut, et on leur préfère des têtes en forme de croix.

```
\relative c'' {
  \new Staff {
    $(remove-grace-property 'Voice 'Stem 'direction)
    $(add-grace-property 'Voice 'NoteHead 'style 'cross)
    \new Voice {
       \acciaccatura { f16 } g4
       \grace { d16 e } f4
       \appoggiatura { f,32 g a } e2
    }
  }
}
```



Redéfinition des réglages de mise en forme par défaut des notes d'ornement

Les réglages par défaut des notes d'ornement sont stockés dans les variables suivantes :
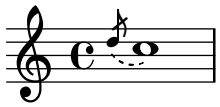
```
startGraceMusic
stopGraceMusic
startAcciaccaturaMusic
stopAcciaccaturaMusic
startAppoggiaturaMusic
stopAppoggiaturaMusic
```

Ces variables sont définies dans le fichier `ly/grace-init.ly`. Amender leur définition permet d'en varier les effets.

```
startAcciaccaturaMusic = {
  <>(
  \override Flag.stroke-style = "grace"
  \slurDashed
}

stopAcciaccaturaMusic = {
  \revert Flag.stroke-style
  \slurSolid
  <>)
}

\relative c'' {
  \acciaccatura d8 c1
}
```



Positionnement des notes d'ornement avec espace flottant

Lorsque la propriété `strict-grace-spacing` est activée, l'espacement des notes d'ornement se fera de manière « élastique ». Autrement dit, elles seront décollées de leur note de rattachement : LilyPond commence par espacer les notes normales, puis les ornements sont placés à la gauche de leur note de rattachement.

Cependant, en raison du ticket 6876 (`https://gitlab.com/lilypond/lilypond/-/issues/6876`), les altérations accidentelles sont ignorées lorsque cette propriété est activée. Le code ci-dessous propose une solution de contournement à ce problème.

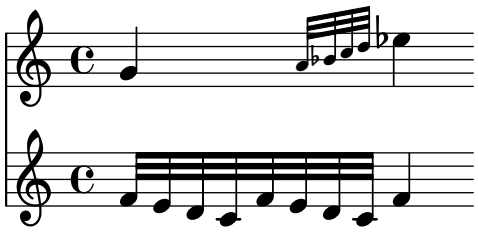
Autre effet indésirable de cette propriété, LilyPond ne vérifie pas qu'il y a suffisamment d'espace pour les notes d'ornement (ceci fait l'objet du ticket 2630 (`https://gitlab.com/lilypond/lilypond/-/issues/2630`)). Il faut donc s'assurer d'avoir suffisamment d'espace disponible en recourant conjointement `\newSpacingSection` et une valeur appropriée pour le `base-shortest-duration` de l'objet graphique `SpacingSpanner`.

```
shiftedGrace =
#(define-music-function (offset music) (number? ly:music?)
   #{
     \override NoteHead.X-offset = #(- offset 0.85)
     \override Stem.X-offset = #offset
     \grace { $music }
     \revert NoteHead.X-offset
     \revert Stem.X-offset
   #})

\relative c'' <<
  { g4 \shiftedGrace #-1.3 a32 \shiftedGrace #-0.5 { bes c d } es4 }
  { f,32 e d c f e d c f4 }
>>

\layout {
```

```
  \context {
    \Score
    \override SpacingSpanner.strict-grace-spacing = ##t
  }
}
```



Voir aussi

Glossaire musicologique : Section “ornements” dans *Glossaire*, Section “acciaccature” dans *Glossaire*, Section “appoggiature” dans *Glossaire*.

Manuel de notation : Section 2.4.3 [Barres de ligature manuelles], page 117, Section 2.1.3 [Changement d’échelle des durées], page 62.

Fichiers d’initialisation : `ly/grace-init.ly`.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.
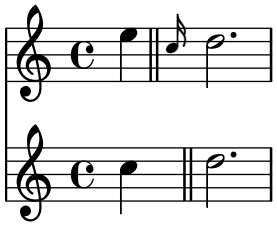
Référence des propriétés internes : Section “GraceMusic” dans *Référence des propriétés internes*, Section “Grace_auto_beam_engraver” dans *Référence des propriétés internes*, Section “Grace_beam_engraver” dans *Référence des propriétés internes*, Section “Grace_engraver” dans *Référence des propriétés internes*, Section “Grace_spacing_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un groupe de notes ligaturées constituant une *acciaccatura* apparaîtra comme une *appoggiatura*, c’est-à-dire sans trait.
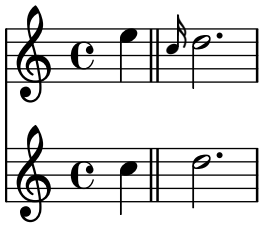
La synchronisation des petites notes se fait de façon parfois surprenante, car les autres objets de la portée – barre de mesure, armure, etc. – sont eux aussi synchrones. Pensez-y lorsque vous mêlez des portées comprenant des petites notes et d’autres sans :

```
<<
  \new Staff \relative { e''4 \section \grace c16 d2. }
  \new Staff \relative { c''4 \section d2. }
>>
```



Il est possible de remédier à cela en insérant, sur les autres portées, des silences invisibles dans une expression précédée de \grace et correspondant à la durée des petites notes :

```
<<
  \new Staff \relative { e''4 \section \grace c16 d2. }
  \new Staff \relative { c''4 \section \grace s16 d2. }
>>
```



Bien que la partie visible contient un `\acciaccatura` ou un `\appoggiatura`, veillez bien à utiliser l'instruction `\grace` dans la partie invisible, au risque de voir apparaître un tronçon de liaison connectant la petite note invisible à la note qui la suit.

Seules des expressions musicales séquentielles peuvent être utilisées pour des petites notes ; il n'est pas possible d'imbriquer ni de juxtaposer des sections de petites notes, faute de quoi le traitement du code peut échouer ou produire des erreurs.

En ce qui concerne la sortie MIDI, les petites notes ont une durée du quart de la valeur que vous leur attribuez. Par voie de conséquence, si la durée globale d'une succession de petites notes venait à dépasser la durée de la note qui précède, vous déclencheriez une erreur du type « `Going back in MIDI time` ». Il vous faudra donc raccourcir les petites notes. Par exemple,

```
c'8 \acciaccatura { c'8[ d' e' f' g'] }
```

deviendrait

```
c'8 \acciaccatura { c'16[ d' e' f' g'] }
```

ou bien modifier explicitement l'échelle des durées :

```
c'8 \acciaccatura { \scaleDurations 1/2 { c'8[ d' e' f' g'] } }
```

Voir Section 2.1.3 [Changement d'échelle des durées], page 62.

2.6.2 Alignement et cadences

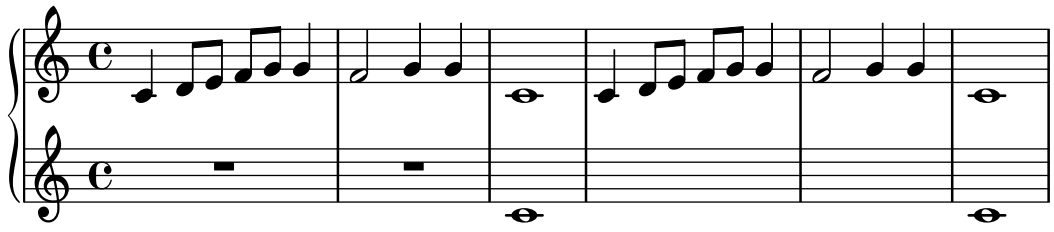
Dans un contexte orchestral, une cadence constitue un problème spécifique. Lors du montage d'une partition contenant une cadence, tous les autres instruments doivent sauter autant de notes que ce qu'en comporte la cadence, faute de quoi ils démarreraient trop tôt ou trop tard.

Les fonctions `mmrest-of-length` ou `skip-of-length` permettent de pallier ce problème. Ces fonctions Scheme prennent en argument un fragment de musique, et génèrent un `\skip` ou un silence multimesure d'une durée correspondant à ce fragment.

```
MyCadenza = \relative {
  c'4 d8 e f g g4
  f2 g4 g
}

\new GrandStaff <<
  \new Staff {
    \MyCadenza c'1
    \MyCadenza c'1
  }
  \new Staff {
    #(mmrest-of-length MyCadenza)
    c'1
    #(skip-of-length MyCadenza)
```

```
    c'1
  }
>>
```

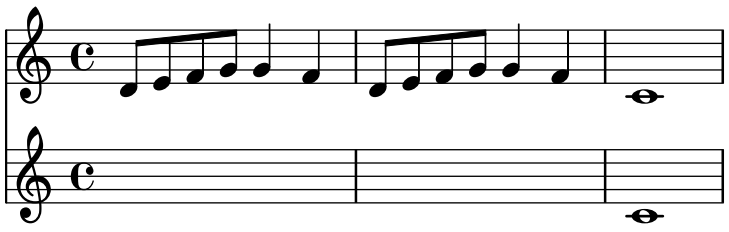


Une autre solution consiste à utiliser la commande \skip.

```
MyCadenza = \fixed c' {
  \repeat volta 2 {
    d8 e f g g4 f4
  }
}

music = <<
  \new Staff {
    \MyCadenza
    c'1
  }
  \new Staff {
    \skip \MyCadenza
    c'1
  }
>>

\unfoldRepeats \music
```
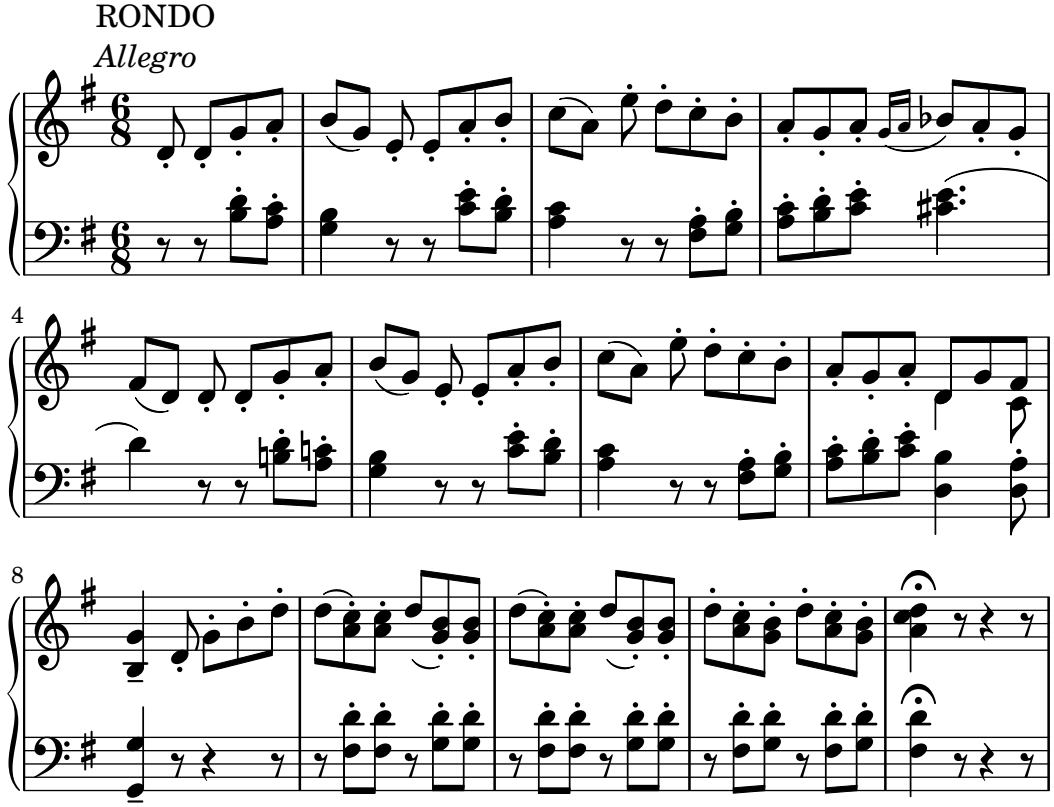


Voir aussi

Glossaire musicologique : Section “cadenza” dans *Glossaire*.

Morceaux choisis : Section “Rythme” dans *Morceaux choisis*.

3 Signes d'interprétation



Ce chapitre traite des différentes indications d'interprétation que l'on peut trouver sur les partitions.

3.1 Signes d'interprétation attachés à des notes

Nous allons voir au cours de ces lignes comment ajouter aux notes des indications d'interprétation – articulation, ornementation, nuance – et aborderons la manière de générer vos propres signes.

3.1.1 Articulations et ornements
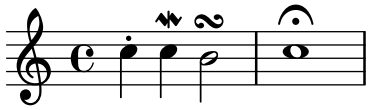
accent accent, articulationmarcato marcato, articulationstaccatissimo staccatissimo, articulationespressivo espressivo, articulationstaccato staccato, articulationtenuto tenuto, articulationportato portato, articulationpouss pouss, archettir tir, archetflageolet flageolet, harmoniquespouce pouce, doigttalon talon, pdalierpointe pointe, pdalierouvert ouvert, articulationsemi-ouvert semi-ouvert, articulationstop stop, articulationsnappizzicato snappizzicato, signegrupetto grupetto, ornementgrupetto renvers grupetto renvers, ornementgrupetto barr grupetto barr, ornementgrupetto Haydn grupetto Haydn, ornementtrille trille, ornementmordant mordant, ornementmordant ascendant mordant ascendant, ornementmordant descendant mordant descendant, ornementsignum congruentiae signum congruentiae, ornementsegno segno, signecoda coda, signevariante coda variante coda, signebachschleifer bachschleifer, articulation

Les différents symboles qui indiquent des ponctuations ou des modes de jeu différents s'ajoutent aux notes de la manière suivante :

```
note\nom
```

Les valeurs de *nom* sont répertoriées dans l'annexe Section B.13 [Liste des signes d'articulation], page 943. En voici un exemple :

```
\relative {
  c''4\staccato c\mordent b2\turn
  c1\fermata
}
```



marcato marcato, articulation

tenuto tenuto, articulation

staccatissimo staccatissimo, articulation
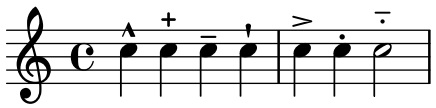
accent accent, articulation

staccato staccato, articulation

portato portato, articulation

Certains signes d'articulation disposent d'un raccourci. On les ajoute à chaque note au moyen d'un tiret suivi du caractère correspondant à l'articulation désirée. C'est entre autres le cas pour *marcato*, *stopped*, *tenuto*, *staccatissimo*, *accent*, *staccato*, et *portato*, comme l'illustre l'exemple ci-dessous.
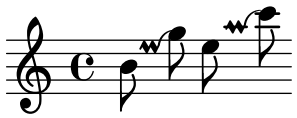
```
\relative {
  c''4-^  c-+  c--  c-!
  c4->  c-.  c2-_
}
```



Même si LilyPond place automatiquement ces symboles, selon les règles contenues dans le fichier `scm/script.scm`, il est possible de l'obliger à les positionner au-dessus ou en dessous de la note, comme indiqué au chapitre Section 36.1 [Direction et positionnement], page 801.

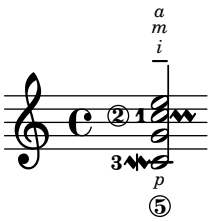
Le `bachschleifer` se positionne sur la gauche d'un `NoteHead`. Cet ornement peut comporter des lignes supplémentaires.

```
{
  \autoBeamOff
  b'8 g''\bachschleifer
  e'' c'''\bachschleifer
}
```



Articulations et ornements courants peuvent se positionner aussi bien sur la gauche que sur la droite d'une tête de note en portant une dérogation à leur `side-axis` et, si nécessaire, à la propriété `direction`. Les fonctions `\atLeft` et `\atRight` permettent de le gérer sans encombre.

```
{
  \set fingeringOrientations = #'(left)
  \set stringNumberOrientations = #'(left down)
  <
   c'-3\5_\rightHandFinger #1 \atLeft \mordent
   g'^\rightHandFinger #2
   c''-1\2^\rightHandFinger #3 \atRight \prall
   e'' ^\rightHandFinger #4
  >2^\tenuto
}
```
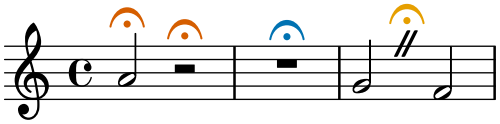


Commandes prédéfinies

`\atLeft`, `\atRight`.

Le type d'objet graphique créé par une articulation dépend de ce à quoi il est attaché.

- Attachées à des notes ou des silences ordinaires, les articulations créent des objets `script`.
- Attachées à des silences multimesures, les articulations créent des objets `MultiMeasureRestScript`.

```
\override Score.Script.color = #(universal-color 'vermillion)
\override Score.MultiMeasureRestScript.color = #(universal-color 'blue)
\override Score.CaesuraScript.color = #(universal-color 'orange)
a'2\fermata r\fermata
R1\fermata
g'2 \caesura \fermata f'2
```



En dehors des articulations habituelles, vous pouvez adjoindre du texte – avec ou sans mise en forme – à n'importe quelle note. Voir à ce propos Section 8.1.2 [Commentaires textuels], page 320.

Pour plus d'information sur la manière d'ordonner `Scripts` et `TextScripts`, consultez le chapitre Section "Positionnement des objets" dans *Manuel d'initiation*.

Morceaux choisis

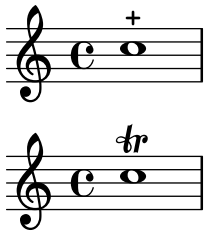
Modification de la signification des raccourcis pour les signes d'articulation

Les raccourcis sont répertoriés dans le fichier `ly/script-init.ly`, dans lequel on retrouve les variables `dashHat`, `dashPlus`, `dashDash`, `dashBang`, `dashLarger`, `dashDot` et `dashUnderscore` ainsi que leur valeur par défaut. Ces valeurs peuvent être modifiées selon vos besoins. Il suffit par exemple, pour affecter au raccourci `-+` (`dashPlus`) le symbole du trille en lieu et place du `+` (caractère plus), d'assigner la valeur `\trill` à la variable `dashPlus` :

```
\relative c'' { c1-+ }

dashPlus = \trill

\relative c'' { c1-+ }
```



Contrôle de l'ordre vertical des articulations et ornements

Les symboles s'ordonnent verticalement suivant la propriété `script-priority`. Plus sa valeur numérique est faible, plus le symbole sera proche de la note. Dans l'exemple suivant, l'objet `TextScript` – le dièse – a d'abord la propriété la plus basse et se voit donc placé au plus près de la note ; ensuite, c'est l'objet `Script` – le mordant – qui a la propriété la plus basse, et se place alors sous le dièse. Lorsque deux objets ont la même priorité, c'est l'ordre dans lequel ils sont indiqués qui détermine lequel sera placé en premier.

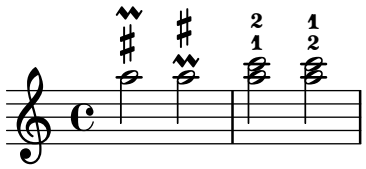
Il est à noter que pour les objets `Fingering`, `StringNumber` et `StrokeFinger`, lorsqu'ils apparaissent dans un accord, leur ordonnancement vertical est aussi déterminé par le positionnement vertical de leur tête de note d'attachement qui sera ajoutée (ou soustraite selon la direction) de la valeur de `script-priority` du *grob*. Ceci permet de s'assurer que, pour des doigtés au-dessous d'un accord, la note la plus basse soit associée au doigté le plus bas, et inversement. L'ordre dans lequel les hauteurs sont saisies dans l'accord est ici sans importance.

Par défaut, les scripts les moins techniques sont inscrits au plus près de la tête de note. L'ordre de base est articulation, flageolet, doigté, doigté main droite, numéro de corde, point d'orgue, archet et script textuel.

```
\relative c''' {
  \once \override TextScript.script-priority = -100
  a2^\prall^\markup { \sharp }

  \once \override Script.script-priority = -100
  a2^\prall^\markup { \sharp }

  \set fingeringOrientations = #'(up)
  <c-2 a-1>2
  <a-1 c\tweak script-priority -100 -2>2
}
```
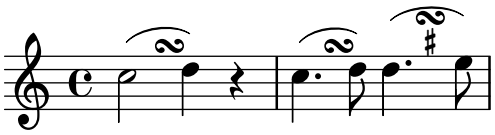


Voir Section B.18 [Valeurs par défaut de `script-priority`], page 953.

Création d'un grupetto retardé

Obtenir un *gruppetto* retardé et dans lequel la note la plus basse est altérée requiert quelques surcharges. La propriété `outside-staff-priority` doit être désactivée (`#f`) pour éviter qu'elle

prenne le pas sur la propriété `avoid-slur`. L'ajustement du premier argument de `\after` (il s'agit d'une durée) permet de jouer sur le positionnement horizontal.

```
\relative c'' {
  \after 2*2/3 \turn c2( d4) r |
  \after 4 \turn c4.( d8)
  \after 4
  {
    \once \set suggestAccidentals = ##t
    \once \override AccidentalSuggestion.outside-staff-priority = ##f
    \once \override AccidentalSuggestion.avoid-slur = #'inside
    \once \override AccidentalSuggestion.font-size = -3
    \once \override AccidentalSuggestion.script-priority = -1
    \once \hideNotes
    cis8\turn \noBeam
  }
  d4.( e8)
}
```



Voir aussi

Glossaire musicologique : Section "accent" dans *Glossaire*, Section "portato" dans *Glossaire*, Section "staccato" dans *Glossaire*, Section "tenuto" dans *Glossaire*.

Manuel d'initiation : Section "Positionnement des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 8.1.2 [Commentaires textuels], page 320, Section 36.1 [Direction et positionnement], page 801, Section B.13 [Liste des signes d'articulation], page 943, Section 3.3.3 [Trilles], page 189.

Fichiers d'initialisation : `scm/script.scm`.

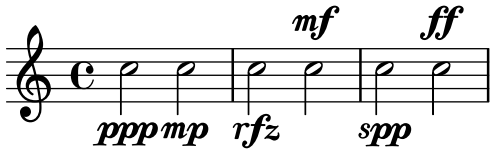
Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "CaesuraScript" dans *Référence des propriétés internes*, Section "MultiMeasureRestScript" dans *Référence des propriétés internes*, Section "Script" dans *Référence des propriétés internes*, Section "TextScript" dans *Référence des propriétés internes*.

3.1.2 Nuances

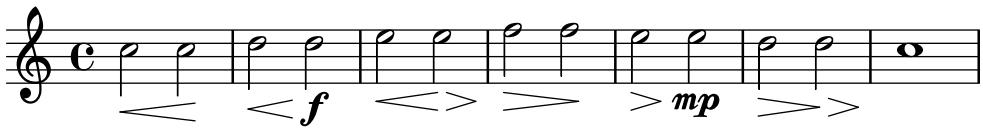
À chaque nuance absolue correspond une commande qui peut être indiquée après une note : `c4\ff` par exemple. Les commandes de nuance disponibles sont `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, `\rfz`et `\n`. Les nuances se placent aussi bien en dessous qu'au-dessus de la portée ; pour plus d'information, consultez Section 36.1 [Direction et positionnement], page 801.

```
\relative c'' {
  c2\ppp c\mp
  c2\rfz c^\mf
  c2_\spp c^\ff
}
```



Un crescendo est délimité par \< et \!, ou peut se terminer par une commande de nuance explicite, ou bien un decrescendo ou un nouveau crescendo. Il en va de même pour un diminuendo. Au lieu de \< et \>, vous pouvez utiliser \cr et \decr, tout comme \endcr et \enddecr au lieu de \!, auquel cas LilyPond n'imprimera pas de soufflet (*hairpin* en anglais).

```
\relative c'' {
  c2\< c\!
  d2\< d\f
  e2\< e\>
  f2\> f\!
  e2\> e\mp
  d2\> d\>
  c1\!
}
```
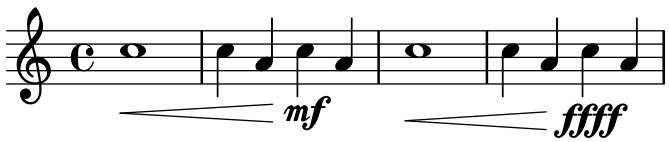


Un soufflet terminé par un simple \! prendra fin sur la droite de la tête de note à laquelle il est attaché. Dans le cas où il se termine par l'intervention d'un autre soufflet (contraire ou non), il prendra fin au milieu de la tête de note affublée d'un \< ou d'un \>, et le nouveau soufflet débutera à l'extrémité droite de cette même tête de note. Un soufflet se terminant sur le premier temps d'une mesure s'arrêtera à la barre de mesure.

```
\relative {
  c''1\< | c4 a c\< a | c4 a c\! a\< | c4 a c a\!
}
```



Il en va de même lorsqu'un soufflet est interrompu par une nuance explicite. Notez bien que la largeur occupée par cette nuance explicite influe sur la terminaison du soufflet :

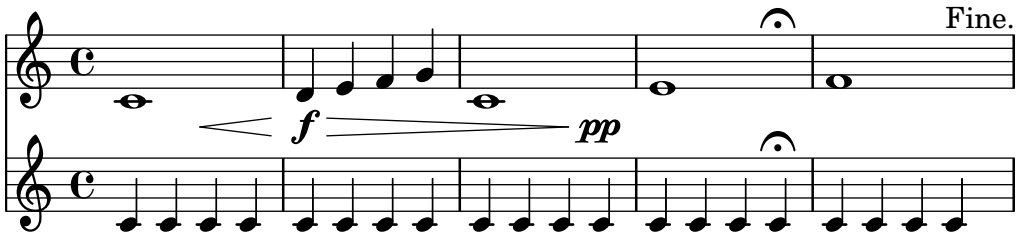
```
\relative {
  c''1\< | c4 a c\mf a | c1\< | c4 a c\ffff a
}
```



Les indications de nuance sont attachées aux notes ; aussi, lorsque l'on veut faire se succéder plusieurs nuances pendant une note tenue, il faudra avoir recours à la commande `\after` qui permet aussi de gérer le différé des articulations et scripts textuels :

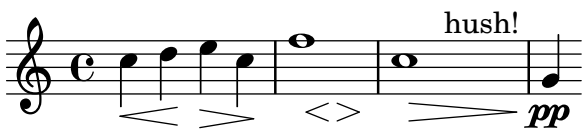
```
<<
  \relative {
    \after 2 \< c'1
    d4\f\> e f g
    \after 2. \pp c,1

    \after 2. \fermata e
    \after 2. ^"Fine." f
  }
  \relative {
    \*12 c'4
    c c c c\fermata
    c c c c
  }
>>
```



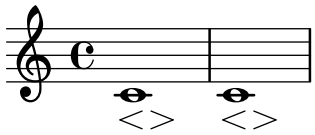
Plusieurs instances de `\after` peuvent se succéder afin de graver plusieurs indications pour une note. Ceci est particulièrement utile lorsqu'il faut ajouter un crescendo et un decrescendo à une même note :

```
\relative {
  c''4\< d\! e\> c\!
  \after 4 \< \after 2\> \after 2. \! f1
  \textLengthOn
  \after 4 \> \after 2. ^"hush!" c1
  g4\pp
}
```
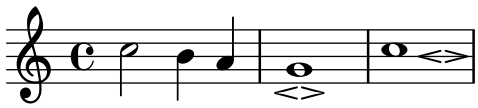


Lorsque la première d'une succession d'indications sur une même note coïncide avec son attaque, il est judicieux de l'attacher à un accord vide <> ; les indications seront alors saisies selon leur ordre d'apparition :

```
{
  <>\< \after 4 \> \after 2 \! c'1
  % plus facile à écrire et lire que :
  \after 4 \> \after 2 \! c'1\<
}
```
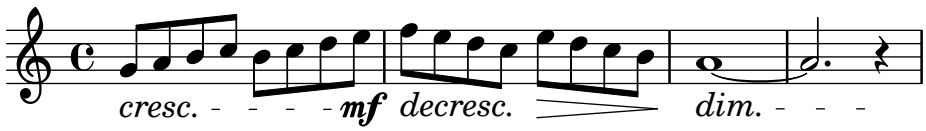


On peut aussi avoir recours à l'indication `\espressivo` pour indiquer un crescendo suivi d'un decrescendo sur une même note. Gardez à l'esprit qu'il s'agit d'une articulation, et en aucun cas d'une nuance.

```
\relative {
  c''2 b4 a
  g1\espressivo
  \after 2. \espressivo c
}
```
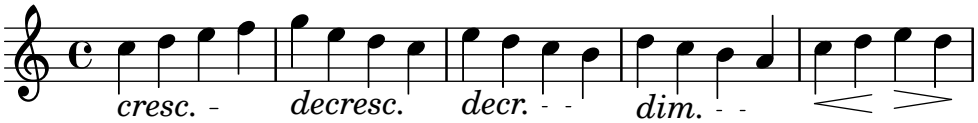


La commande `\cresc` permet d'indiquer textuellement le début d'un crescendo. `\decresc` ou `\dim` marquent le début d'un decrescendo. Les lignes d'extension sont gérées automatiquement.

```
\relative {
  g'8\cresc a b c b c d e\mf |
  f8\decresc e d c e\> d c b |
  a1\dim ~ |
  a2. r4\! |
}
```



Une indication textuelle peut indiquer, au lieu d'un soufflet, un changement de nuance :

```
\relative c'' {
  \crescTextCresc
  c4\< d e f\! |
  \dimTextDecresc
  g4\> e d c\! |
  \dimTextDecr
  e4\> d c b\! |
  \dimTextDim
  d4\> c b a\! |
  \crescHairpin
  \dimHairpin
  c4\< d\! e\> d\! |
}
```
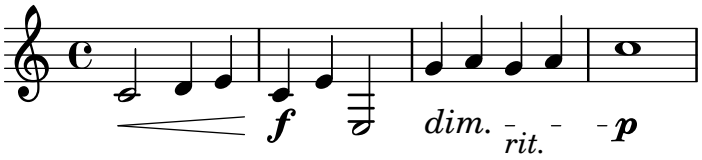


Pour créer des indications de nuance qui restent alignées avec les nuances habituelles, reportez-vous au chapitre Section 3.1.3 [Personnalisation des indications de nuance], page 171.

Le positionnement vertical des nuances est géré par le Section "DynamicLineSpanner" dans *Référence des propriétés internes*.

L'utilisation d'un contexte `Dynamics` permet de graver les nuances sur leur propre ligne – Il suffit de placer des silences invisibles pour gérer le temps. Bien que le contexte `Dynamics` accepte des notes pour indiquer les durées, celles-ci ne seront pas imprimées. Le contexte `Dynamics` peut aussi contenir des indications textuelles avec ou sans extenseur, ainsi que les indications de pédale.

```
<<
  \new Staff \relative {
    c'2 d4 e |
    c4 e e,2 |
    g'4 a g a |
    c1 |
  }
  \new Dynamics {
    s1\< |
    s1\f |
    s2\dim s2-"rit." |
    s1\p |
  }
>>
```



Note : Même si la partition ne comporte qu'une seule indication absolue de nuance, comme un \p, LilyPond crée *deux* objets la concernant – un `DynamicText` et un objet `DynamicLineSpanner` – et les propriétés contrôlant cette indication de nuance sont partagées entre ces deux objets. Par exemple, la taille peut se modifier par la propriété `DynamicText.font-size` alors que le positionnement vertical est contrôlé par `DynamicLineSpanner.Y-offset`.

Par voie de conséquence, un code tel que

```
\tweak font-size 5 \p
```

est fonctionnel, mais

```
\tweak Y-offset 5 \p
```

ne l'est pas. Il faudra donc écrire

```
\tweak DynamicLineSpanner.Y-offset 5 \p
```

Commandes prédéfinies

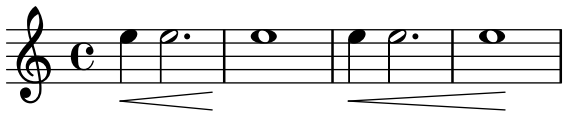
`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`, `\crescTextCresc`, `\dimTextDim`, `\dimTextDecr`, `\dimTextDecresc`, `\crescHairpin`, `\dimHairpin`.

Morceaux choisis

Soufflets et barres de mesure

En principe, un soufflet – (de)crescendo imprimé sous forme graphique – commence au bord gauche de la note de départ, et se termine au bord droit de la note d'arrivée. Cependant, si la note d'arrivée est sur un premier temps, le soufflet s'arrêtera au niveau de la barre de mesure qui la précède. Ce comportement peut être annulé en assignant *faux* (`#f`) à la propriété `to-barline`.

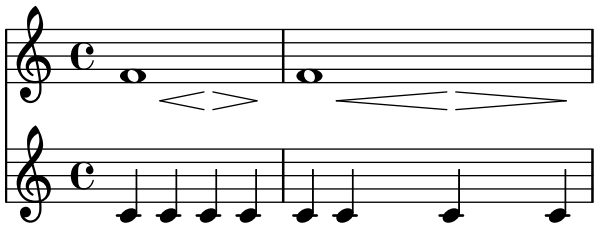
```
\relative c'' {
  e4\< e2.
  e1\!
  \override Hairpin.to-barline = ##f
  e4\< e2.
  e1\!
}
```



Ajustement de la longueur d'un soufflet

Si un soufflet est trop court, il suffit d'ajuster la propriété `minimum-length` de l'objet `Hairpin` pour l'allonger.
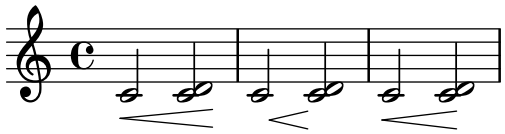
```
<<
  {
    \after 4 \< \after 2 \> \after 2. \! f'1
    \override Hairpin.minimum-length = 8
    \after 4 \< \after 2 \> \after 2. \! f'1
  }
  {
    \*8 c'4
  }
>>
```



Alignement des bornes de soufflet relativement aux `NoteColumn`*s*

Les terminaisons des soufflets peuvent s'aligner sur la gauche, au centre ou sur la droite des *grobs* `NoteColumn` à l'aide d'une dérogation à la propriété `endpoint-alignments` – `LEFT`, `CENTER` ou `RIGHT` – sous forme de paire représentant les extrémités gauche et droite du soufflet. Les `endpoint-alignments` devraient être des directions (soit −1, 0 ou 1), les autres valeurs entraînant l'émission d'un avertissement. L'extrémité droite d'un soufflet se terminant sur un silence ne sera pas affectée et s'alignera toujours sur le bord gauche de ce silence.

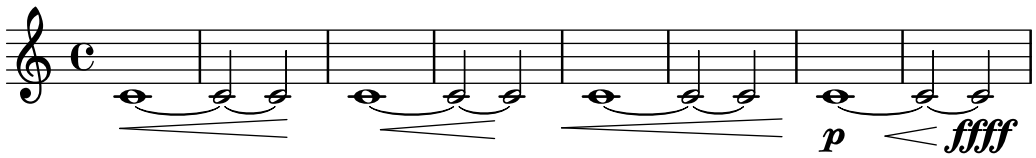
```
{
  c'2\< <c' d'>\! |
  \override Hairpin.endpoint-alignments = #'(1 . -1)
  c'2\< <c' d'>\! |
  \override Hairpin.endpoint-alignments = #`(,LEFT . ,CENTER)
  c'2\< <c' d'>\! |
}
```



Déplacer les extrémités de soufflets

Les terminaisons des soufflets peuvent être décalées en jouant sur la propriété `shorten-pair` de l'objet`Hairpin`. Les valeurs positives déplacent vers l'intérieur, les négatives vers l'extérieur. Contrairement à la propriété `minimum-length`, cette propriété n'affecte que l'apparence du soufflet sans ajuster l'espacement horizontal (y compris avec les nuances textuelles). Cette méthode est donc appropriée aux situations dans lesquelles un soufflet requiert un ajustement fin dans l'espace qui lui est alloué.
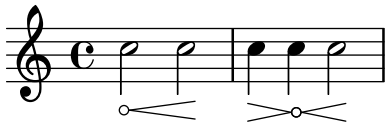
```
{
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(2 . 2)
  c'1~\<
  c'2~ c'\!
  \once \override Hairpin.shorten-pair = #'(-2 . -2)
  c'1~\<
  c'2~ c'\!
  c'1~\p-\tweak shorten-pair #'(2 . 0)\<
  c'2~ c'\ffff
}
```



Impression de soufflets « al niente »

Des crescendos ou decrescendos *al niente* peuvent être indiqués de manière graphique, en assignant *vrai* (`#t`) à la propriété `circled-tip`, ce qui affiche un cercle à leur extrémité.

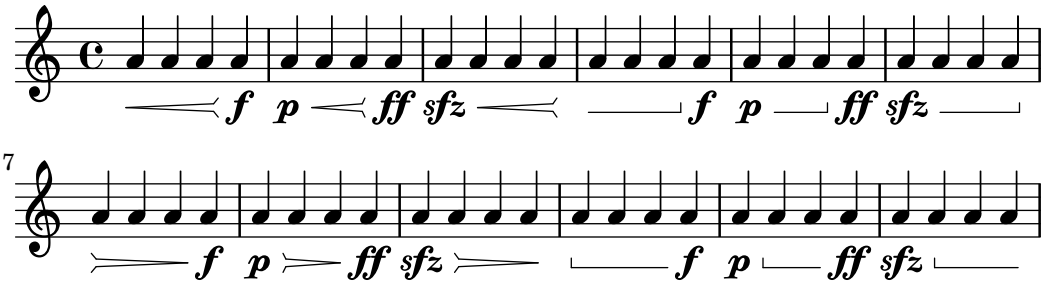
```
\relative c'' {
  \override Hairpin.circled-tip = ##t
  c2\< c\!
  c4\> c\< c2\!
}
```



Différents styles de soufflet

Les soufflets de nuance peuvent adopter des styles différents.

```
\relative c'' {
  \override Hairpin.stencil = #flared-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\< a a a\f
  a4\p\< a a a\ff
  a4\sfz\< a a a\!
  \override Hairpin.stencil = #flared-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
  \override Hairpin.stencil = #constante-hairpin
  a4\> a a a\f
  a4\p\> a a a\ff
  a4\sfz\> a a a\!
}
```



Alignement vertical des nuances et indications textuelles

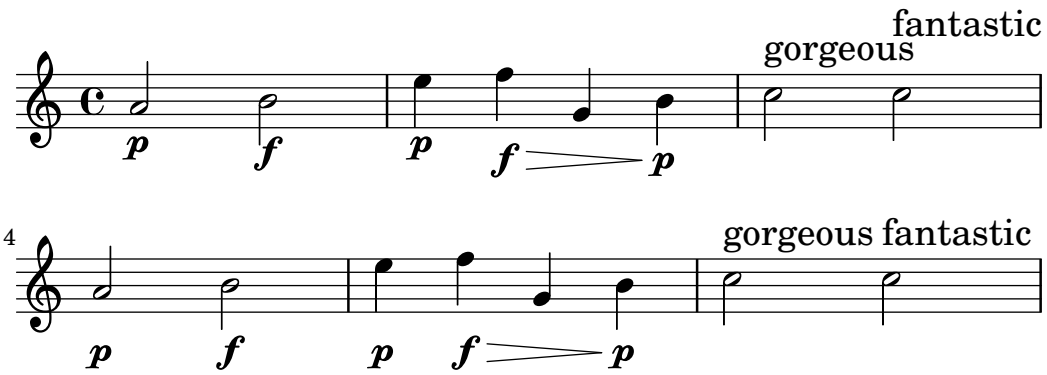
Tous les objets `DynamicLineSpanner` (soufflets ou nuances textuelles) viennent s'aligner sur une ligne de référence placée, par rapport à la portée, à au moins la valeur de la propriété `staff-padding` sauf lorsque d'autres éléments de notation les en éloignent plus. Les nuances seront centrés sur une même ligne dès lors que `staff-padding` aura été défini à une valeur suffisante.

C'est le même principe – en combinaison avec `\textLengthOn` – qui sert à aligner les indications textuelles sur une ligne de référence.

```
music = \relative c' {
  a'2\p b\f
  e4\p f\f\> g, b\p
  c2^\markup { \huge gorgeous } c^\markup { \huge fantastic }
}

{
  \music
  \break
  \override DynamicLineSpanner.staff-padding = 3
  \textLengthOn
  \override TextScript.staff-padding = 1
  \music
```

```
}
```



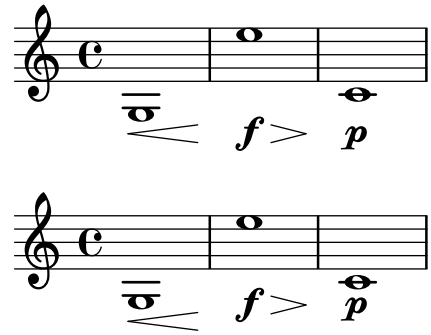
Rupture d'alignement vertical des nuances graphiques et textuelles

Par défaut, LilyPond utilise des objets `DynamicLineSpanner` pour aligner verticalement une succession d'objets nuance tels que soufflets et indications textuelles. Ceci peut, parfois, ne pas être désirable. L'insertion d'un `\breakDynamicSpan`, qui rompt prématurément l'étendue de l'alignement, permet de pallier ce désagrément.

Voir aussi l'exemple « Rupture d'alignement horizontal des nuances graphiques et textuelles ».

```
{ g1\< |
  e''\f\> |
  c'\p }

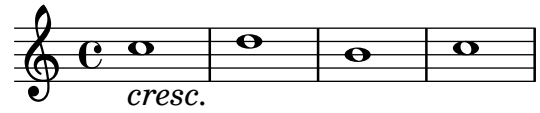
{ g1\< |
  e''\breakDynamicSpan\f\> |
  c'\p }
```



Masquage de l'extension des nuances textuelles

Les crescendos et decrescendos indiqués textuellement – tels que *cresc.* ou *dim.* – sont suivis de pointillés qui montrent leur étendue. On peut empêcher l'impression de ces pointillés.

```
\relative c'' {
  \override DynamicTextSpanner.style = #'none
  \crescTextCresc
  c1\< | d | b | c\!
}
```
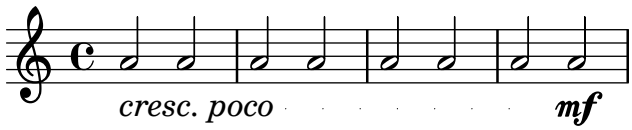


Modification du texte et de l'extension de nuances textuelles

Le texte par défaut des crescendos et decrescendos se change en modifiant les propriétés de contexte `crescendoText` et `decrescendoText`.

L'aspect de la ligne d'extension est fonction de la propriété `style` du `DynamicTextSpanner`. Sa valeur par défaut est `dashed-line`, mais d'autres valeurs sont disponibles, comme `line`, `dotted-line` et `none`.

```
\relative c'' {
  \set crescendoText = \markup { \italic { cresc. poco } }
  \set crescendoSpanner = #'text
  \override DynamicTextSpanner.style = #'dotted-line
  a2\< a
  a2 a
  a2 a
  a2 a\mf
}
```



Voir aussi

Glossaire musicologique : Section "al niente" dans *Glossaire*, Section "crescendo" dans *Glossaire*, Section "decrescendo" dans *Glossaire*, Section "soufflet" dans *Glossaire*.

Manuel d'initiation : Section "Articulations et nuances" dans *Manuel d'initiation*.

Manuel de notation : Section 24.9 [Amélioration du rendu MIDI], page 683, Section 36.1 [Direction et positionnement], page 801, Section 24.4 [Gestion des nuances en MIDI], page 674, Section 3.1.3 [Personnalisation des indications de nuance], page 171.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.
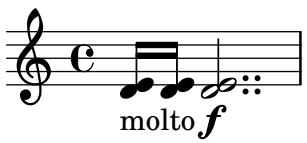
Référence des propriétés internes : Section "DynamicText" dans *Référence des propriétés internes*, Section "Hairpin" dans *Référence des propriétés internes*, Section "DynamicLineSpanner" dans *Référence des propriétés internes*, Section "Dynamics" dans *Référence des propriétés internes*.

3.1.3 Personnalisation des indications de nuance

La manière la plus simple de personnaliser une indication de nuance consiste à utiliser un objet `\markup`.

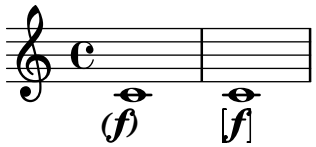
```
moltoF = \markup { molto \dynamic f }

\relative {
  <d' e>16_\moltoF <d e>
  <d e>2..
}
```
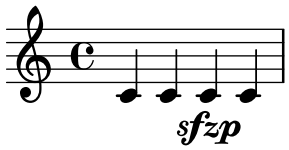


Vous pouvez créer des indications de nuance éditoriales (entre parenthèses ou crochets) grâce aux étiquettes (*mode markup*) ; la syntaxe en est abordée au chapitre Section 8.2 [Mise en forme du texte], page 329.

```
roundF = \markup {
    \center-align \concat { \bold { \italic ( }
          \dynamic f \bold { \italic ) } } }
boxF = \markup { \bracket { \dynamic f } }
\relative {
  c'1_\roundF
  c1_\boxF
}
```



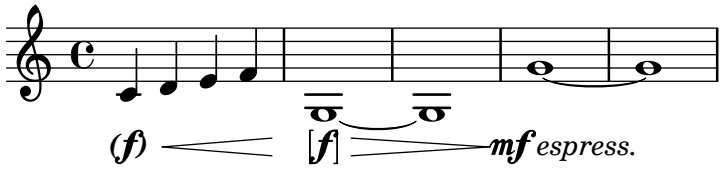
Grâce à la fonction `make-dynamic-script`, vous pouvez créer de nouvelles marques textuelles que vous combinerez avec les signes de nuance.

```
sfzp = #(make-dynamic-script "sfzp")
\relative {
  c'4 c c\sfzp c
}
```



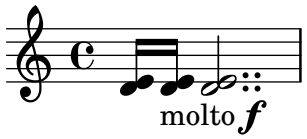
`make-dynamic-script` accepte en argument tout objet de type *markup*. Notez bien que la police des nuances ne contient que les caractères `f`, `m`, `p`, `r`, `s`, `z` et `n`, et que les marques de nuance possèdent des propriétés particulières et prédéfinies quant à leur police. Aussi, lorsque vous créez du texte en pareille situation, nous vous recommandons d'utiliser `\normal-text` pour annuler ces propriétés. Recourir à la fonction `make-dynamic-script` plutôt qu'à un simple *markup* permet de s'assurer que ces objets personnalisés et les soufflets seront alignés lorsque attachés à une même note.

```
roundF = \markup { \center-align \concat {
          \normal-text { \bold { \italic ( } }
          \dynamic f
          \normal-text { \bold { \italic ) } } } }
boxF = \markup { \bracket { \dynamic f } }
mfEspress = \markup { \center-align \line {
              \hspace #3.7 mf \normal-text \italic espress. } }
roundFdynamic = #(make-dynamic-script roundF)
boxFdynamic = #(make-dynamic-script boxF)
mfEspressDynamic = #(make-dynamic-script mfEspress)
\relative {
  c'4_\roundFdynamic\< d e f
  g,1~_\boxFdynamic\>
  g
  g'~\mfEspressDynamic
  g
}
```
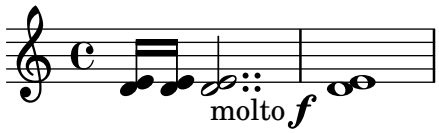


La construction d'une indication de nuance personnalisée peut aussi se faire en langage Scheme ; voir Section "Construction d'un markup en Scheme" dans *Extension de LilyPond* pour en connaître les modalités.

```
moltoF = #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF
}
```



L'utilisation d'un \tweak permettra d'aligner par la gauche cette nuance textuelle sur la tête de note, plutôt qu'un centrage :

```
moltoF = \tweak DynamicText.self-alignment-X #LEFT
         #(make-dynamic-script
            (markup #:normal-text "molto"
                    #:dynamic "f"))
\relative {
  <d' e>16 <d e>
  <d e>2..\moltoF <d e>1
}
```



L'utilisation des fontes en mode *markup* est abordée au chapitre Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

Voir aussi

Manuel de notation : Section 24.9 [Amélioration du rendu MIDI], page 683, Section 24.4 [Gestion des nuances en MIDI], page 674, Section 8.2 [Mise en forme du texte], page 329, Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Manuel d'extension : Section "Construction d'un markup en Scheme" dans *Extension de LilyPond*.

3.2 Signes d'interprétation sous forme de courbe
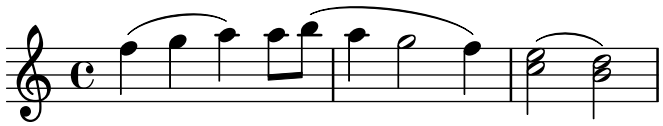
Ce chapitre traite des signes d'interprétation imprimés sous forme de courbe : liaisons d'articulation ou de phrasé, respirations, chutes et sauts.

3.2.1 Liaisons d'articulation

Une liaison d'articulation indique que les notes doivent être jouées liées, ou *legato*. Ces liaisons s'indiquent au moyen de parenthèses.

> **Note :** Lorsque la musique est polyphonique, la liaison doit se terminer dans la voix où elle a été entamée.
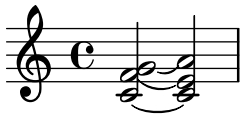
```
\relative {
  f''4( g a) a8 b(
  a4 g2 f4)
  <c e>2( <b d>2)
}
```



Vous pouvez décider de l'orientation des liaisons par rapport à la portée, comme indiqué au chapitre Section 36.1 [Direction et positionnement], page 801.
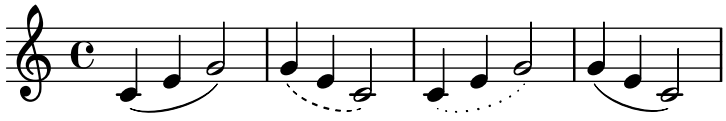
Plusieurs liaisons peuvent intervenir simultanément ou concurremment, ce qui requiert une attention spéciale. Dans la majorité des cas, la liaison externe indique un phrasé, et un phrasé peut recouvrir plusieurs liaisons d'articulation – voir Section 3.2.2 [Liaisons de phrasé], page 177. Dans le cas où plusieurs liaisons d'articulation interviennent au sein d'un même contexte `Voice`, leurs début et fin doivent être labellisés par un `\=` suivi d'un identifiant (symbole ou entier positif).

```
\fixed c' {
  <c~ f\=1( g\=2( >2 <c e\=1) a\=2) >
}
```
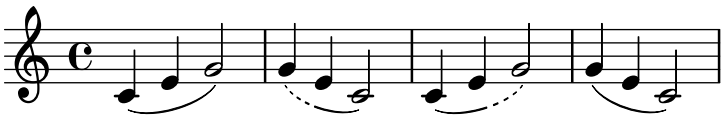


Une liaison est par défaut dessinée d'un trait plein. Il est aussi possible de l'imprimer sous la forme de tirets ou en pointillé :

```
\relative {
  c'4( e g2)
  \slurDashed
  g4( e c2)
  \slurDotted
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```
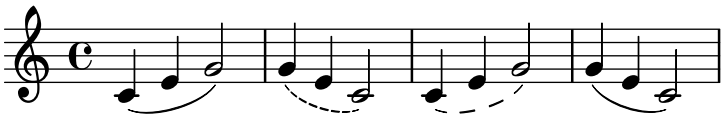


En utilisant `\slurHalfDashed`, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L'inverse s'obtient avec `\slurHalfSolid`.

```
\relative {
  c'4( e g2)
  \slurHalfDashed
  g4( e c2)
  \slurHalfSolid
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
\relative {
  c'4( e g2)
  \slurDashPattern 0.7 0.75
  g4( e c2)
  \slurDashPattern 0.5 2.0
  c4( e g2)
  \slurSolid
  g4( e c2)
}
```



Commandes prédéfinies

\slurUp, \slurDown, \slurNeutral, \slurDashed, \slurDotted, \slurHalfDashed, \slurHalfSolid, \slurDashPattern, \slurSolid.

Morceaux choisis

Ajustement du positionnement vertical d'une liaison

Il est possible, à l'aide d'un \override Slur.positions, de définir le positionnement vertical des extrémités d'une liaison à des valeurs absolues (ou, plutôt, forcer l'algoritme de LilyPond à considérer ces valeurs comme étant préférables). Dans de nombreux cas, ceci demande de procéder à tâtons pour obtenir une valeur acceptable, d'autant que la commande \offset est inefficace pour les liaisons et affiche un avertissement.

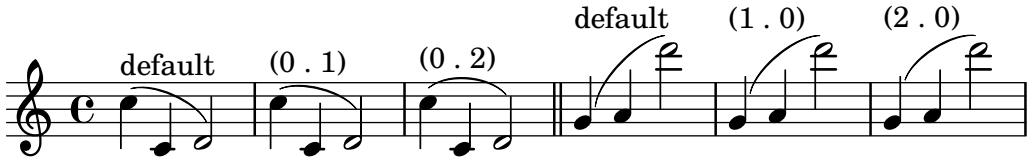
Le code ci-dessous parmet d'ajuster le positionnement du début et de la fin d'une liaison en spécifiant des modifications *relatives* comme le ferait un \offset.

Syntaxe : \offsetPositions #'(*dy1 . dy2*)

```
offsetPositions =
#(define-music-function (offsets) (number-pair?)
  #{
     \once \override Slur.control-points =
       #(lambda (grob)
          (match-let ((((_ . y1) _ _ (_ . y2))
                       (ly:slur::calc-control-points grob))
```

```
                    ((off1 . off2) offsets))
              (set! (ly:grob-property grob 'positions)
                    (cons (+ y1 off1) (+ y2 off2)))
              (ly:slur::calc-control-points grob)))
  #})

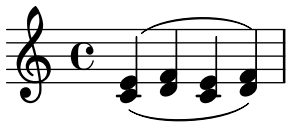
\relative c'' {
  c4(^"default" c, d2)
  \offsetPositions #'(0 . 1)
  c'4(^"(0 . 1)" c, d2)
  \offsetPositions #'(0 . 2)
  c'4(^"(0 . 2)" c, d2)
  \bar "||"
  g4(^"default" a d'2)
  \offsetPositions #'(1 . 0)
  g,,4(^"(1 . 0)" a d'2)
  \offsetPositions #'(2 . 0)
  g,,4(^"(2 . 0)" a d'2)
}
```



Accords et double liaison d'articulation

Certains compositeurs utilisent deux liaisons lorsqu'ils veulent lier des accords. Dans LilyPond, il faut pour cela activer la propriété de contexte `doubleSlurs`.

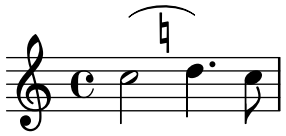
```
\relative c' {
  \set doubleSlurs = ##t
  <c e>4( <d f> <c e> <d f>)
}
```



Positionnement d'une annotation à l'intérieur d'une liaison

Lorsqu'une annotation doit s'incrire à l'intérieur d'une liaison, la propriété `outside-staff-priority` doit être désactivée.

```
\relative c'' {
  \override TextScript.avoid-slur = #'inside
  \override TextScript.outside-staff-priority = ##f
  c2(^\markup { \halign #-10 \natural } d4.) c8
}
```
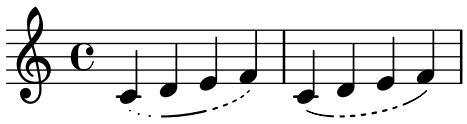


Dessin d'une liaison d'articulation au trait discontinu

Une liaison d'articulation peut être formée de traits discontinus variables. La propriété `dash-definition` se compose d'une liste de segments de liaison. Chaque segment contient une liste de paramètres qui déterminent le comportement du trait pour une section de la liaison.

Les segments de cette liaison se définissent selon le paramètre de Bézier `t` qui est compris entre 0 (l'extrémité gauche de la liaison) et 1 (l'extrémité droite de la liaison). Chaque segment se composera selon la liste (*t-début t-fin segment-style segment-taille*). La portion de liaison allant de *t-début* à *t-fin* aura un trait *segment-style* de longueur *segment-taille*. *segment-taille* est exprimé en espace de portée ; un *segment-style* à 1 donnera un trait plein. *segment-taille* s'exprime en espace de portée.

```
\relative c' {
  \once \override
    Slur.dash-definition = #'((  0  0.3  0.1 0.75)
                              (0.3  0.6  1   1   )
                              (0.65 1.0  0.4 0.75))
  c4( d e f)
  \once \override
    Slur.dash-definition = #'((0    0.25  1   1   )
                              (0.3  0.7   0.4 0.75)
                              (0.75 1.0   1   1   ))
  c4( d e f)
}
```



Voir aussi

Glossaire musicologique : Section "liaison" dans *Glossaire*.

Manuel d'initiation : Section "Non-imbrication des crochets et liaisons" dans *Manuel d'initiation*.

Manuel de notation : Section 36.1 [Direction et positionnement], page 801, Section 3.2.2 [Liaisons de phrasé], page 177.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Slur" dans *Référence des propriétés internes*.

3.2.2 Liaisons de phrasé

Une liaison de phrasé relie plusieurs notes en délimitant une phrase musicale. On indique les points de départ et d'arrivée avec `\(` et `\)` respectivement.

```
\relative {
  c''4\( d( e) f(
  e2) d\)
}
```
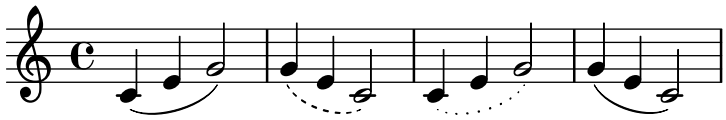


D'un point de vue typographique, rien ne distingue une liaison de phrasé d'une liaison d'articulation. Cependant, LilyPond les considère comme des objets différents. Une commande

\slurUp n'affectera donc pas une liaison de phrasé. Vous pouvez décider de l'orientation des liaisons de phrasé par rapport à la portée, comme indiqué au chapitre Section 36.1 [Direction et positionnement], page 801.

Plusieurs liaisons de phrasé peuvent intervenir en même temps, dès lors qu'elles sont labellisées, comme des liaisons normales – voir Section 3.2.1 [Liaisons d'articulation], page 174.
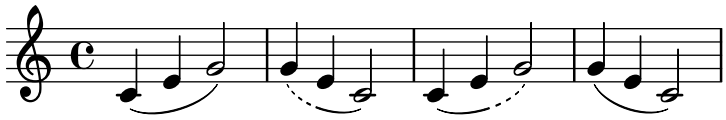
Une liaison est par défaut dessinée d'un trait plein. Il est aussi possible de l'imprimer sous la forme de tirets ou en pointillé :

```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashed
  g4\( e c2\)
  \phrasingSlurDotted
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```
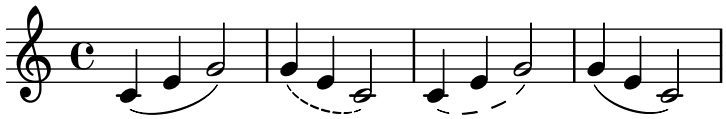


En utilisant \phrasingSlurHalfDashed, la première moitié de la liaison aura un trait discontinu et continu pour la seconde. L'inverse s'obtient avec \phrasingSlurHalfSolid.

```
\relative {
  c'4\( e g2\)
  \phrasingSlurHalfDashed
  g4\( e c2\)
  \phrasingSlurHalfSolid
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



Vous pouvez même personnaliser la densité des tirets d'une liaison :

```
\relative {
  c'4\( e g2\)
  \phrasingSlurDashPattern #0.7 #0.75
  g4\( e c2\)
  \phrasingSlurDashPattern #0.5 #2.0
  c4\( e g2\)
  \phrasingSlurSolid
  g4\( e c2\)
}
```



La personnalisation des lignes discontinues est identique pour les liaisons de phrasé et les liaisons d'articulation. Pour plus de détails, référez-vous aux morceaux choisis de la section Section 3.2.1 [Liaisons d'articulation], page 174.

Commandes prédéfinies

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`, `\phrasingSlurDashed`, `\phrasingSlurDotted`, `\phrasingSlurHalfDashed`, `\phrasingSlurHalfSolid`, `\phrasingSlurDashPattern`, `\phrasingSlurSolid`.

Voir aussi

Manuel d'initiation : Section "Non-imbrication des crochets et liaisons" dans *Manuel d'initiation*.

Manuel de notation : Section 36.1 [Direction et positionnement], page 801, Section 3.2.1 [Liaisons d'articulation], page 174.
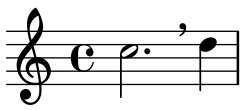
Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PhrasingSlur" dans *Référence des propriétés internes*.

3.2.3 Signes de respiration

La commande `\breathe` indique à l'interprète de raccourcir la note qui précède pour prendre une respiration..

```
\fixed c'' { c2. \breathe d4 }
```
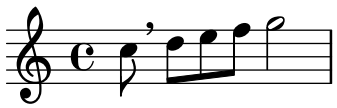


Pour une courte interruption du son qui ne soit pas prise sur la note précédente, voir Section 2.2.4 [Césures], page 77.

Contrairement aux autres signes d'interprétation, une respiration constitue un événement musical à part entière. Par voie de conséquence, toute autre marque attachée à la note précédente, telle un crochet indiquant une ligature manuelle ou une parenthèse indiquant une liaison, doit impérativement se placer avant l'instruction `\breathe`. L'instruction `\breathe` ne tolère pas d'articulation en soi – voir Section 2.2.4 [Césures], page 77.
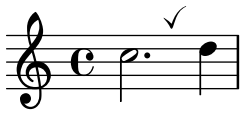
Un signe de respiration interrompt obligatoirement les ligatures automatiques. Pour passer outre ce fonctionnement, voir Section 2.4.3 [Barres de ligature manuelles], page 117.

```
\fixed c'' { c8 \breathe d e f g2 }
```



La propriété de contexte `breathMarkType` permet de contrôler laquelle des différentes indications de respiration la commande `\breathe` se chargera de créer – voir Section B.14 [Liste des marques de respiration], page 946.

```
\fixed c'' {
  \set breathMarkType = #'tickmark
  c2. \breathe d4
}
```



Voir aussi

Glossaire musicologique : Section "respiration" dans *Glossaire*.

Manuel de notation : Section 2.2.4 [Césures], page 77, Section 17.4.4 [Divisions], page 571.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "BreathingEvent" dans *Référence des propriétés internes*, Section "BreathingSign" dans *Référence des propriétés internes*, Section "Breathing_sign_engraver" dans *Référence des propriétés internes*.

3.2.4 Chutes et sauts

Des indications de désinence peuvent être obtenues au moyen de la commande `\bendAfter`. Leur direction s'indique au moyen des signes plus (vers le haut) ou moins (vers le bas). Le chiffre indique l'intervalle avec la note de départ.

```
\relative c'' {
  c2\bendAfter 4
  c2\bendAfter -4
  c2\bendAfter 6.5
  c2\bendAfter -6.5
  c2\bendAfter 8
  c2\bendAfter -8
}
```



Morceaux choisis

Ajustement du galbe des chutes ou sauts

La propriété `shortest-duration-space` peut devoir être retouchée pour ajuster l'apparence des chutes ou sauts.

```
\relative c'' {
  \override Score.SpacingSpanner.shortest-duration-space = 4.0
  c2-\bendAfter 5
  c2-\bendAfter -4.75
  c2-\bendAfter 8.5
  c2-\bendAfter -6
}
```



Voir aussi

Glossaire musical : Section "chute" dans *Glossaire*, Section "saut" dans *Glossaire*.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

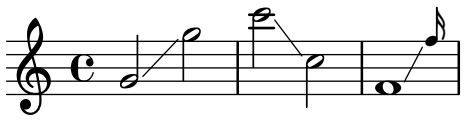
3.3 Signes d'interprétation sous forme de ligne

Cette partie traite de la manière de générer des signes d'interprétation d'aspect linéaire, tels les glissandos, arpèges et trilles.

3.3.1 Glissando

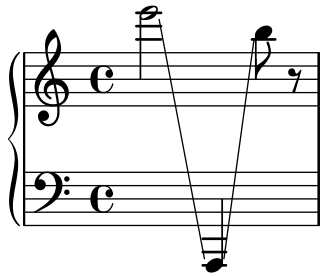
Un glissando relie une hauteur à une autre en passant par chaque hauteur intermédiaire. On l'obtient en accolant la commande `\glissando` à la première note.

```
\relative {
  g'2\glissando g'
  c2\glissando c,
  \afterGrace f,1\glissando f'16
}
```
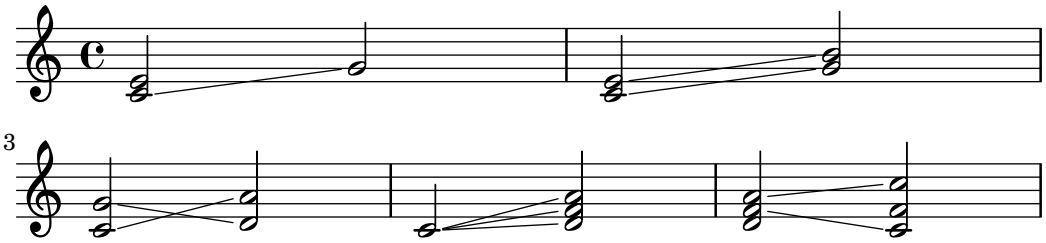


Un glissando peut intervenir au moment d'un changement de portée :

```
\new PianoStaff <<
  \new Staff = "right" {
    e'''2\glissando
    \change Staff = "left"
    a,,4\glissando
    \change Staff = "right"
    b''8 r |
  }
  \new Staff = "left" {
    \clef bass
    s1
  }
>>
```



Un glissando peut affecter des notes d'un accord. En dehors du cas où les notes des deux accords sont reliées directement l'une à l'autre, les relations s'établissent à l'aide d'une liste Scheme `glissandoMap` dont les éléments sont des paires d'entiers ; chaque paire *(x . y)* crée une ligne de glissando de la note *x* à la note *y* – les notes d'un accord sont numérotées à partir de zéro et dans leur ordre d'apparition dans le fichier `.ly`. Toutes les notes ne font pas forcément partie d'un glissando.

```
\relative {
  <c' e>2\glissando g'
  <c, e>\glissando <g' b>
  \break
  \set glissandoMap = #'((0 . 1) (1 . 0))
  <c, g'>\glissando <d a'>
  \set glissandoMap = #'((0 . 0) (0 . 1) (0 . 2))
  c\glissando <d f a>
  \set glissandoMap = #'((2 . 2) (0 . 0))
  <f d a'>\glissando <c f c'>
}
```
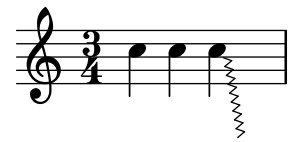


Un glissando est indiqué graphiquement, par une ligne ou des vaguelettes – voir Section 36.5 [Styles de ligne], page 809.

Morceaux choisis

Glissando contemporain

De nos jours, il peut arriver que la note d'arrivée d'un glissando soit absente de la partition. Pour ce faire, il vous faudra utiliser une cadence et « masquer » la note d'arrivée.

```
\relative c'' {
  \time 3/4
  \override Glissando.style = #'zigzag
  c4 c
  \cadenzaOn
  c4\glissando
  \hideNotes
  c,,4
  \unHideNotes
  \cadenzaOff
  \bar "|"
}
```



Ajout de marques temporelles à un long glissando
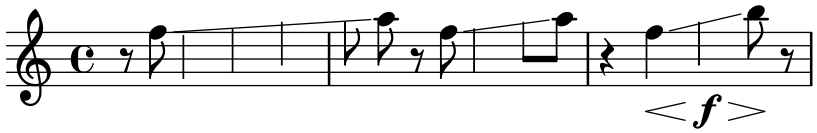
Lorsqu'un glissando s'étend dans la durée, on trouve parfois des indications temporelles, matérialisées par des hampes sans tête de note. De telles hampes permettent aussi d'indiquer des éléments intermédiaires.

L'alignement des hampes avec la ligne de glissando peut requérir quelques aménagements.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}

glissandoSkipOff = {
  \revert NoteColumn.glissando-skip
  \undo \hide NoteHead
  \revert NoteHead.no-ledgers
}

\relative c'' {
  r8 f8\glissando \glissandoSkipOn f4 g a |
  a8\noBeam \glissandoSkipOff a8
    r8 f8\glissando \glissandoSkipOn g4 a8 \glissandoSkipOff a8 |
  r4 f\glissando\< \glissandoSkipOn a4\f\> \glissandoSkipOff b8\! r |
}
```



Saut de ligne et glissando

Normalement, LilyPond refuse de sauter automatiquement à la ligne à l'endroit où un glissando se prolonge sur la mesure qui suit. Ce comportement est modifiable par l'affectation de la valeur `#t` à la propriété `Glissando.breakable`. L'activation de la propriété `after-line-breaking` permet la continuation de l'indication de glissando après le saut de ligne.

La propriété `breakable` n'affecte pas les sauts manuels insérés par les commandes comme `\break`.

```
glissandoSkipOn = {
  \override NoteColumn.glissando-skip = ##t
  \hide NoteHead
  \override NoteHead.no-ledgers = ##t
}
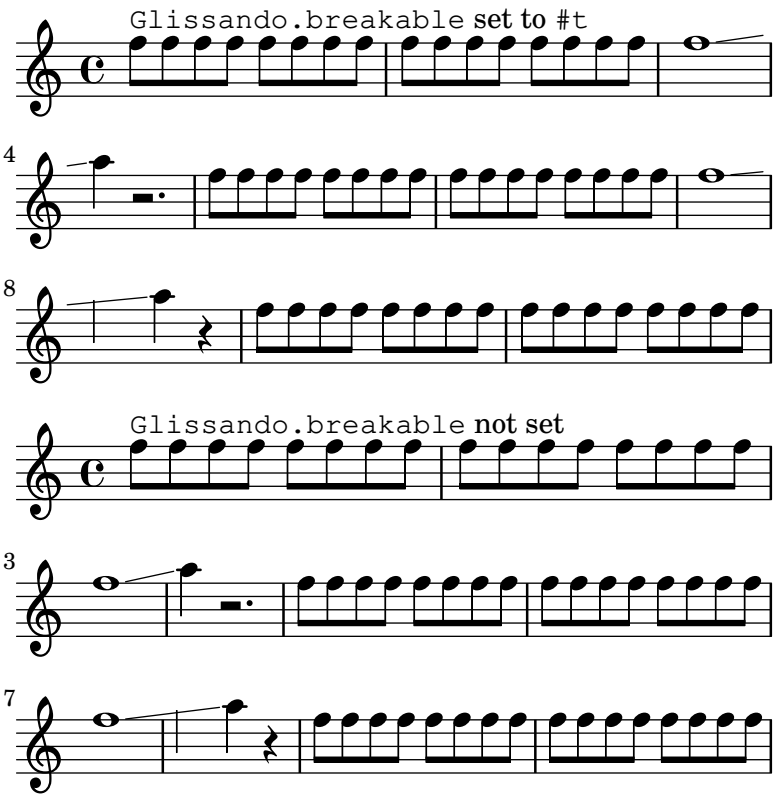
music = {
  \*16 f8 |
  f1\glissando |
  a4 r2. |
  \*16 f8 |
  f1\glissando \once\glissandoSkipOn |
  a2 a4 r4 |
  \*16 f8
}

\relative c'' {
  <>^\markup { \typewriter Glissando.breakable
               set to \typewriter "#t" }
  \override Glissando.breakable = ##t
  \override Glissando.after-line-breaking = ##t
```

```
  \music
}

\relative c'' {
  <>^\markup { \typewriter Glissando.breakable not set }
  \music
}

\paper {
  line-width = 100\mm
}
```



Rappel du glissando à l'occasion d'une alternative

Un glissando qui se prolonge sur plusieurs sections `\alternative` peut se rappeler à l'aide d'une note d'ornement supplémentaire et masquée, à laquelle sera attaché le départ du glissando, ce dans chaque bloc `\alternative`. Cette note d'ornement devrait avoir la même hauteur que la note où commençait le glissando originel. Ceci est géré par une fonction musicale qui prendra en argument la hauteur de la note d'ornement.

Dans le cadre d'une musique polyphonique, il ne faudra pas oublier d'ajouter une note d'ornement dans toutes les autres voix afin de préserver la synchronisation.

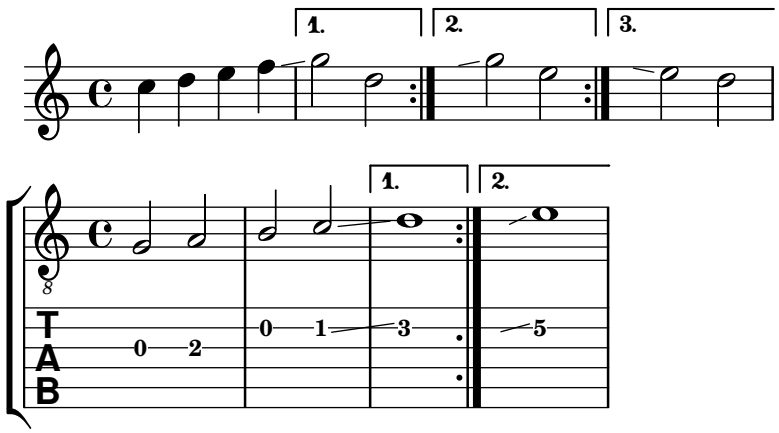
```
repeatGliss = #(define-music-function (grace)
  (ly:pitch?)
  #{
    % the next two lines ensure the glissando is long enough
    % to be visible
    \once \override Glissando.springs-and-rods
      = #ly:spanner::set-spacing-rods
    \once \override Glissando.minimum-length = 3.5
```

```
    \once \hideNotes
    \grace $grace \glissando
  #})

\score {
  \relative c'' {
    \repeat volta 3 { c4 d e f\glissando }
    \alternative {
      { g2 d }
      { \repeatGliss f g2 e }
      { \repeatGliss f e2 d }
    }
  }
}

music =  \relative c' {
  \voiceOne
  \repeat volta 2 {
    g a b c\glissando
  }
  \alternative {
    { d1 }
    { \repeatGliss c \once \omit StringNumber e1\2 }
  }
}

\score {
  \new StaffGroup <<
    \new Staff <<
      \new Voice { \clef "G_8" \music }
    >>
    \new TabStaff  <<
      \new TabVoice { \clef "moderntab" \music }
    >>
  >>
}
```



Voir aussi

Glossaire musicologique : Section "glissando" dans *Glossaire*.

Manuel de notation : Section 36.5 [Styles de ligne], page 809.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Glissando" dans *Référence des propriétés internes*.

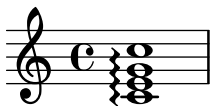
Problèmes connus et avertissements

Il n'est pas possible d'imprimer un texte (tel que *gliss.*) le long de la ligne de glissando.

3.3.2 Arpèges

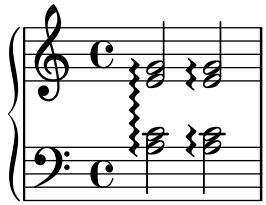
On peut indiquer qu'un accord doit être arpégé en lui accolant la commande `\arpeggio` :

```
\relative { <c' e g c>1\arpeggio }
```



Il est possible de distribuer un arpège sur plusieurs voix ou portées en activant la propriété `connectArpeggios` dans un contexte de regroupement de portées. Ceci aura pour résultat que tous les arpéges de ce contexte seront connectés :

```
\new PianoStaff \fixed c' <<
  \new Staff {
    \once \set PianoStaff.connectArpeggios = ##t
    <e g>2\arpeggio q\arpeggio
  }
  \new Staff {
    \clef "bass"
    <a, c>2\arpeggio q\arpeggio
  }
>>
```
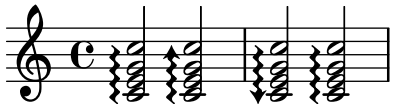


LilyPond dispose de différents graphismes pour indiquer l'exécution d'un arpège ; `\arpeggioNormal` reviendra au style par défaut.

```
\relative {
  <c' e g c>2\arpeggio

  \arpeggioArrowUp
  <c e g c>2\arpeggio

  \arpeggioArrowDown
  <c e g c>2\arpeggio

  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Ces commandes prédéfinies modifient en interne la propriété `arpeggio-direction` ; leur définition complète se trouve dans le fichier `ly/property-init.ly`.

Le symbole créé par la commande `\arpeggio` peut se modifier :

```
\relative {
  <c' e g c>2

  \arpeggioBracket
  <c e g c>2\arpeggio

  \arpeggioParenthesis
  <c e g c>2\arpeggio

  \arpeggioParenthesisDashed
  <c e g c>2\arpeggio
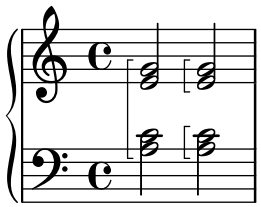
  \arpeggioNormal
  <c e g c>2\arpeggio
}
```



Ces commandes prédéfinies apportent une dérogation à la propriété `stencil` de l'objet `Arpeggio` et peuvent aussi adapter son `X-extent` (en la matière son étendue horizontale, de telle sorte qu'il n'entre pas en collision avec d'autres objets).

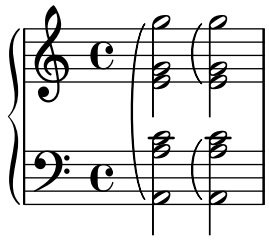
Pour un crochet indiquant qu'un accord ne doit pas être arpégé, mieux vaut recourir à la commande `\nonArpeggiato` que de modifier l'apparence rendue par `\arpeggio`. `\nonArpeggiato` crée un objet `ChordBracket`.

```
\new PianoStaff \fixed c' <<
  \new Staff {
    \once \set PianoStaff.connectChordBrackets = ##t
    <e g>2\nonArpeggiato q\nonArpeggiato
  }
  \new Staff {
    \clef "bass"
    <a, c>2\nonArpeggiato q\nonArpeggiato
  }
>>
```



Pour une liaison verticale indiquant qu'un accord n'est presque pas arpégé, mieux vaut recourir à la commande `\chordSlur` que de modifier l'apparence rendue par `\arpeggio`. `\chordSlur` crée un objet `ChordSlur`.

```
\new PianoStaff \fixed c <<
  \new Staff {
    \once \set PianoStaff.connectChordSlurs = ##t
    <e' g' g''>2\chordSlur q\chordSlur
  }
  \new Staff {
    \clef "bass"
    <a, a c'>2\chordSlur q\chordSlur
  }
>>
```



Les indications d'arpeggio peuvent se présenter sous la forme de ligne discontinue à l'aide de la propriété `dash-definition`. Pour plus de détails à ce propos, consultez Section 3.2.1 [Liaisons d'articulation], page 174.

Un arpège peut parfois s'écrire de manière explicite, à l'aide de liaisons de tenue. Pour plus d'information, voir Section 2.1.4 [Liaisons de prolongation], page 64.

Commandes prédéfinies
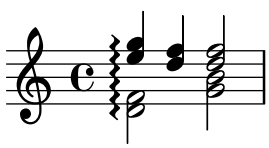
`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`, `\arpeggioParenthesis` `\arpeggioParenthesisDashed`, `\chordSlur`, `\nonArpeggiato`.

Morceaux choisis

Arpège distribué sur plusieurs voix

Affecter le graveur `Span_arpeggio_engraver` au contexte de la portée (`Staff`) permet de distribuer un arpège sur plusieurs voix.

```
\new Staff \with {
  \consists "Span_arpeggio_engraver"
}
\relative c' {
  \set Staff.connectArpeggios = ##t
  <<
    { <e' g>4\arpeggio <d f> <d f>2 }
    \\
    { <d, f>2\arpeggio <g b>2 }
  >>
}
```



Voir aussi

Glossaire musicologique : Section "arpeggio" dans *Glossaire*.

Manuel de notation : Section 3.2.1 [Liaisons d'articulation], page 174, Section 2.1.4 [Liaisons de prolongation], page 64.

Fichiers d'initialisation : `ly/property-init.ly`.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Arpeggio" dans *Référence des propriétés internes*, Section "ChordBracket" dans *Référence des propriétés internes*, Section "ChordSlur" dans *Référence des propriétés internes*, Section "PianoStaff" dans *Référence des propriétés internes*, Section "Slur" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les commandes prédéfinies telle que `\arpeggioArrowUp` s'appliquent uniquement au contexte en cours ; elles n'affecteront donc pas un arpège s'étendant sur d'autres voix ou portées. En pareil cas, ces commandes demandent à être utilisées au sein d'une section `\context` dans le bloc `\layout` ou bien avec une clause `\with` comme indiqué dans Section 33.5 [Modification des réglages par défaut d'un contexte], page 771. Il peut être judicieux, au lieu d'utiliser ces raccourcis prédéfinis, d'introduire une dérogation aux propriétés concernées de l'objet `Arpeggio` directement dans le contexte approprié, comme par exemple

```
\override Staff.Arpeggio.stencil = #ly:chord-bracket::print
\override Staff.Arpeggio.X-extent = #ly:chord-bracket::width
```

pour que le crochet d'arpège couvre plusieurs voix au niveau `Staff`, ou

```
\override PianoStaff.Arpeggio.arpeggio-direction = #UP
```

pour obtenir un arpège à terminaison en flèche (pointant vers le haut) couvrant les portées d'un contexte `PianoStaff`.

Il est impossible de mêler au même instant, dans un contexte `PianoStaff`, des lignes d'arpèges connectées et d'autres non connectées.

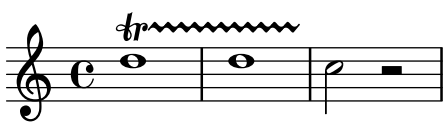
La manière simple de créer des lignes d'arpège sous forme de parenthèse n'est pas opérationnelle pour des arpèges inter-portées ; voir [Cross-staff chord brackets], page 434.

3.3.3 Trilles

Les trilles brefs s'indiquent comme n'importe quelle ponctuation, avec un simple `\trill` ; voir Section 3.1.1 [Articulations et ornements], page 158.
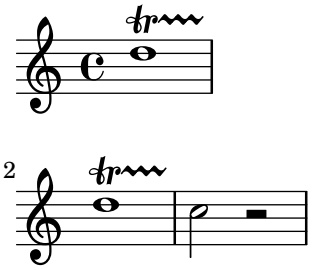
Les trilles plus longs sont délimités par `\startTrillSpan` et `\stopTrillSpan` :

```
\relative {
  d''1\startTrillSpan
  d1
  c2\stopTrillSpan r2
}
```



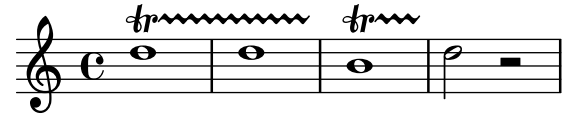
Lorsqu'un saut de ligne intervient alors qu'une prolongation de trille est présente, l'indication de trille et sa prolongation sont rappelées sur la première note de la nouvelle ligne :

```
\relative {
  d''1\startTrillSpan
  \break
  d1
  c2\stopTrillSpan r2
}
```
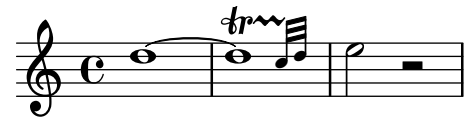


Lorsque des trilles interviennent sur une succession de hauteurs différentes, point n'est besoin d'expliciter la commande `\stopTrillSpan` puisque l'apparition d'un nouveau trille interrompt de fait celui qui le précédait :

```
\relative {
  d''1\startTrillSpan
  d1
  b1\startTrillSpan
  d2\stopTrillSpan r2
}
```



Dans l'exemple suivant, un trille se combine avec des notes d'ornement. La syntaxe d'une telle construction ainsi que le moyen de positionner les notes d'ornement avec précision est expliquée au chapitre Section 2.6.1 [Notes d'ornement], page 150.

```
\relative {
  d''1~\afterGrace
  d1\startTrillSpan { c32[ d]\stopTrillSpan }
  e2 r2
}
```



Les trilles qui font intervenir une hauteur précise peuvent être indiqués par la commande `pitchedTrill`. Le premier argument est la note réelle ; le second est une hauteur qui sera imprimée comme une tête de note noire entre parenthèses.

```
\relative {
  \pitchedTrill
  e''2\startTrillSpan fis
  d2 c2\stopTrillSpan
}
```
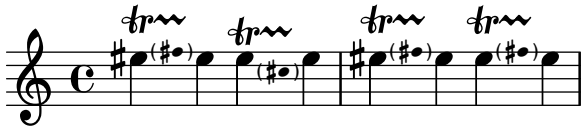


L'altération de cette hauteur explicite sera indiquée automatique pour le premier trille d'une mesure, même s'il s'agit d'un bécarre.

```
{
  \key d \major
  \pitchedTrill
  d'2\startTrillSpan cis d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan c d\stopTrillSpan
  \pitchedTrill
  d2\startTrillSpan e d\stopTrillSpan
}
```



L'impression de l'altération (sur la même note dans la même mesure) devra être forcée en ajoutant un `!` à la note considérée.

```
\relative {
  \pitchedTrill
  eis''4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan cis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis
  eis4\stopTrillSpan
  \pitchedTrill
  eis4\startTrillSpan fis!
  eis4\stopTrillSpan
}
```



Commandes prédéfinies

`\startTrillSpan`, `\stopTrillSpan`.

Voir aussi
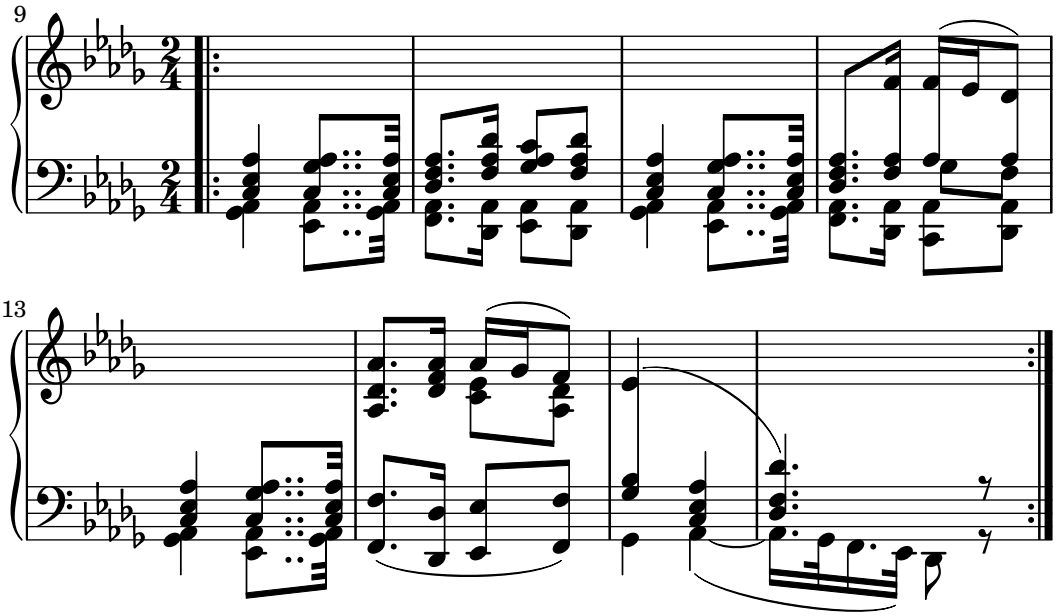
Glossaire musicologique : Section "trille" dans *Glossaire*.

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Section 2.6.1 [Notes d'ornement], page 150.

Morceaux choisis : Section "Signes d'interprétation" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TrillSpanner" dans *Référence des propriétés internes*, Section "TrillPitchHead" dans *Référence des propriétés internes*, Section "TrillPitchAccidental" dans *Référence des propriétés internes*, Section "TrillPitchParentheses" dans *Référence des propriétés internes*, Section "TrillPitchGroup" dans *Référence des propriétés internes*, Section "Pitched_trill_engraver" dans *Référence des propriétés internes*.

4 Répétitions et reprises



La répétition est une notion essentielle en musique, et il existe de nombreuses façons de mettre en œuvre et noter ce concept. LilyPond prend en charge les types de répétition suivants :

`volta` Il s'agit ici de la notation courante des reprises avec ou sans fins alternatives. Le passage répété est encadré par des barres de reprise. Lorsque la répétition commence au début de la pièce, aucune barre de reprise n'est gravée en début de partition. Les fins alternatives s'impriment en séquence, avec un crochet et un numéro de *volta*.

`segno` Ceci permet de gérer différentes formes de *da capo* et *dal segno*. Le fragment repris débute par une indication de segno hormis dans le cas où il s'agit du début du morceau. Les fins alternatives apparaissent en séquence, indiquées par des marques de coda, et un libellé de section, tel que « Coda », peut éventuellement apparaître avec la dernière alternative. Le fragment répété se termine par une instruction telle que *D.S.*

`unfold` La musique répétée est développée dans la partition autant de fois qu'indiqué.

`percent` Des barres obliques ou signes de pourcentage indiquent la répétition de temps ou de mesures.

`tremolo` Ce type permet de réaliser des trémolos sous forme de liens de croches.

Les constructions d'accord peuvent se répéter à l'aide du symbole spécifique `q` – voir Section 5.1.2 [Répétition d'accords], page 223.

4.1 Répétition d'un long passage
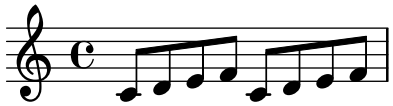
Cette section présente la syntaxe des répétitions longues.

4.1.1 Répétitions explicites

Adjoindre à la commande `\repeat` l'option `unfold` permet de s'affranchir de ressaisir de la musique répétitive. La syntaxe est identique à celle des commandes `\repeat volta` et `\repeat segno` abordées plus avant.

Le raccourci `\*` se révèle pratique dans une œuvre qui utilise beaucoup de musique répétitive :

```
\*2 { c'8 d' e' f' }
```



Afin d'éviter les redites, l'expansion n'est ici pas abordée en détail. Certains exemples des paragraphes qui suivent illustrent différentes fonctionnalités de la commande `\unfoldRepeats` qui permet de convertir les formes `volta` ou `segno` en une forme `unfold`. Un autre sujet d'intérêt de la commande `unfoldRepeats` est abordé dans Section 24.6 [Gestion des répétitions en MIDI], page 679.

Certains aspects de la commande `\repeat unfold` méritent des précisions.

Dans certains cas, et tout particulièrement dans un contexte `\relative`, la fonction `\repeat unfold` ne revient pas à écrire littéralement la même expression musicale plusieurs fois. Ainsi :

```
\repeat unfold 2 { a'4 b c }
```

introduit un saut d'octave, contrairement à

```
a'4 b c |
a'4 b c |
```

Vous pouvez facilement imbriquer plusieurs fonctions `\repeat unfold`, ce qui n'est pas aussi simple avec `\repeat volta` ou `\repeat segno`.

> **Note :** L'insertion d'un `\relative` dans une section `\repeat` sans déclaration explicite du contexte `Voice` générera une portée supplémentaire – voir Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Voir aussi

Morceaux choisis : Section "Répétitions" dans *Morceaux choisis*.

Référence des propriétés internes : Section "UnfoldedRepeatedMusic" dans *Référence des propriétés internes*.
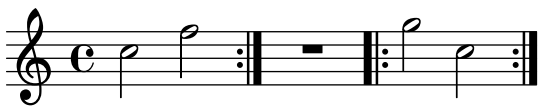
4.1.2 Répétitions simples

Les reprises courantes, sans alternative, s'indiquent comme ceci :

```
\repeat volta nombre_de_fois expression_musicale
```
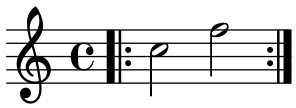
où *expression_musicale* représente ce qui doît être répété.

```
\fixed c'' {
  \repeat volta 2 { c2 f }
  R1
  \repeat volta 2 { g2 c }
}
```
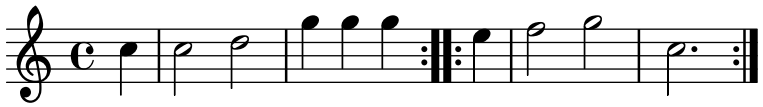


Aucun « début » de reprise n'est indiqué par défaut pour la première mesure d'un morceau selon la tradition des graveurs. Néanmoins, et dans certains contextes tels que les standards du jazz, elle peut apparaître grâce à la propriété `printInitialRepeatBar`.

```
\fixed c'' {
  \set Score.printInitialRepeatBar = ##t
  \repeat volta 2 { c2 f }
}
```



Lorsqu'une reprise sans fin alternative débute au milieu d'une mesure, elle devrait se terminer aussi au milieu d'une mesure, de telle sorte que les mesures soient complètes. En pareil cas, les indications de reprise ne constituent pas des barres de mesure à proprement parler ; il n'est donc pas nécessaire de faire appel à la commande \partial ou à des contrôles d'intégrité de mesure. Le recours à l'instruction \partial est toutefois nécessaire lorsque la reprise est au début d'une pièce commençant par une levée afin de respecter cette entame.

```
\fixed c'' {
  \partial 4
  \repeat volta 2 {
    c4
    c2 d
    g4 g g
  }
  \repeat volta 2 {
    e4
    f2 g
    c2.
  }
}
```
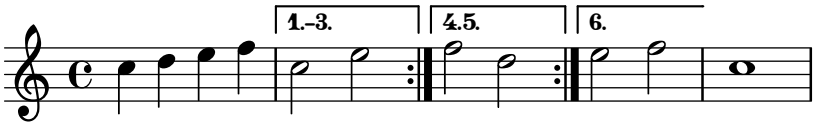


4.1.3 Fins alternatives

Une répétition avec fins alternatives peut s'indiquer de deux manières différentes. La syntaxe préférentielle est la suivante :

```
\repeat volta nombre_de_fois {
  expression_musicale
  \alternative {
    \volta liste_de_nombres expression_musicale
    \volta liste_de_nombres expression_musicale
    ...
  }
}
```

où *liste_de_nombres* est une liste de nombres, séparés par des virgules, indiquant les occurrences à répéter, de 1 à *nombre_de_fois*.

```
\fixed c'' {
  \repeat volta 6 {
    c4 d e f
    \alternative {
      \volta 1,2,3 { c2 e }
      \volta 4,5 { f2 d }
      \volta 6 { e2 f }
    }
  }
  c1
}
```
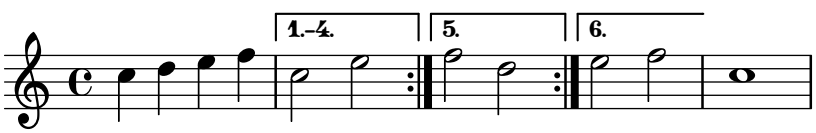


Une autre syntaxe, plus ancienne et toujours prise en charge, consiste à instancier le bloc `\alternative` à la suite de l'expression musicale répétée. Elle produit les mêmes effets. Cette syntaxe requiert toutefois que `\repeat` soit littéralement présent : un `\alternative` final ne se rattache pas à une fonction musicale ou une variable contenant `\repeat`.

```
\repeat volta nombre_de_fois expression_musicale
\alternative {
  \volta liste_de_nombres expression_musicale
  \volta liste_de_nombres expression_musicale
  ...
}
```

Les spécifications de `\volta` sont toutes au sein du bloc `\alternative`, soit il n'y en a pas. Si l'on donne trop peu d'alternatives en regard du nombre de fois où le passage doit être rejoué, la première alternative sera jouée plusieurs fois.

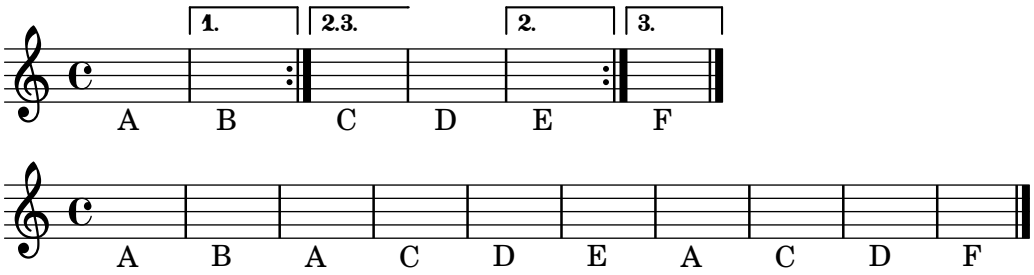
```
\fixed c'' {
  \repeat volta 6 {
    c4 d e f
    \alternative {
      { c2 e }
      { f2 d }
      { e2 f }
    }
  }
  c1
}
```



Les blocs `\alternative` peuvent s'imbriquer.

```
music = \fixed c' {
  \repeat volta 3 {
    s1_"A"
    \alternative {
      \volta 1 { s1_"B" }
      \volta 2,3 {
        \once \override Score.VoltaBracket.musical-length =
          \musicLength 1
        s1_"C"
        s1_"D"
        \alternative {
          \volta 2 { s1_"E" }
          \volta 3 { s1_"F" }
        }
      }
    }
  }
  \fine
}

\score { \music }
\score { \unfoldRepeats \music }
```



Note : Tout élément inclus dans un bloc `\alternative` sera traité en tant qu'alternative. Quelque chose d'aussi simple qu'un contrôle de mesure placé du mauvais côté d'une accolade peut générer des résultats inattendus.

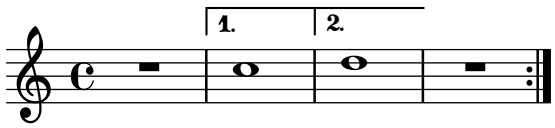
Note : Une clause `\relative` ne doit jamais se trouver à l'intérieur d'une section `\repeat` : vous aurez immanquablement des portées parasites. Voir Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Note : Lorsque la numérotation des mesures est activée au sein des alternatives, elle s'applique sur l'ensemble de ces alternatives.

4.1.4 Autres aspects des fragments répétés

Un bloc `\alternative` peut prendre place à l'intérieur d'un bloc `\repeat` afin de produire une notation comparable à des alternatives – voir Section 4.1.3 [Fins alternatives], page 195.

```
\fixed c'' {
  \repeat volta 2 {
    R1
    \alternative {
      \volta 1 { c1 }
      \volta 2 { d1 }
    }
    R1
  }
}
```

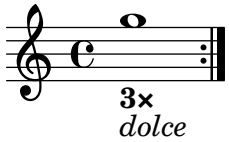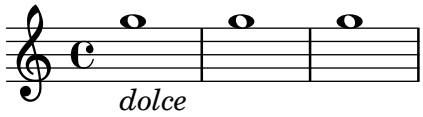


La présence d'une commande \volta n'est pas réservée aux éléments d'un bloc \alternative. Elle peut s'utiliser n'importe où dans un bloc \repeat pour désigner un fragment qui concerne des reprises particulières, tout en ne créant pas de crochet pour les autres cas.

Lorsqu'un bloc \repeat est expansé, le fragment spécifique à une occurrence sera omis pour toutes celles auxquelles il ne s'applique pas. Fournir une liste Scheme vide au lieu du nombre de fois supprime carrément la musique.

```
music = \repeat volta 3 {
  \volta #'() { <>_\markup { \bold "3×" } }
  \volta 1 { <>_\markup { \italic dolce } }
  g''1
}

\score { \music }
\score { \unfoldRepeats \music }
```



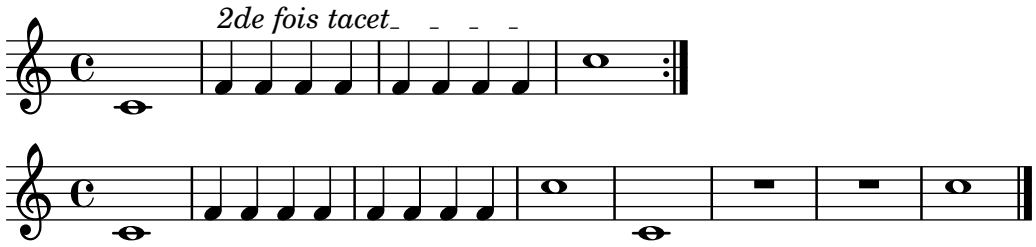


Lorsqu'un bloc \repeat est expansé, il peut être souhaitable de non seulement filtrer les fragments spécifiques, mais aussi d'ajouter de la musique qui serait absente de la forme « repliée ». La commande \unfolded permet de désigner la musique qui sera ignorée jusqu'à l'expansion du bloc \repeat qui la contient.

```
music = \fixed c' {
  \repeat volta 2 {
    c1
    <<
      \volta #'() {
        \once \override TextSpanner.bound-details.left.text =
          "2de fois tacet"
        s4*7\startTextSpan s4\stopTextSpan
      }
```

```
      \volta 1 { f4 f f f | f f f f }
      \volta 2 { \unfolded { R1*2 } }
    >>
    c'1
    \volta 2 { \unfolded { \bar "|." } }
  }
  \fine
}

\score { \music }
\score { \unfoldRepeats \music }
```



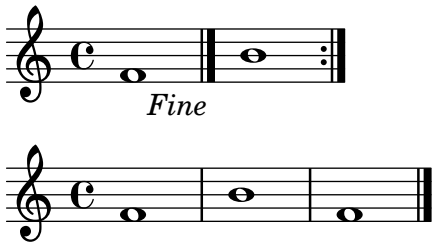
Note : Les commandes `\volta` et `\unfolded` fonctionnent en respectant la répétition la plus imbriquée qui les entoure.

4.1.5 Reprises *Al-fine*

La commande `\fine` met un point final à la musique, sans toutefois le forcer. Lorsqu'une reprise contenant `\fine` est expansée, la commande `\fine` est déroulée avec le reste de la musique et interviendra là où elle doit se produire dans la musique expansée. Il est donc nécessaire de bien spécifier à quelle occurrence le *Fine* doit se réaliser – voir Section 4.1.4 [Autres aspects des fragments répétés], page 197.

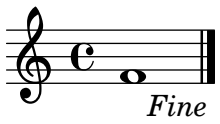
```
music = \fixed c' {
  \repeat volta 2 {
    f1
    \volta 2 \fine
    \volta 1 b1
  }
}

\score { \music }
\score { \unfoldRepeats \music }
```
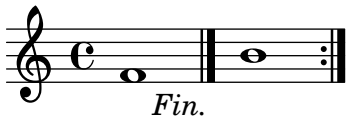


Comme illustré ci-dessus, `\fine` crée, à la toute fin de la musique, une barre finale sans instruction *Fine*. Il est possible de forcer l'impression de ce *Fine* en activant la propriété de contexte `finalFineTextVisibility`.

```
\fixed c' {
  \set Score.finalFineTextVisibility = ##t
  f1
  \fine
}
```



Le texte affiché par la commande `\fine` est modifiable à l'aide de la propriété de contexte `fineText`.

```
\fixed c' {
  \set Score.fineText = "Fin."
  \repeat volta 2 {
    f1
    \volta 2 \fine
    \volta 1 b1
  }
}
```



Pour de plus amples détails concernant les interactions avec les autres barres de mesure et les options pour modifier leur apparence, voir Section 2.5.2 [Barres de mesure automatiques], page 133.

Voir aussi

Glossaire musicologique : Section “fine” dans *Glossaire*.

Manuel de notation : Section 4.1.4 [Autres aspects des fragments répétés], page 197, Section 2.5.2 [Barres de mesure automatiques], page 133, Section 2.5.7 [Division en sections], page 150.

Morceaux choisis : Section “Reprises” dans *Morceaux choisis*.

Référence des propriétés internes : Section “FineEvent” dans *Référence des propriétés internes*, Section “Jump_engraver” dans *Référence des propriétés internes*, Section “JumpScript” dans *Référence des propriétés internes*.

4.1.6 Structure d'une reprise *Segno*

Les commandes `\repeat segno` et `\repeat volta` ne diffèrent que par la notation qu'il en résultera. Si les sections précédentes contiennent des informations d'ordre général quant à la manière de saisir de la musique répétée, des alternatives et de différentes variations, les paragraphes qui suivent couvrent les particularités liées à la notation avec *segno* sans reprendre tout ce qui a trait à la syntaxe de saisie.

`\repeat segno` permet de noter des reprises comportant des instructions *D.C.* ou *D.S.* dans leur présentation. Il s'agit d'indiquer par le signe *segno* le début d'un fragment répété lorsqu'il n'est pas en début de pièce. D'autre part, les fins alternatives seront indiquées par un signe *coda* en lieu et place des crochets d'alternative, si tant est que celles-ci soient exécutées en séquence – autrement dit pas dans le cas de `\volta 1,3` puis `\volta 2,4`.

Lorsque la numérotation des mesures est activée pour les alternatives, elle s'applique en présence de crochets, qu'il soient créés par `\repeat segno` ou `\repeat volta`, mais pas en présence d'indications de coda.

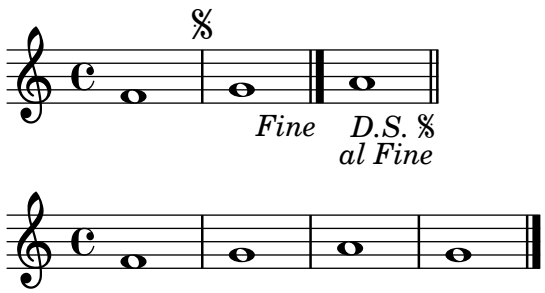
Les cas habituels d'utilisation sont les suivants.

al fine

Les instructions de reprise comportent un *al Fine* si la commande `\fine` a déjà été libellée – voir Section 4.1.5 [Reprises *Al-fine*], page 199.

```
music = \fixed c' {
  f1
  \repeat segno 2 {
    g1
    \volta 2 \fine
    \volta 1 a1
  }
  \section
}

\score { \music }
\score { \unfoldRepeats \music }
```
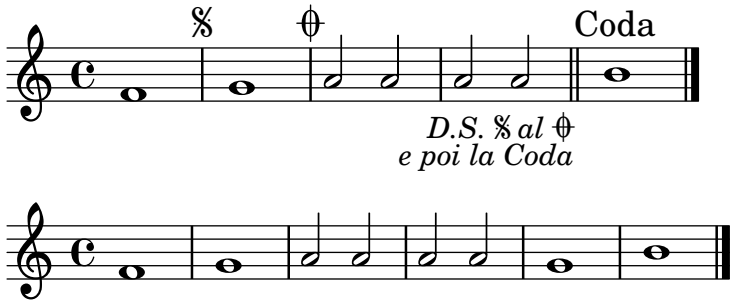


alla coda

Le début de chaque alternative est implicitement marqué d'un `\codaMark \default`. Les instructions de reprise des alternatives incluent « *al . . . e poi la . . .* » renvoyant à la marque de la première alternative et au repère où se reporter. Si tant est que la durée de la dernière alternative est nulle, le repère automatique est supprimé, ce qui permet de le remplacer par un libellé de section.

```
music = \fixed c' {
  f1
  \repeat segno 2 {
    g1
    \alternative {
      \volta 1 { \*4 a2 }
      \volta 2 \volta #'() {
        \section
        \sectionLabel "Coda"
      }
    }
  }
  b1
  \fine
}
```

```
\score { \music }
\score { \unfoldRepeats \music }
```



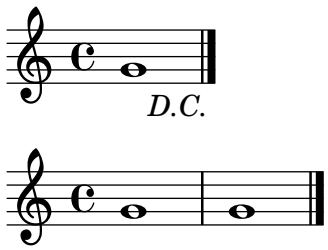
L'instruction de retour peut s'abréger par la définition d'une procédure de mise en forme alternative – voir Section 4.1.7 [Apparence d'une reprise *Segno*], page 204.

da capo

Les instructions de reprise incluent *D.C.* lorsque le fragment à répéter est en début de partition. Les cas d'emploi de *da capo* suivent le même principe que les cas de *dal segno*.

```
music = \fixed c' {
  \repeat segno 2 {
    g1
  }
  \fine
}

\score { \music }
\score { \unfoldRepeats \music }
```



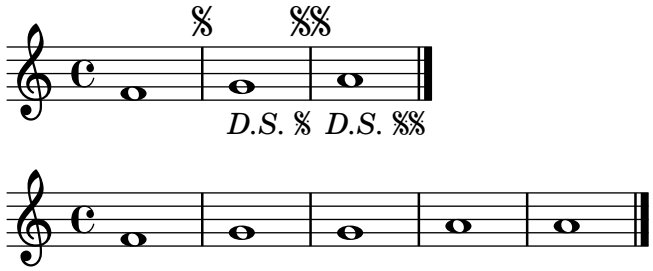
dal segno

Les instructions de reprise incluent *D.S.* lorsque le fragment à répéter n'est pas en début de partition. Le début du fragment répété est implicitement marqué d'un `\segnoMark \default`.

```
music = \fixed c' {
  f1
  \repeat segno 2 {
    g1
  }
  \repeat segno 2 {
    a1
  }
  \fine
```

```
}

\score { \music }
\score { \unfoldRepeats \music }
```



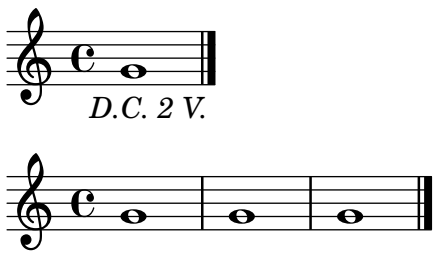
Il est possible de forcer une reprise *dal-segno* en début de partition – voir Section 4.1.7 [Apparence d'une reprise *Segno*], page 204.

retour multiple

L'instruction pour une reprise qui doit être exécutée plus d'une fois inclue « *. . . V.* ». Le passage concerné devra être joué une fois de plus que le nombre indiqué par l'instruction.

```
music = \fixed c' {
  \repeat segno 3 {
    g1
  }
  \fine
}

\score { \music }
\score { \unfoldRepeats \music }
```



Voir aussi

Glossaire musicologique : Section "da capo" dans *Glossaire*, Section "dal segno" dans *Glossaire*, Section "fine" dans *Glossaire*.

Manuel de notation : Section 4.1.4 [Autres aspects des fragments répétés], page 197, Section 2.5.2 [Barres de mesure automatiques], page 133, Section 2.5.7 [Division en sections], page 150, Section 4.1.3 [Fins alternatives], page 195, Section 8.1.4 [Marque de section], page 323, Section 4.1.2 [Répétitions simples], page 194, Section 4.1.5 [Reprises *Al-fine*], page 199.

Morceaux choisis : Section "Répétitions et reprises" dans *Morceaux choisis*.

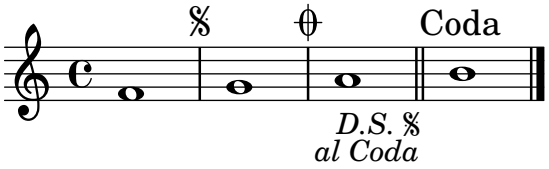
Référence des propriétés internes : Section "CodaMark" dans *Référence des propriétés internes*, Section "JumpScript" dans *Référence des propriétés internes*, Section "SectionLabel" dans *Référence des propriétés internes*, Section "SegnoMark" dans *Référence des propriétés internes*.

4.1.7 Apparence d'une reprise *Segno*

Les marques et instructions de retour créées par `\repeat segno` peuvent bénéficier de quelques adaptations.
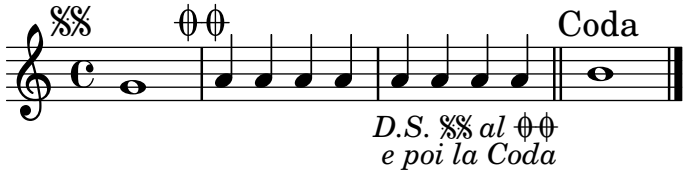
Les instructions de retour sont formatées par la procédure Scheme spécifiée dans la propriété `dalSegnoTextFormatter`. Un formateur prédéfini permet d'obtenir des instructions raccourcies.

```
\fixed c' {
  \set Score.dalSegnoTextFormatter = #format-dal-segno-text-brief
  f1
  \repeat segno 2 {
    g1
    \alternative {
      \volta 1 { a1 }
      \volta 2 \volta #'() {
        \section
        \sectionLabel "Coda"
      }
    }
  }
  b1
  \fine
}
```



Les numéros de séquence des marques en début de fragment répété et en tête de première alternative peuvent être définis explicitement sans pour autant interférer avec les instructions automatiques de retour – voir Section 4.1.8 [Indications de reprise manuelles], page 210.

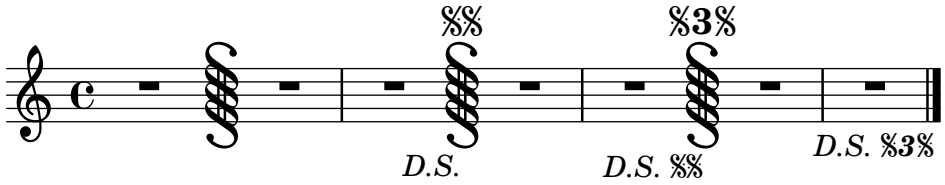
```
\fixed c' {
  \repeat segno 2 {
    \volta #'() { \segnoMark 2 }
    g1
    \alternative {
      \volta 1 {
        \volta #'() { \codaMark 2 }
        \*8 a4
      }
      \volta 2 \volta #'() {
        \section
        \sectionLabel "Coda"
      }
    }
  }
  b1
  \fine
}
```



Sans le `\segnoMark 2` explicité, ceci aurait été présenté comme une reprise *da-capo*.

Au lieu d'afficher un *segno* en tant que repère surplombant la portée, il est possible de l'imprimer sous forme de barre de mesure en affectant la propriété `segnoStyle` à `bar-line`. Afin de lever toute ambiguïté, seule la première barre de *segno* sera dépourvue de marque.

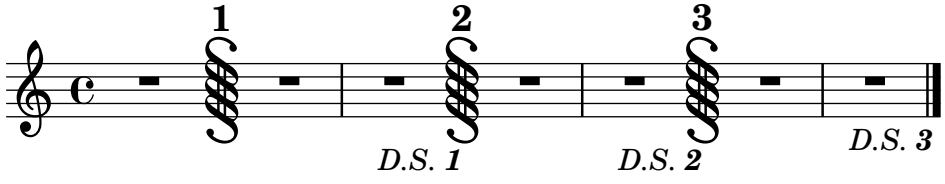
```
\fixed c' {
  \set Score.segnoStyle = #'bar-line
  R1
  \repeat unfold 3 {
    \repeat segno 2 {
      R1*2
    }
  }
  \fine
}
```



Lorsqu'une barre de segno coïncide avec une autre barre spéciale, sera automatiquement choisie une combinaison de barres parmi un jeu prédéfini. Pour chaque combinaison prise en compte, la barre de mesure peut être personnalisée à l'aide de propriétés de contexte – voir Section 2.5.2 [Barres de mesure automatiques], page 133.

Les marques de segno et de coda sont formatées par des procédures spécifiées dans les propriétés `segnoMarkFormatter` et `codaMarkFormatter`. Elles sont comparables aux procédures chargées de formater les repères – voir Section 2.5.5 [Indications de repère], page 144.

```
\fixed c' {
  \set Score.segnoMarkFormatter = #format-mark-numbers
  \set Score.segnoStyle = #'bar-line
  R1
  \repeat unfold 3 {
    \repeat segno 2 {
      R1*2
    }
  }
  \fine
}
```



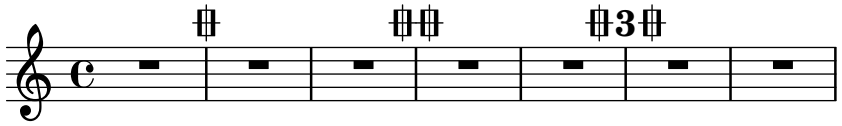
Il existe un formateur prédéfini alternatif de segno qui imprime un repère même sur la première barre de mesure :

```
\fixed c' {
  \set Score.segnoMarkFormatter = #format-segno-mark
  \set Score.segnoStyle = #'bar-line
  R1
  \repeat unfold 3 {
    \segnoMark \default
    R1*2
  }
}
```



Il existe un formateur prédéfini alternatif de coda qui imprime des signes \varcoda.

```
\fixed c' {
  \set Score.codaMarkFormatter = #format-varcoda-mark
  R1
  \repeat unfold 3 {
    \codaMark \default
    R1*2
  }
}
```
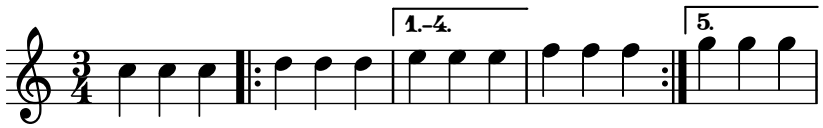


Morceaux choisis

Diminution de la taille du crochet d'alternative

Les crochets indiquant les fins alternatives s'étalent tout au long de celles-ci. On peut les raccourcir en jouant sur la propriété VoltaBracket.musical-length. Dans l'exemple suivant, le crochet ne s'étend que sur une mesure à 3/4.

```
\fixed c'' {
  \time 3/4
  c4 c c
  \repeat volta 5 {
    d4 d d
    \alternative {
      \volta 1,2,3,4 {
        \once \override Score.VoltaBracket.musical-length =
          \musicLength 2.
        e4 e e
        f4 f f
      }
      \volta 5 {
        g4 g g } } }
}
```

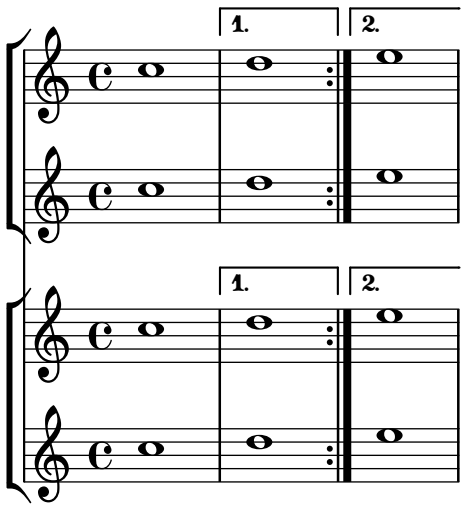


Crochets de volta sur plusieurs portées

L'ajout du `Volta_engraver` à une portée particulière permet que les crochets de *volta* s'affichent sur d'autres portées que la première d'une partition.

```
voltaMusic = \relative c'' {
  \repeat volta 2 {
    c1
    \alternative {
      \volta 1 { d1 }
      \volta 2 { e1 }
    }
  }
}

<<
  \new StaffGroup <<
    \new Staff \voltaMusic
    \new Staff \voltaMusic
  >>
  \new StaffGroup <<
    \new Staff \with { \consists "Volta_engraver" }
      \voltaMusic
    \new Staff \voltaMusic
  >>
>>
```



Numérotation des mesures et alternatives

Un réglage de la propriété de contexte `alternativeNumberingStyle` permet d'accéder à deux méthodes alternatives de gestion de la numérotation des mesures en cas de reprises.
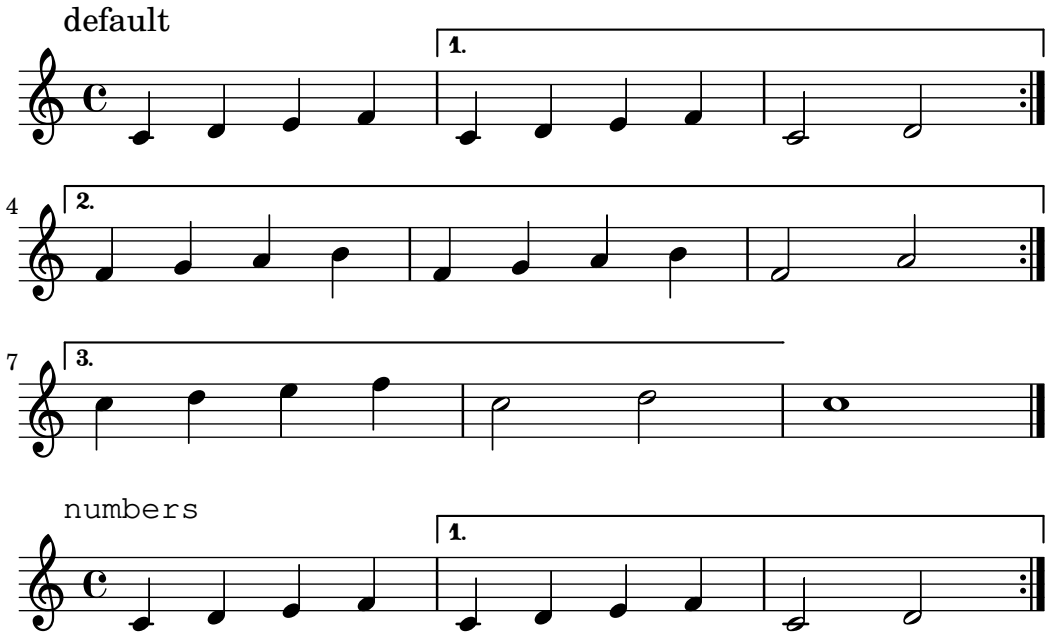
```
music = \relative c' {
  \repeat volta 3 {
    c4 d e f |
    \alternative {
      \volta 1 { c4 d e f | c2 d \break }
      \volta 2 { f4 g a b | f4 g a b | f2 a | \break }
      \volta 3 { c4 d e f | c2 d } } }
  c1 \bar "|."
}

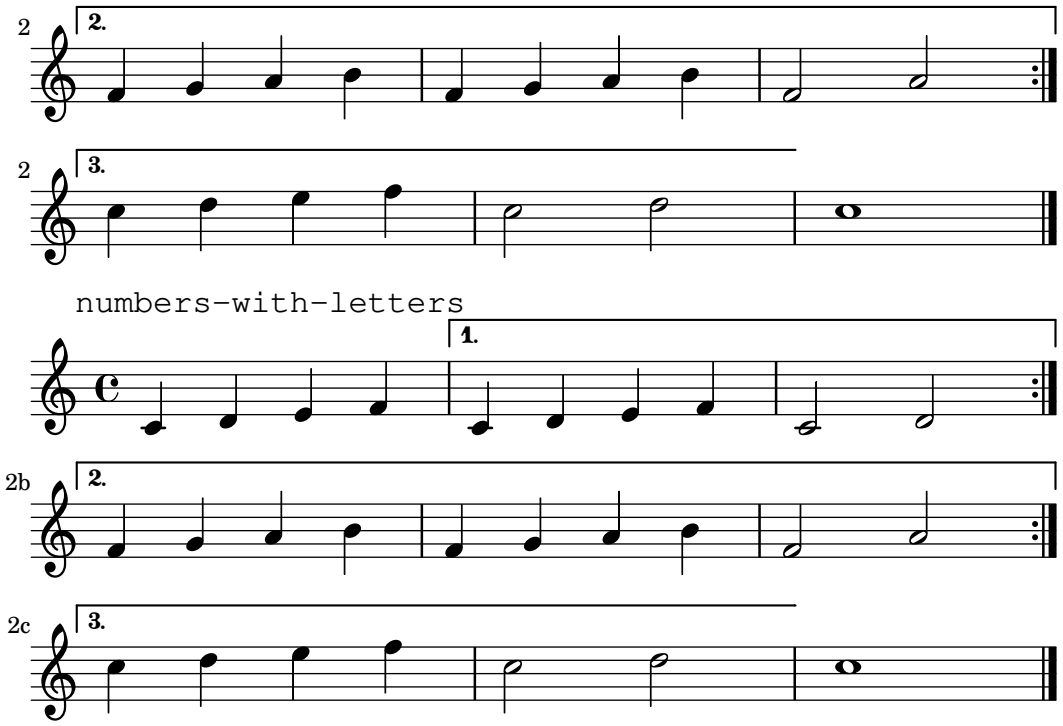
{
  \textMark \markup \large "default"
  \music
}

{
  \textMark \markup \large \typewriter "numbers"
  \set Score.alternativeNumberingStyle = #'numbers
  \music
}

{
  \textMark \markup \large \typewriter "numbers-with-letters"
  \set Score.alternativeNumberingStyle = #'numbers-with-letters
  \music
}

\layout {
  \context {
    \Score
    \override TextMark.Y-offset = #5
  }
}
```





Voir aussi

Glossaire musicologique : Section “répétition” dans *Glossaire*, Section “volta” dans *Glossaire*.

Manuel de notation : Section 2.5.1 [Barres de mesure], page 123, Section 2.5.2 [Barres de mesure automatiques], page 133, Section 2.3.1 [Mesures], page 80, Section 33.4 [Modification des greffons de contexte], page 769, Section 36.11.1 [Modification des liaisons], page 830.

Fichiers d’initialisation : `ly/engraver-init.ly`.

Morceaux choisis : Section “Répétitions et reprises” dans *Morceaux choisis*.

Référence des propriétés internes : Section “VoltaBracket” dans *Référence des propriétés internes*, Section “VoltaRepeatedMusic” dans *Référence des propriétés internes*, Section “UnfoldedRepeatedMusic” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L’extension d’une liaison, dans la forme `\volta`, n’est possible que pour la première alternative. Selon le même principe, une liaison ne saurait partir de la fin d’une alternative pour se terminer au début de la reprise.

L’aspect visuel d’une liaison se continuant dans les autres alternatives peut être simulé à l’aide de la commande `\repeatTie` lorsqu’elle s’arrête sur sa première note – méthode qui cependant ne fonctionne pas pour un `TabStaff` ; voir [Tenue et reprise], page 65. D’autres moyens existent pour indiquer la prolongation d’une liaison sur des alternatives, y compris dans un `TabStaff`, en suivant les préceptes donnés à la rubrique Section 36.11.1 [Modification des liaisons], page 830.

L’aspect visuel d’un glissando se continuant dans les autres alternatives peut être simulé à l’aide d’un glissando partant d’une note d’ornement supplémentaire et masquée. Un exemple se trouve à la rubrique Voir [Rappel du glissando à l’occasion d’une alternative], page 184.

Le développement, à l’aide de la commande `\unfoldRepeats`, d’une répétition qui commence sur une mesure incomplète et contient un bloc `alternative` avec modification de la propriété `measureLength` entraînera des messages d’erreur concernant le placement des barres de mesure.

Des reprises imbriquées utilisant l’ancienne syntaxe sont ambiguës :

```
\repeat ...
```

```
\repeat ...
\alternative
```

On ne sait à quelle section `\repeat` attribuer la section `\alternative`. Pour lever cette ambiguïté, il convient de toujours insérer la commande `\alternative` à l'intérieur de la section `\repeat`. Il est préférable, dans une telle situation, d'utiliser des accolades au sein du bloc de reprise pour plus de clarté.

4.1.8 Indications de reprise manuelles
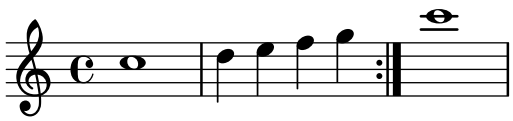
> **Note :** Les méthodes présentées dans les lignes qui suivent ne devraient servir à indiquer que des constructions de répétition inhabituelles. En règle générale, il vaut mieux recourir à la fonction `\repeat` pour créer une reprise ou bien insérer la barre de mesure adéquate. Pour plus d'information, voir Section 2.5.1 [Barres de mesure], page 123.

La propriété `repeatCommands` sert à contrôler la mise en forme des reprises du style « volta ». On la définit par une liste de commandes en Scheme. En règle générale, chaque élément est lui-même une liste de la forme `'(`*commande args...*`)`, tout en sachant qu'une commande sans argument peut s'abrégér en un symbole – par exemple, `'((start-repeat))` peut se libeller simplement en `'(start-repeat)`.

`end-repeat`

Pour terminer un fragment répété.

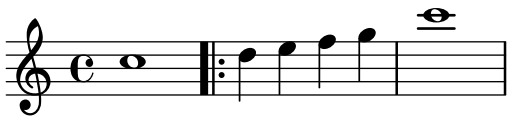
```
\relative {
  c''1
  d4 e f g
  \set Score.repeatCommands = #'(end-repeat)
  c1
}
```



`start-repeat`

Pour entamer un fragement répété.

```
\relative {
  c''1
  \set Score.repeatCommands = #'(start-repeat)
  d4 e f g
  c1
}
```
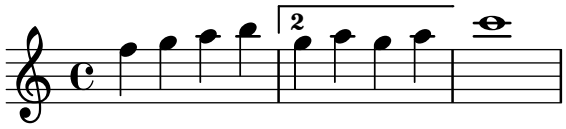


Traditionnellement, on n'imprime pas de signe de reprise en début de morceau.

`volta` *texte*

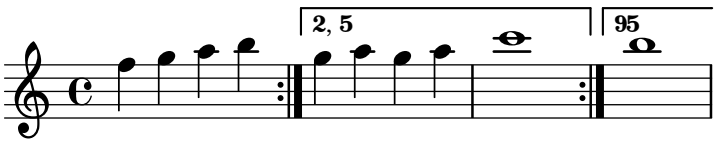
Lorsque *texte* est un *markup*, ouvre un crochet d'alternative et lui affecte le label mentionné. Le crochet prendra fin lorsque *texte* est égal à `#f`. Pour que le crochet s'imprime effectivement, il faut spécifier explicitement l'endroit où il doit se terminer.

```
\relative {
  f''4 g a b
  \set Score.repeatCommands =
    #`((volta ,#{ \markup \volta-number "2" #}))
  g4 a g a
  \set Score.repeatCommands = #'((volta #f))
  c1
}
```
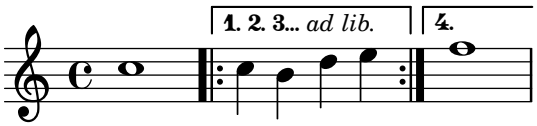


Plusieurs commandes de reprise peuvent intervenir au même moment :

```
\relative {
  f''4 g a b
  \set Score.repeatCommands =
    #`((volta ,#{ \markup { \concat { \volta-number 2 , }
                            \volta-number 5 } #})
       end-repeat)
  g4 a g a
  c1
  \set Score.repeatCommands =
    #`((volta #f)
       (volta ,#{ \markup \volta-number 95 #})
       end-repeat)
  b1
  \set Score.repeatCommands = #'((volta #f))
}
```



Le crochet indiquant une alternative peut contenir aussi du texte. Il peut s'agir d'un ou plusieurs nombres ou bien d'une indication textuelle (*markup*) – voir Section 8.2 [Mise en forme du texte], page 329. Le plus simple, dans le cas d'une indication textuelle, est de tout d'abord définir ce *markup*, puis de l'inclure dans une liste Scheme.

```
voltaAdLib = \markup { \volta-number { 1. 2. 3... }
                       \italic { ad lib. } }
\relative {
  c''1
  \set Score.repeatCommands = #`((volta ,voltaAdLib) start-repeat)
  c4 b d e
  \set Score.repeatCommands =
    #`((volta #f)
       (volta ,#{ \markup \volta-number "4." #})
       end-repeat)
  f1
  \set Score.repeatCommands = #'((volta #f))
}
```

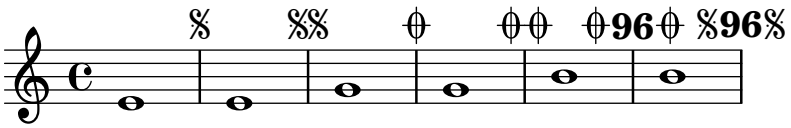


Les repères de *Segno* et *Coda* se créent avec les commandes `\segnoMark`, `\inStaffSegno` et `\codaMark`. La syntaxe consacrée est :

```
\codaMark n
\segnoMark n
```

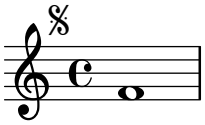
où *n* sont des numéros de séquence, ou bien `\default` pour utiliser une numérotation automatique. Les indications de repère, de segno et de coda sont incrémentées indépendamment l'une de l'autre.

```
\fixed c' {
  e1
  \segnoMark \default
  e1
  \segnoMark \default
  g1
  \codaMark \default
  g1
  \codaMark \default
  b1
  \codaMark 96
  b1
  \segnoMark 96
}
```



Au commencement d'une pièce, ni `\segnoMark \default` ni `\codaMark \default` ne créent de marque. Spécifier '1' permet de forcer leur apparition.

```
\fixed c' {
  \segnoMark 1
  f1
}
```



La commande `\inStaffSegno` est équivalente à `\segnoMark \default` à ceci près qu'elle affectera temporairement la propriété `segnoStyle` à `bar-line` afin de forcer son impression en tant que barre de mesure.
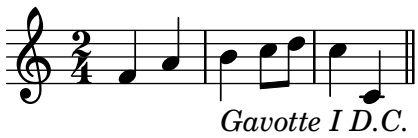
```
\fixed c' {
  e1
  \inStaffSegno
  g1
  \segnoMark \default
  b1
}
```



Pour de plus amples informations sur la manière d'affecter l'affichage des repères de segno ou de coda, voir Section 4.1.7 [Apparence d'une reprise *Segno*], page 204.

La création d'instructions de saut arbitraires s'obtient à l'aide de la commande `\jump`.

```
\fixed c' {
  \time 2/4
  f4 a
  b4 c'8 d'
  c'4 c
  \jump "Gavotte I D.C."
  \section
}
```



Voir aussi

Manuel de notation : Section 4.1.7 [Apparence d'une reprise *Segno*], page 204, Section 2.5.1 [Barres de mesure], page 123, Section 2.5.5 [Indications de repère], page 144, Section 8.2 [Mise en forme du texte], page 329.

Morceaux choisis_: Section "Répétitions et reprises" dans *Morceaux choisis*.

Référence des propriétés internes : Section "CodaMark" dans *Référence des propriétés internes*, Section "Jump_engraver" dans *Référence des propriétés internes*, Section "JumpScript" dans *Référence des propriétés internes*, Section "Mark_engraver" dans *Référence des propriétés internes*, Section "SegnoMark" dans *Référence des propriétés internes*, Section "SegnoRepeatedMusic" dans *Référence des propriétés internes*, Section "VoltaBracket" dans *Référence des propriétés internes*, Section "VoltaRepeatedMusic" dans *Référence des propriétés internes*.

4.2 Autres types de répétition

Nous abordons ici les reprises de courte durée. Il en existe deux formes, à savoir la répétition d'une même note sur quelques mesures – représentée par une barre oblique ou le signe pourcent – et les trémolos.

4.2.1 Répétitions de mesure
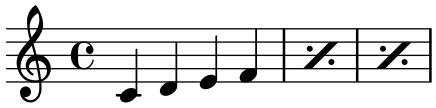
Le style de « reprise en pourcent » sert à répéter une séquence de notes. Elle sera imprimée une fois, puis remplacée par un symbole spécial.

En voici la syntaxe :

```
\repeat percent nombre expression_musicale
```
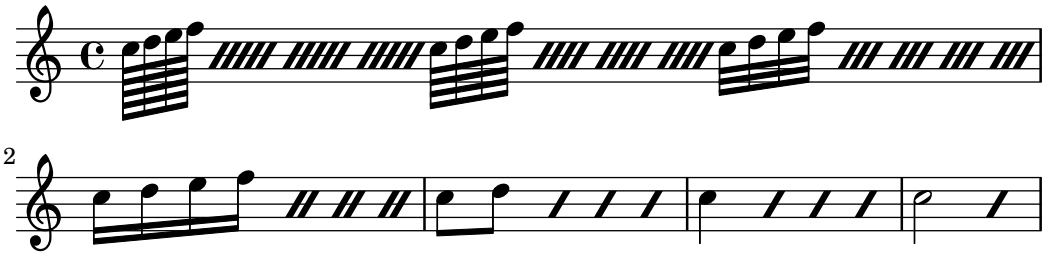
Le raccourci `\%` se révèle pratique dans une œuvre qui utilise beaucoup de répétitions en pourcent :

```
\%3 { c'4 d' e' f' }
```
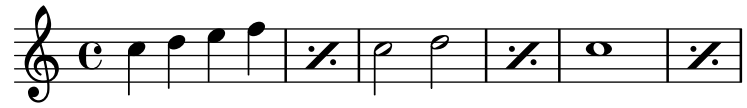


Les séquences inférieures à une mesure sont remplacées par une barre oblique.

```
\relative c'' {
  \%4 { c128 d e f }
  \%4 { c64 d e f }
  \%5 { c32 d e f }
  \%4 { c16 d e f }
  \%4 { c8 d }
  \%4 { c4 }
  \%2 { c2 }
}
```
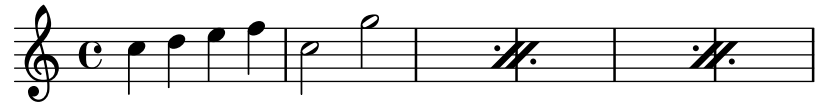


Les séquences d'une ou deux mesures sont remplacées par un symbole qui ressemble au symbole de pourcentage.

```
\relative c'' {
  \%2 { c4 d e f }
  \%2 { c2 d }
  \%2 { c1 }
}
```

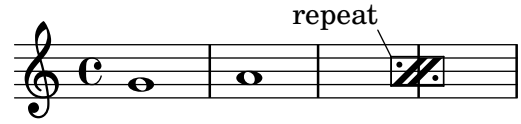


```
\relative {
  \%3 { c''4 d e f | c2 g' }
}
```



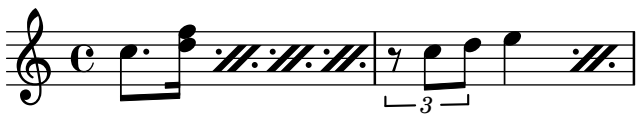
Notez bien le décalage dans le temps pour accéder à l'objet graphique `DoublePercentRepeat` : il s'agit de la deuxième mesure de la répétition. On y accède aisément à l'aide de la commande `\after`.

```
\new Voice \with { \consists Balloon_engraver }
  { \after 1*3
      \balloonGrobText DoublePercentRepeat #'(-1 . 2) "repeat"
    \%2 { g'1 | a'1 } }
```



Les séquences inférieures à la mesure et qui contiennent des durées différentes sont remplacées par un double symbole de pourcentage.

```
\relative {
  \%4 { c''8. <d f>16 }
  \%2 { \tuplet 3/2 { r8 c d } e4 }
}
```

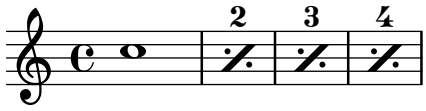


Morceaux choisis

Compteur de répétition en pourcent

Les répétitions de plus de deux mesures sont surmontées d'un compteur, si l'on active la propriété de contexte `countPercentRepeats`.

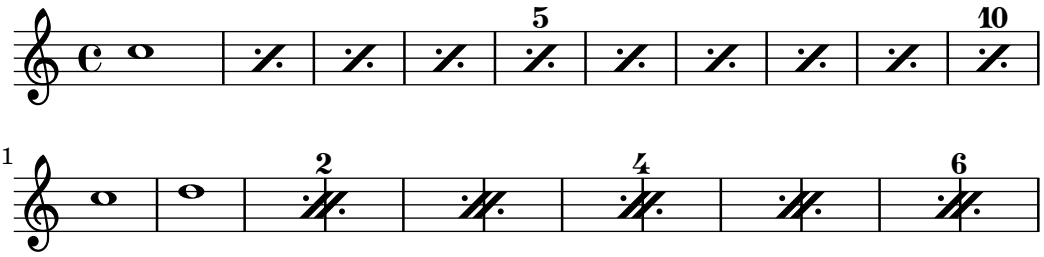
```
\relative c'' {
  \set countPercentRepeats = ##t
  \%4 c1
}
```



Affichage du numéro de répétition en pourcent

Le numéro de mesure répétée sera imprimé à intervalle régulier si vous déterminez la propriété de contexte `repeatCountVisibility`.

```
\relative c'' {
  \set countPercentRepeats = ##t
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 5)
  \%10 c1 \break
  \set repeatCountVisibility = #(every-nth-repeat-count-visible 2)
  \%6 { c1 d1 }
}
```



Répétition en pourcent isolée

Des symboles de pourcentage isolés peuvent aussi être obtenus au moyen d'un silence multimesure dont on modifie l'aspect :

```
makePercent =
#(define-music-function (note) (ly:music?)
   "Make a percent repeat the same length as NOTE."
   (make-music 'PercentEvent
               'length (ly:music-length note)))

\relative c'' {
  \makePercent s1
}
```



Voir aussi

Glossaire musicologique : Section “percent repeat” dans *Glossaire*, Section “simile” dans *Glossaire*.
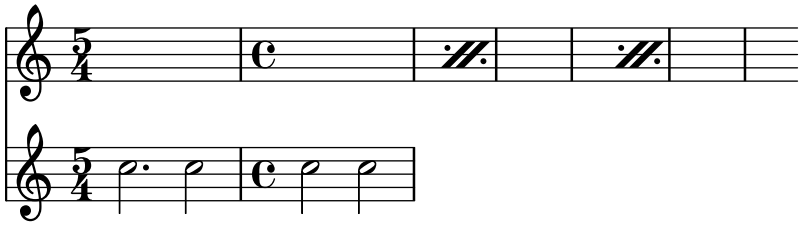
Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

Référence des propriétés internes : Section “RepeatSlash” dans *Référence des propriétés internes*, Section “RepeatSlashEvent” dans *Référence des propriétés internes*, Section “DoubleRepeatSlash” dans *Référence des propriétés internes*, Section “PercentRepeat” dans *Référence des propriétés internes*, Section “PercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatedMusic” dans *Référence des propriétés internes*, Section “Percent_repeat_engraver” dans *Référence des propriétés internes*, Section “DoublePercentEvent” dans *Référence des propriétés internes*, Section “DoublePercentRepeat” dans *Référence des propriétés internes*, Section “DoublePercentRepeatCounter” dans *Référence des propriétés internes*, Section “Double_percent_repeat_engraver” dans *Référence des propriétés internes*, Section “Slash_repeat_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements
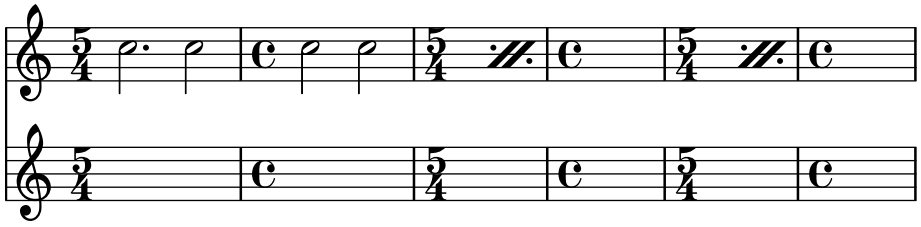
Les répétitions en pourcent ne peuvent contenir rien d’autre que le signe pourcent lui-même ; en particulier, les changements de métrique ne seront pas répétés.

```
\%3 { \time 5/4 c2. 2 \time 4/4 2 2 }
```



Tout changement de métrique ou insertion d’une commande `\partial` devra impérativement se traiter sous forme de construction parallèle, **en dehors** de la répétition en pourcentage :

```
<<
  \%3 { c2. 2 2 2 }
  \*3 { \time 5/4 s4*5 \time 4/4 s1 }
>>
```
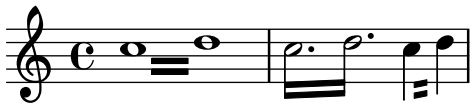


4.2.2 Répétitions en trémolo

Il y a deux formes de trémolo : la répétition alternative de deux notes ou accords, et la répétition rapide d'une seule note ou d'un accord. Lorsqu'il est constitué d'une alternance répétitive, le trémolo s'indique en ajoutant des barres de ligature entre les deux notes ou accords concernés. Lorsqu'il s'agit d'une répétition rapide, des barres de ligature penchées sont ajoutées à la note en question.

On peut placer une notation de trémolo entre deux notes, avec la commande `\repeat` suivie du style trémolo :

```
\relative c'' {
  \repeat tremolo 8 { c16 d }
  \repeat tremolo 6 { c16 d }
  \repeat tremolo 2 { c16 d }
}
```
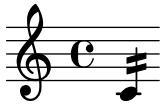


La syntaxe de `\repeat tremolo` requiert expressément deux notes encadrées par des accolades, et le nombre de répétitions exprimé en durée d'une note (pointée ou non). Ainsi, dans l'exemple ci-dessus, `\repeat tremolo 7` est valide car correspond à une note doublement pointée, à l'inverse de `\repeat tremolo 9`.

La durée d'un trémolo est égale à la durée de l'expression entre accolades multipliée par le nombre de fois à répéter : `\repeat tremolo 8 { c16 d16 }` correspond donc à la valeur d'une ronde, et sera représenté par deux rondes séparées par des barres de trémolo.
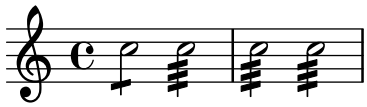
On peut indiquer de la même manière un trémolo sur une seule note, qu'il faudra alors laisser sans accolades :

```
\repeat tremolo 4 c'16
```



Le même résultat s'obtient en faisant suivre la note considérée de deux points et d'un nombre (`note:`*`nombre`*). Le nombre en question correspond à la valeur de la subdivision ; il doit être au moins de 8, auquel cas la hampe sera barrée par un seul trait de ligature. Si ce nombre est omis, la dernière valeur sera utilisée.

```
\relative {
  c''2:8 c:32
  c: c:
}
```

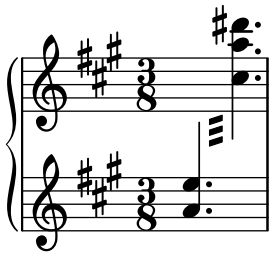


Morceaux choisis

Trémolo et changement de portée

Dans la mesure où `\repeat tremolo` requiert deux arguments musicaux pour un trémolo d'accords, la note ou l'accord de la portée opposée doit être encadré par des accolades et se voir adjoindre la commande `\change Staff`.

```
\new PianoStaff <<
  \new Staff = "up" \relative c'' {
    \key a \major
    \time 3/8
    s4.
  }
  \new Staff = "down" \relative c'' {
    \key a \major
    \time 3/8
    \voiceOne
    \repeat tremolo 6 {
      <a e'>32
      {
        \change Staff = "up"
        \voiceTwo
        <cis a' dis>32
      }
    }
  }
>>
```



Contrôle de l'apparence des traits de trémolo

Différentes propriétés de l'objet graphique `StemTremolo` permettent de gérer l'apparence des traits de trémolo.

- La propriété `slope` définit la pente des traits.
- La propriété `shape` détermine si les traits doivent ressembler à des rectangles (valeur `rectangle`) ou bien à des petites ligatures (valeur `beam-like`).

- La propriété `style` gère à la fois la pente et la forme selon que la note est affublée d'un crochet, est ligaturée ou n'a qu'une simple hampe. Ceci diffère des deux autres propriétés qui, elles, modifient l'apparence de façon inconditionnelle. Deux styles sont définis.
  - `default` : les traits en présence de hampe descendante et de crochet sont plus longs et plus pentus que pour des hampes avec crochet ascendantes ; les traits sur des notes ligaturées prennent l'apparence de rectangles et sont parallèles à la ligature.
  - `constant` : tous les traits ont l'apparence de petites ligatures et ont la même pente, hormis lorsque sur des hampes descendantes crochetées.

```
music = {
  a''4:32 a':
  e''8: \noBeam e':
  a'':[ a':]
  f':[ g':]
  d':[ d':]
}
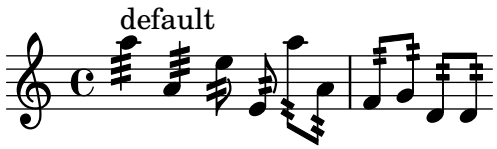
\new Staff {
  <>^\markup "default"
  \music
}

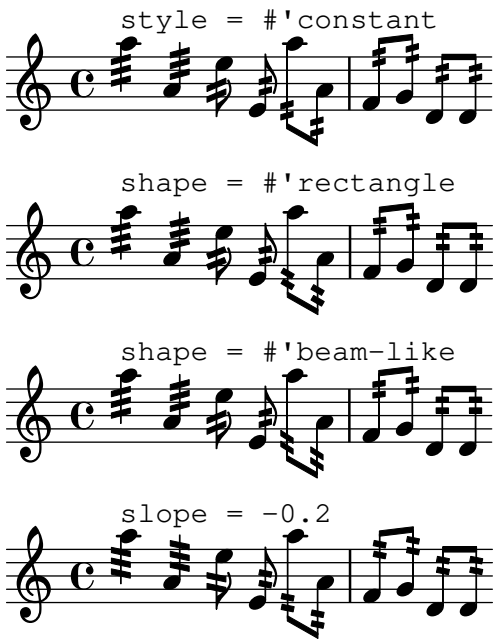
\new Staff {
  <>^\markup \typewriter "style = #'constant"
  \override StemTremolo.style = #'constant
  \music
}

\new Staff {
  <>^\markup \typewriter "shape = #'rectangle"
  \override StemTremolo.shape = #'rectangle
  \music
}

\new Staff {
  <>^\markup \typewriter "shape = #'beam-like"
  \override StemTremolo.shape = #'beam-like
  \music
}

\new Staff {
  <>^\markup \typewriter "slope = -0.2"
  \override StemTremolo.slope = -0.2
  \music
}
```
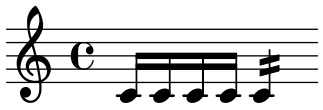




Voir aussi

Morceaux choisis : Section “Répétitions” dans *Morceaux choisis*.

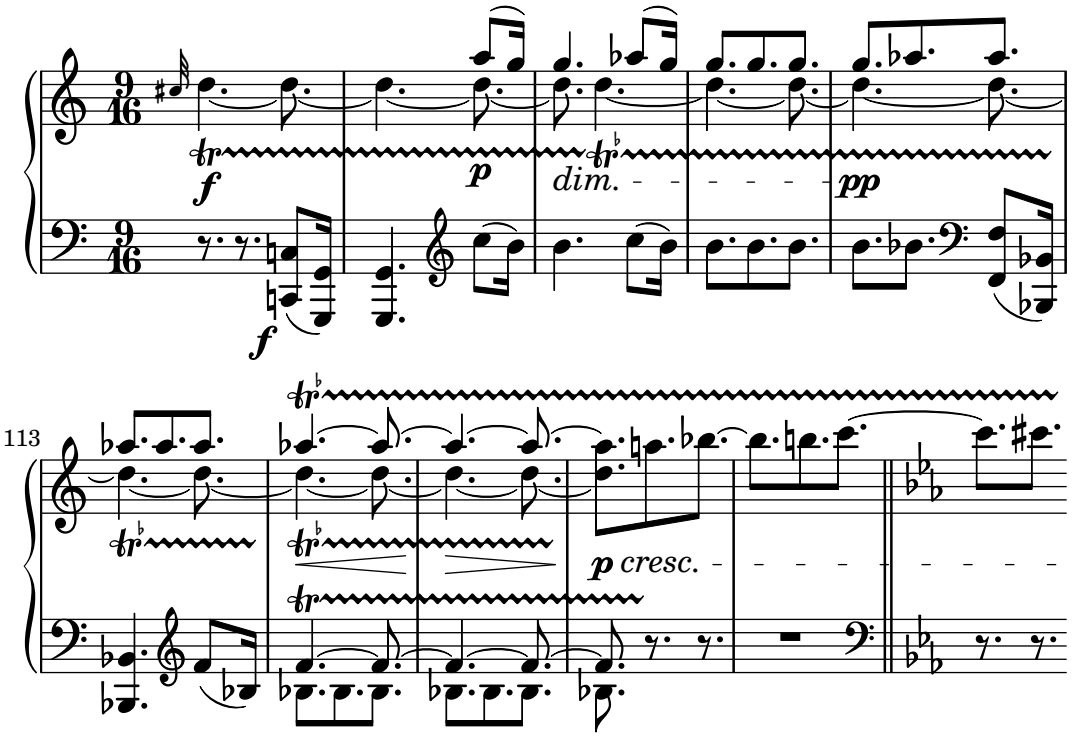
Problèmes connus et avertissements

Bien que la syntaxe `\repeat tremolo` soit expansée par un `\unfoldRepeats`, son raccourci ne l’est pas (ceci fait l’objet du ticket 6145 (`https://gitlab.com/lilypond/lilypond/-/issues/6145`)) :

```
{
  \unfoldRepeats \repeat tremolo 4 c'16
  \unfoldRepeats c'4:16
}
```



5 Notes simultanées



La notion musicale de polyphonie fait référence au fait d'avoir plus d'une voix simultanément dans une pièce. Dans LilyPond, la notion de polyphonie fait référence au fait d'avoir plus d'une voix sur la même portée.

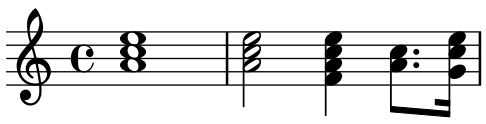
5.1 Monophonie

Nous allons voir ici comment gérer plusieurs notes simultanées dans une même voix.

5.1.1 Notes en accords

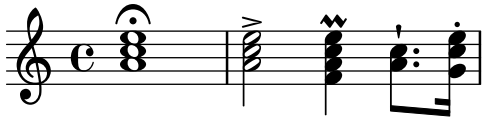
Un accord est formé en mettant une série de hauteurs entre < et >. Un accord peut être suivi d'une durée comme une simple note.

```
\relative {
  <a' c e>1 <a c e>2 <f a c e>4 <a c>8. <g c e>16
}
```



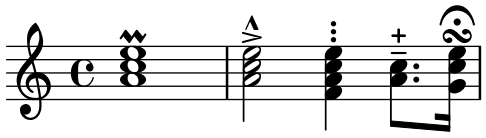
Un accord peut être suivi d'une indication d'articulation comme une simple note.

```
\relative {
  <a' c e>1\fermata <a c e>2-> <f a c e>4\prall <a c>8.^! <g c e>16-.
}
```



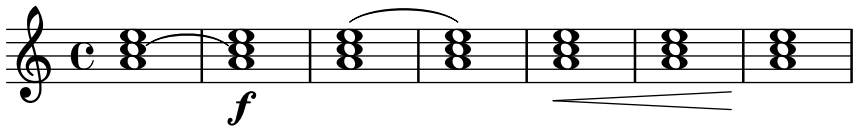
Certaines notes, au sein même d'un accord, peuvent être affectées d'une articulation ou d'un ornement :

```
\relative {
  <a' c\prall e>1 <a-> c-^ e>2 <f-. a c-. e-.>4
  <a-+ c-->8. <g\fermata c e\turn>16
}
```
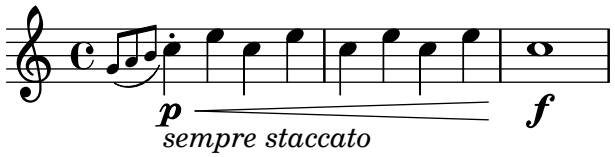


Certains éléments de notation, tels que nuances et soufflets ne peuvent se rattacher qu'à un accord et non aux notes qui le composent, sous peine de ne les voir s'imprimer. D'autres éléments, tels que doigtés ou liaisons, seront placés différemment selon qu'ils sont rattachés à des notes composant un accord, à un accord dans sa globalité ou à des notes individuelles.

```
\relative {
  <a'\f c( e>1 <a c) e>\f <a\< c e>( <a\! c e>)
  <a c e>\< <a c e> <a c e>\!
}
```



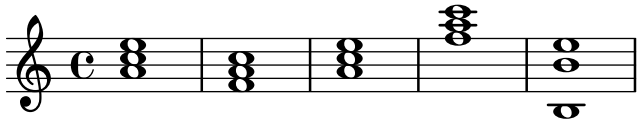
Un accord peut se voir comme un conteneur de notes, articulations et autres éléments rattachés. Par voie de conséquence, un accord dépourvu de note n'a pas de durée ; toute articulation qui lui serait attachée interviendra au même moment que la note ou l'accord qui le suit et lui sera donc combiné – pour des combinaisons plus complexes, voir Section 5.1.3 [Expressions simultanées], page 225.

```
\relative {
  \grace { g'8( a b }
  <>) \p \< -. -\markup \italic "sempre staccato"
  \*4 { c4 e } c1\f
}
```



Les accords peuvent être saisis en mode relatif. Dans un accord, l'octave de chaque hauteur saisie est fonction de la précédente, à l'exception de la première qui, elle, sera positionnée en fonction de la première hauteur mentionnée dans l'accord précédent ou de la note individuelle précédente. Les autres notes au sein de l'accord se positionnent relativement à la précédente *dans cet accord*.

```
\relative {
  <a' c e>1 <f a c> <a c e> <f' a c> <b, e b,>
}
```



Pour plus d'information à propos des accords, voir Chapitre 15 [Notation des accords], page 522.

Voir aussi

Glossaire musicologique : Section “accord” dans *Glossaire*.

Manuel d’initiation : Section “Combinaison de notes en accords” dans *Manuel d’initiation*.

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Chapitre 15 [Notation des accords], page 522, Section 1.1.2 [Octaves relatives], page 4, Section 5.2 [Plusieurs voix], page 227.

Morceaux choisis : Section “Notes simultanées” dans *Morceaux choisis*.

Problèmes connus et avertissements

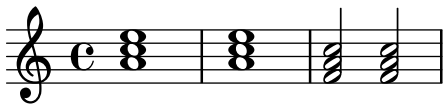
Un accord comportant plus de deux notes dans le même « espace de portée » – tel que ‘`<e f! fis!>`’ – conduit immanquablement à des chevauchements. En fonction de la situation, un meilleur rendu peut nécessiter de recourir à

- l’utilisation temporaire de Section 5.2 [Plusieurs voix], page 227, ‘`<< f! \\ <e fis!> >>`’,
- une transcription enharmonique d’une ou plusieurs hauteurs, ‘`<e f ges>`’, ou
- des Section 5.1.4 [Clusters], page 226.

5.1.2 Répétition d’accords

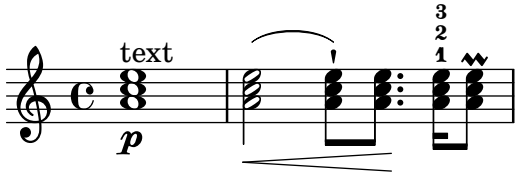
Dans le but de vous épargner de la saisie, LilyPond dispose d’un raccourci – symbolisé par la lettre `q` – qui a pour effet de répéter le dernier accord saisi :

```
\relative {
  <a' c e>1 q <f a c>2 q
}
```



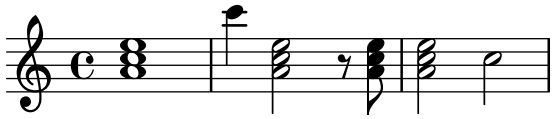
À l’instar de n’importe quel accord, le symbole de répétition peut être affublé d’une durée, de signes d’articulation, *markups*, liaisons, ligatures... En fait, c’est la structure du dernier accord qui est dupliquée.

```
\relative {
  <a' c e>1\p^"text" q2\<( q8)[-! q8.]\! q16-1-2-3 q8\prall
}
```
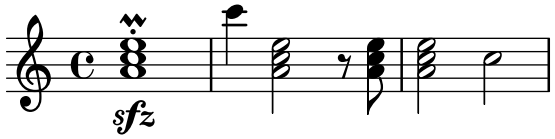


Dans la mesure où le symbole de répétition d’accord enregistre la structure du dernier accord construit, il est tout à fait possible de l’utiliser même après une succession de notes individuelles et de silences :

```
\relative {
  <a' c e>1 c'4 q2 r8 q8 |
  q2 c, |
}
```



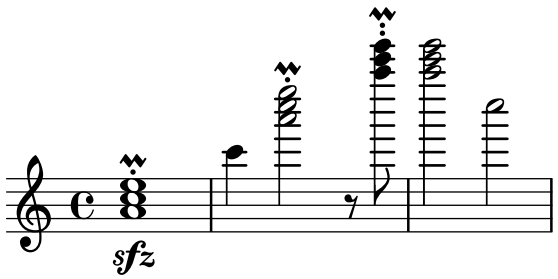
Le symbole de répétition d'accord ne prend en charge que les hauteurs, en aucun cas les nuances, articulations ou ornementations, qu'elles aient été attachées aux notes le composant ou à l'ensemble.

```
\relative {
  <a'-. c\prall e>1\sfz c'4 q2 r8 q8 |
  q2 c, |
}
```



Le seul moyen de les reproduire consiste à utiliser explicitement la fonction `\chordRepeats`, en lui adjoignant un argument supplémentaire qui recense les *types d'événement* à répéter et qui seraient absents de l'accord construit par un `q`.
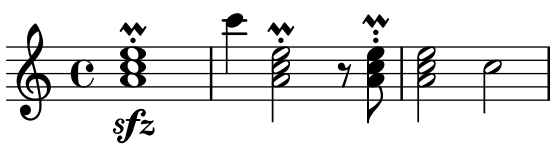
```
\relative {
  \chordRepeats #'(articulation-event)
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c, |
}
```



Comme vous pouvez le constater, l'utilisation de `\chordRepeats` au sein d'un bloc `\relative` ne produit pas le résultat escompté : les événements de l'accord expansés sont identiques à la saisie traditionnelle d'un accord, ce qui a pour conséquence que l'octave affectée par `\relative` repose sur le contexte en cours.

Dans la mesure où l'imbrication de clauses `\relative` n'est pas source d'interférence, l'ajout d'un `\relative` à ce qui sera concerné par l'instruction `\chordRepeats` permet d'établir une relation d'octave entre les accords dès avant leur expansion. Dans le cas présent, l'intégralité du bloc `\relative` intérieur n'affecte en rien ce qui l'entoure, ce qui explique la spécification d'octave attachée à la dernière note :

```
\new Voice
\relative {
  \chordRepeats #'(articulation-event)
  \relative
  { <a'-. c\prall e>1\sfz c'4 q2 r8 q8-. } |
  q2 c'' |
}
```



Les interactions avec `\relative` ne se produisent que lors d'un appel explicite de `\chordRepeats` : l'expansion implicite en début de saisie intervient à un moment où toutes les instances de `\relative` ont déjà été interprétées.

Voir aussi

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Chapitre 15 [Notation des accords], page 522.
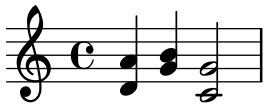
Fichiers d'initialisation : `ly/chord-repetition-init.ly`.

5.1.3 Expressions simultanées
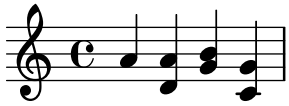
Lorsqu'une ou plusieurs expressions musicales sont encadrées par des doubles chevrons, elles sont considérées comme étant simultanées. Si la première expression débute par une note unique ou si l'intégralité de l'expression simultanée est explicitement rattachée à une voix en particulier, elle sera placée sur une seule portée. Dans le cas contraire, les éléments d'une expression simultanée seront placés sur des portées distinctes.

Voici deux exemples d'expression simultanée sur une même portée :

```
\new Voice {  % explicit single voice
  << \relative { a'4 b g2 }
     \relative { d'4 g c,2 } >>
}
```
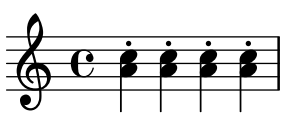


```
\relative {
  % single first note
  a' << \relative { a'4 b g }
        \relative { d'4 g c, } >>
}
```



Cette manière de procéder est utile lorsque les éléments de l'expression ont des rythmes identiques. Dès que vous tenterez d'attacher sur une même hampe des notes de durée différente, vous générerez des erreurs. Notes, articulations et modifications de propriétés au sein d'un même `Voice` sont enregistrées et gravées selon l'ordre musical :
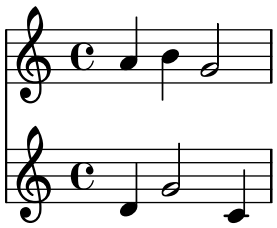
```
\relative {
  <a' c>4-.  <>-. << c a >>  << { c-. <c a> } { a s-. } >>
}
```



La présence de plusieurs hampes, ligatures, durées ou propriétés au même instant musical nécessite l'utilisation de plusieurs voix.

Dans l'exemple suivant, l'expression simultanée génère implicitement plusieurs portées :

```
% no single first note
<< \relative { a'4 b g2 }
   \relative { d'4 g2 c,4 } >>
```



En pareil cas, des rythmes différents ne sont source d'aucun problème puisqu'ils sont interprétés dans des voix différentes.

Problèmes connus et avertissements

Des notes, bien qu'appartenant à des voix différentes, mais dont les hampes ont la même orientation, peuvent se retrouver au même endroit sur la portée, ce quelque soit le décalage que vous auriez pu leur appliquer. Ceci ne manque pas de faire apparaître un message

```
warning: This voice needs a \voiceXx or \shiftXx setting

en français :
Avertissement : Cette voix requiert un voiceXx ou un réglage \shiftXx
```

lors de la compilation. Le déclenchement de cet avertissement peut être désactivé par une clause

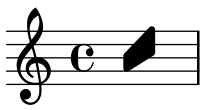
```
\override NoteColumn.ignore-collision = ##t
```

Ceci n'aura pas pour seule conséquence que ce message ne sera plus émis ; les procédures d'évitement de collision de quelqu'ordre que ce soit seront désactivées, ce qui peut conduire à quelques effets inattendus (voir aussi *Problèmes connus et avertissements* à la rubrique Section 5.2.3 [Résolution des collisions], page 233).

5.1.4 Clusters

Un cluster indique un agrégat de sons. On peut le représenter par une plage limitée par un *ambitus* (notes extrêmes). On obtient une telle notation en appliquant la fonction `\makeClusters` à une séquence d'accords, comme

```
\relative \makeClusters { <g' b>2 <c g'> }
```



Des notes ordinaires et des clusters peuvent cohabiter sur une même portée, y compris simultanément – en pareil cas, rien ne sera fait pour tenter d'empêcher les chevauchements entre notes et clusters.

Plusieurs styles de cluster sont disponibles : `ramp`, `leftsided-stairs`, `rightsided-stairs` et `centered-stairs`.

```
fragment = { <e' d''>4 <g' a'> <e' a'> r }

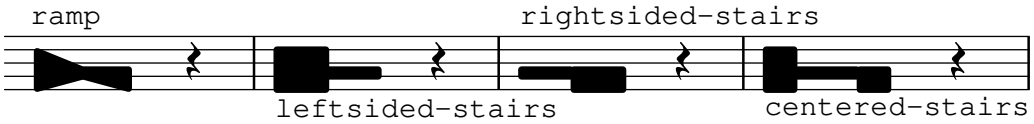
{
  \omit Staff.Clef
  \omit Staff.TimeSignature

  <>^\markup \typewriter "ramp"
  \override ClusterSpanner.style = #'ramp
  \makeClusters \fragment

  <>_\markup \typewriter "leftsided-stairs"
  \override ClusterSpanner.style = #'leftsided-stairs
  \makeClusters \fragment

  <>^\markup \typewriter "rightsided-stairs"
  \override ClusterSpanner.style = #'rightsided-stairs
  \makeClusters \fragment

  <>_\markup \typewriter "centered-stairs"
  \override ClusterSpanner.style = #'centered-stairs
  \makeClusters \fragment
}
```



Voir aussi

Glossaire musicologique : Section “cluster” dans *Glossaire*.

Morceaux choisis : Section “Notation simultanée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “ClusterSpanner” dans *Référence des propriétés internes*, Section “ClusterSpannerBeacon” dans *Référence des propriétés internes*, Section “Cluster_spanner_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L’apparence d’un cluster sera extrêmement resserrée s’il ne comporte au moins deux accords.

Dans la mesure où un cluster ne possède pas de hampe, il n’y a aucun moyen d’en connaître la durée ; cependant la longueur du signe imprimé dépend directement de la durée affectée aux accords qui le définissent.

Seul un silence peut séparer deux clusters.

Les clusters ne sont pas reproduits en MIDI.

5.2 Plusieurs voix
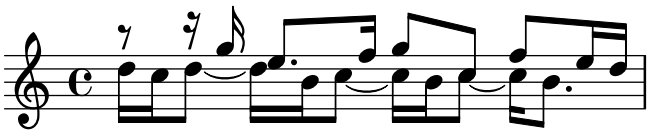
Nous allons nous intéresser, dans les paragraphes qui suivent, à la gestion de notes simultanées réparties sur plusieurs voix ou plusieurs portées.

5.2.1 Polyphonie sur une portée

Instanciation explicite des voix

La manière la plus facile d’entrer des fragments avec plus d’une voix sur une portée est la suivante :

```
\new Staff <<
  \new Voice = "first"
    \relative { \voiceOne r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \new Voice= "second"
    \relative { \voiceTwo d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Vous constaterez que les voix sont créées explicitement et qu'elles sont nommées. Les commandes `\voiceOne ... \voiceFour` déterminent les voix de telle sorte que les première et troisième auront des hampes vers le haut, et les deuxième et quatrième vers le bas. D'autre part, les notes des troisième et quatrième voix seront quelque peu décalées, tout comme leurs silences, afin d'éviter les collisions. La commande `\oneVoice` permet de retrouver les réglages par défaut.

N'oubliez pas que `Voice` est un contexte de bas niveau – voir Section 33.1.5 [Contextes de bas niveau – les voix], page 762. Dans le cadre d'un `TabStaff`, il faudrait donc utiliser un `TabVoice`.
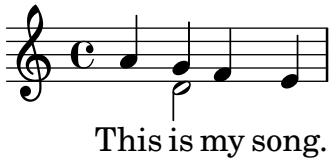
Polyphonie temporaire

Un fragment temporairement polyphonique se construit de la manière suivante :

```
<< { \voiceOne ... }
  \new Voice { \voiceTwo ... }
>> \oneVoice
```

En fait, la première expression d'une polyphonie temporaire reste dans le même contexte `Voice` que celui existant auparavant et qui perdurera après ce fragment. Les autres expressions entre doubles chevrons seront assignées à des voix temporaires distinctes. C'est la raison pour laquelle les paroles qui suivaient la voix avant la polyphonie continueront à le faire durant ce passage polyphonique et après lui :

```
\relative <<
  \new Voice = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Voice {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }
  \new Lyrics \lyricsto "melody" {
  This is my song.
  }
>>
```



Vous remarquerez que les commandes `\voiceOne` et `\voiceTwo` permettent d'obtenir des réglages différents pour chacune des voix.
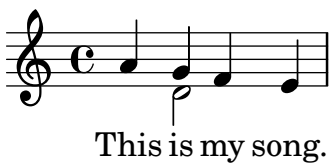
Dans un `TabStaff` doit être utilisé un `TabVoice`.

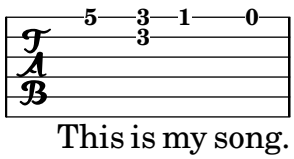
Lorsque la même musique doit apparaître à la fois dans un `Staff` **et** dans un `TabStaff`, il est possible de recourir au contexte générique `Bottom` – voir Section 33.1.5 [Contextes de bas niveau – les voix], page 762.

```
mus =
\relative
  \new Bottom = "melody" {
    a'4
    <<
      {
        \voiceOne
        g f
      }
      \new Bottom {
        \voiceTwo
        d2
      }
    >>
    \oneVoice
    e4
  }

<<
  \new Staff \mus
  \new Lyrics \lyricsto "melody" {
    This is my song.
  }
>>

<<
  \new TabStaff \mus
  \new Lyrics \lyricsto "melody" {
    This is my song.
  }
>>
```
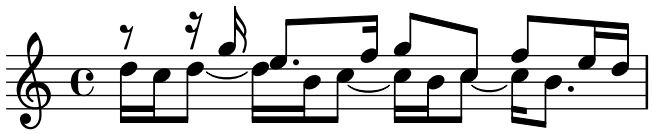




La construction avec un double antislash

Une construction de la forme `<< {...} \\ {...} >>`, dans laquelle plusieurs expressions sont séparées par des doubles obliques inversées, se comporte différemment de celle sans séparateur : **tous** les membres de cette construction seront assignés à de nouveaux contextes `Bottom` du même type, à savoir `Voice` ou `TabVoice`. Ces nouveaux contextes `Bottom`, créés implicitement, portent les noms `"1"`, `"2"`, etc. Dans chacun de ces contextes, le positionnement des liaisons, la direction des hampes, etc. sont réglés de manière appropriée. En voici un exemple :
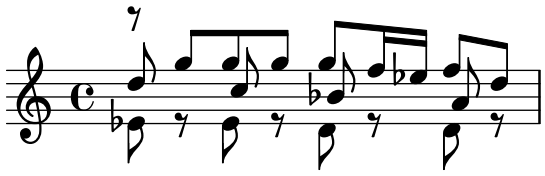
```
<<
  \relative { r8 r16 g'' e8. f16 g8[ c,] f e16 d }
  \\
  \relative { d''16 c d8~ 16 b c8~ 16 b c8~ 16 b8. }
>>
```



Cette syntaxe peut être utilisée dans la mesure où la création puis la disparition de voix temporaires sont sans conséquence. Les réglages de ces voix créées implicitement sont les mêmes que si elles avaient été créées à l'aide des commandes `\voiceOne` à `\voiceFour`, dans leur ordre d'apparition.

Dans l'exemple qui suit, la voix intermédiaire a des hampes vers le haut. Nous la plaçons donc en troisième position, de telle sorte qu'elle adopte les réglages de `\voiceThree` qui correspondent à ce que nous voulons. Grâce à des espaces invisibles, nous évitons de surcharger la portée avec des demi-soupirs.

```
<<
  \relative { r8 g'' g g g f16 ees f8 d }
  \\
  \relative { ees'8 r ees r d r d r }
  \\
  \relative { d''8 s c s bes s a s }
>>
```



En dehors des cas les plus simples, nous vous invitons à toujours créer les contextes de voix de manière explicite. Voir à ce sujet Section "Contextes et graveurs" dans *Manuel d'initiation* et Section "Instanciation explicite des voix" dans *Manuel d'initiation*.

Ordre des voix
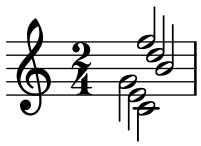
L'ordre dans lequel doivent apparaître les voix d'une construction simultanée suit le schéma suivant :

```
Voix 1 : la plus haute
```

```
Voix 2 : la plus basse
Voix 3 : deuxième plus haute
Voix 4 : deuxième plus basse
Voix 5 : troisième plus haute
Voix 6 : troisième plus basse
etc.
```
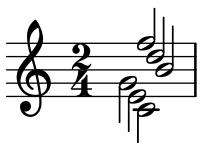
Cette présentation en entonnoir peut sembler quelque peu contre-intuitive ; elle simplifie cependant grandement le processus de mise en forme. Vous noterez que les hampes des voix au numéro impair vont vers le haut, celles des voix paires vers le bas :

```
\new Staff <<
  \time 2/4
  { f''2 }  % 1: highest
  \\
  { c'2  }  % 2: lowest
  \\
  { d''2 }  % 3: second-highest
  \\
  { e'2  }  % 4: second-lowest
  \\
  { b'2  }  % 5: third-highest
  \\
  { g'2  }  % 6: third-lowest
>>
```



La commande \voices permet de gérer l'ordre de saisie des voix :

```
\new Staff \voices 1,3,5,6,4,2 <<
  \time 2/4
  { f''2 }  % 1: highest
  \\
  { d''2 }  % 3: second-highest
  \\
  { b'2  }  % 5: third-highest
  \\
  { g'2  }  % 6: third-lowest
  \\
  { e'2  }  % 4: second-lowest
  \\
  { c'2  }  % 2: lowest
>>
```
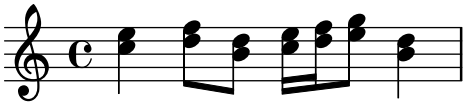


Note : Paroles et objets étendus (liaisons, soufflets, etc.) ne peuvent passer d'une voix à l'autre.

Identité rythmique

Lorsque l'on doit saisir des fragments de musique parallèle qui ont le même rythme, on peut les combiner dans un contexte de voix unique et par voie de conséquence former des accords. Il suffit pour cela de les regrouper dans une construction de musique simultanée simple au sein d'une voix explicite :

```
\new Voice <<
  \relative { e''4 f8 d e16 f g8 d4 }
  \relative { c''4 d8 b c16 d e8 b4 }
>>
```



Prenez garde que les différents éléments doivent impérativement avoir la même structure rythmique, sous peine de ligature aléatoire et de messages d'avertissement.

Commandes prédéfinies

`\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`, `\oneVoice`.

Voir aussi

Manuel d'initiation : Section "Instanciation explicite des voix" dans *Manuel d'initiation*, Section "Les voix contiennent la musique" dans *Manuel d'initiation*.

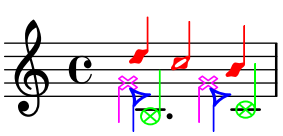
Manuel de notation : Section 7.1.9 [Hampes], page 306, Section 13.1.5 [Portées de percussion], page 502, Section 2.2.2 [Silences invisibles], page 71.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

5.2.2 Styles de voix

Opter pour des couleurs et des têtes de notes spécifiques selon la voix permet de les identifier plus facilement :

```
<<
  \relative { \voiceOneStyle d''4 c2 b4 }
  \\
  \relative { \voiceTwoStyle e'2 e }
  \\
  \relative { \voiceThreeStyle b2. c4 }
  \\
  \relative { \voiceFourStyle g'2 g }
>>
```



La commande `\voiceNeutralStyle` permet de revenir à une présentation normale.

Commandes prédéfinies

`\voiceOneStyle`, `\voiceTwoStyle`, `\voiceThreeStyle`, `\voiceFourStyle`, `\voiceNeutralStyle`.

Voir aussi
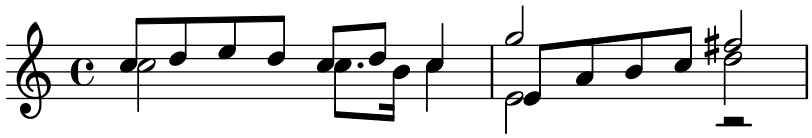
Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "J'entends des voix" dans *Manuel d'initiation*.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

5.2.3 Résolution des collisions

Les notes de hauteur identique appartenant à des voix différentes, même si leur hampe sont opposées, verront leur tête automatiquement fusionner. Les notes dont la tête diffère ou bien qui ont la hampe dans la même direction ne seront pas automatiquement fusionnées. Les silences, lorsqu'ils sont dans une autre voix et à l'opposé des hampes seront décalés verticalement. Vous constaterez, dans l'exemple suivant, que la fusion échoue aux premier et troisième temps de la première mesure, ainsi qu'au premier temps de la deuxième mesure.
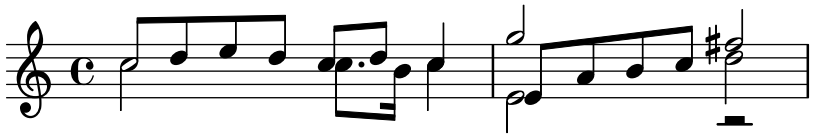
```
<<
  \relative {
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Cependant, vous pouvez fusionner une tête de blanche avec une tête de croche – jamais avec une noire. Les têtes du premier temps de la première mesure ont bien fusionné :
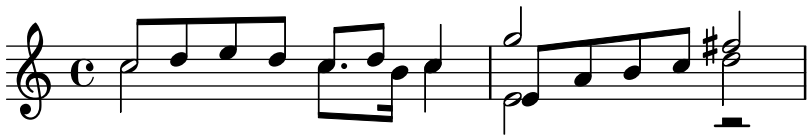
```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
```

```
    s1
    e'8 a b c d2
  }
>>
```



De même, vous pouvez fusionner les têtes de notes pointées et non pointées comme au troisième temps de la première mesure :
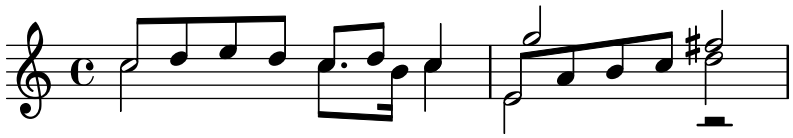
```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```



Lorsque trois notes ou plus s'agglutinent dans un même empilement, `\mergeDifferentlyHeadedOn` ne peut mener à bien la fusion des deux notes qui devraient l'être. Pour obtenir une fusion optimale, appliquez un décalage (`\shift`) à la note qui ne devrait pas fusionner. Ici, on applique un `\shiftOn` pour décaler le *sol* de l'empilement ; le rendement de `\mergeDifferentlyHeadedOn` est alors comme il faut.

```
<<
  \relative {
    \mergeDifferentlyHeadedOn
    \mergeDifferentlyDottedOn
    c''8 d e d c d c4
    \shiftOn
    g'2 fis
  } \\
  \relative {
    c''2 c8. b16 c4
    e,2 r
  } \\
  \relative {
```

```
    \oneVoice
    s1
    e'8 a b c d2
  }
>>
```
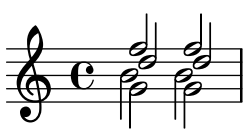


La commande \shiftOn permet, sans pour autant le forcer, un décalage des notes d'une voix en particulier. Une note ou un accord appartenant à cette voix ne seront décalés que si leur hampe menaçait d'entrer en collision avec une hampe appartenant à une autre voix allant dans la même direction. La commande \shiftOff interdit l'apparition de décalage.

Les voix externes – habituellement les voix une et deux – sont affectées de \shiftOff, alors que les voix internes – trois et quatre – sont affectées de \shiftOn. Lorsqu'un décalage intervient, les notes dont les hampes sont ascendantes (voix impaire) iront vers la droite, et les notes à hampe descendante (voix paire) iront vers la gauche.

Voici un exemple qui vous permettra de bien visualiser ce qui se passe en interne.

> **Note :** Lorsqu'il y a trois voix ou plus, prenez garde au fait que l'ordre d'apparition des voix dans votre fichier ne correspond pas à l'ordre vertical des voix tel qu'il apparaîtra sur la portée.

```
\new Staff \relative {
  %% saisie abrégée
  <<
    { f''2  }  % 1 : extrême haute
    \\
    { g,2 }  % 2 : extrême basse
    \\
    { d'2 }  % 3 : intermédiaire haute
    \\
    { b2  }  % 4 : intermédiaire basse
  >>
  %% expansion en interne de ce qui précède
  <<
    \new Voice = "1" { \voiceOne   \shiftOff f'2 }
    \new Voice = "2" { \voiceTwo   \shiftOff g,2 }
    \new Voice = "3" { \voiceThree \shiftOn  d'2 } % décale à droite
    \new Voice = "4" { \voiceFour  \shiftOn  b2  } % décale à gauche
  >>
}
```



Les commandes \shiftOnn et \shiftOnnn sont des niveaux supplémentaires de décalage qui peuvent s'adopter temporairement dans certaines situations complexes – voir Section "Exemple concret" dans *Manuel d'initiation*.

Les têtes de notes ne fusionneront que dans la mesure où leur hampe sont opposées – implicitement parce qu'appartenant aux voix une ou deux, ou bien explicitement.

Commandes prédéfinies

`\mergeDifferentlyDottedOn`, `\mergeDifferentlyDottedOff`, `\mergeDifferentlyHeadedOn`, `\mergeDifferentlyHeadedOff`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff`.
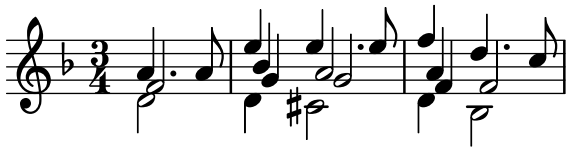
Morceaux choisis

Ajout de voix pour éviter les collisions

Dans certains cas de musique polyphonique complexe, une voix supplémentaire peut permettre d'éviter les risques de collision. Lorsque quatre voix parallèles ne suffisent pas, la fonction Scheme `context-spec-music` permet d'ajouter encore d'autres voix.

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)
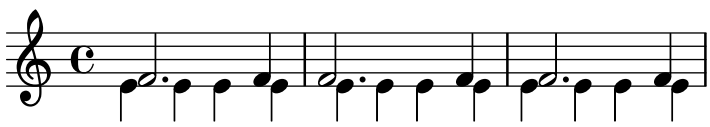
\relative c'' {
  \time 3/4
  \key d \minor
  \partial 2
  <<
    \new Voice  {
      \voiceOne
      a4. a8
      e'4 e4. e8
      f4 d4. c8
    }
    \new Voice {
      \voiceTwo
      d,2
      d4 cis2
      d4 bes2
    }
    \new Voice {
      \voiceThree
      f'2
      bes4 a2
      a4 s2
    }
    \new Voice {
      \voiceFive
      s2
      g4 g2
      f4 f2
    }
  >>
}
```



Déplacement des notes pointées dans une polyphonie

Une note pointée appartenant à la voix supérieure d'une portée polyphonique sera par défaut décalée vers la droite afin d'éviter les collisions avec les autres voix. Ce comportement peut être modifié par un réglage de la propriété `prefer-dotted-right` de l'objet graphique `NoteCollision`.
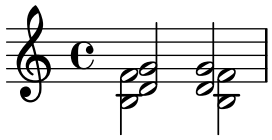
```
\new Staff \relative c' <<
  {
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##f
    f2. f4
    \override Staff.NoteCollision.prefer-dotted-right = ##t
    f2. f4
  }
  \\
  { e4 e e e e e e e e e e }
>>
```



Décalage horizontal forcé

Quand LilyPond est dépassé, la propriété `force-hshift` de l'objet `NoteColumn` et des silences à hauteur déterminée peuvent s'avérer utiles pour dicter au programme les choix de placement. On travaille ici en espace de portée.

```
\relative c' <<
  {
    <d g>2 <d g>
  }
  \\
  {
    <b f'>2
    \once \override NoteColumn.force-hshift = 1.7
    <b f'>2
  }
>>
```



Voir aussi

Glossaire musicologique : Section "polyphonie" dans *Glossaire*.

Manuel d'initiation : Section "Exemple concret" dans *Manuel d'initiation*, Section "Les voix contiennent la musique" dans *Manuel d'initiation*, Section "Notes simultanées" dans *Manuel d'initiation*.

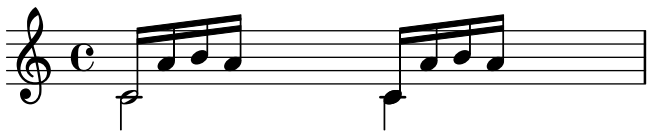
Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "NoteColumn" dans *Référence des propriétés internes*, Section "NoteCollision" dans *Référence des propriétés internes*, Section "RestCollision" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Une clause `\override NoteColumn.ignore-collision = ##t` aura pour effet une fusion incorrecte des têtes de note différentes à partir de voix différentes.
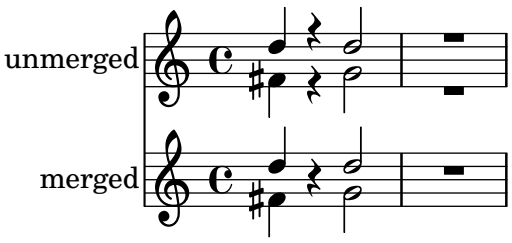
```
\mergeDifferentlyHeadedOn
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
\override NoteColumn.ignore-collision = ##t
<< \relative { c'16 a' b a } \\ \relative { c'2 } >>
```



5.2.4 Fusion de silences

Il est d'usage, lorsque plusieurs voix cohabitent, de fusionner les silences qui apparaissent simultanément dans différentes parties. Ceci peut s'obtenir à l'aide du `Merge_rests_engraver`.

```
voiceA = \relative { d''4 r d2 | R1 | }
voiceB = \relative { fis'4 r g2 | R1 | }
\score {
  <<
    \new Staff \with {
      instrumentName = "unmerged"
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
    \new Staff \with {
      instrumentName = "merged"
      \consists Merge_rests_engraver
    }
    <<
      \new Voice { \voiceOne \voiceA }
      \new Voice { \voiceTwo \voiceB }
    >>
  >>
}
```



La propriété de contexte `suspendRestMerging`, lorsque activée par un `#t`, permet de suspendre temporairement la fusion des silences.

5.2.5 Regroupement automatique de parties

Le regroupement automatique de parties vous permet de fusionner deux pupitres sur une seule portée, ceci dans le but de créer des partitions d'orchestre. Lorsque les deux parties sont iden-

tiques sur une certaine durée, une seule s'affiche. Lorsqu'elles diffèrent, deux voix séparées apparaissent, avec des hampes dont la direction est gérée automatiquement. Vous pouvez aussi identifier et faire ressortir les solos et parties *a due*.

Voici la syntaxe qui permet de combiner des parties :
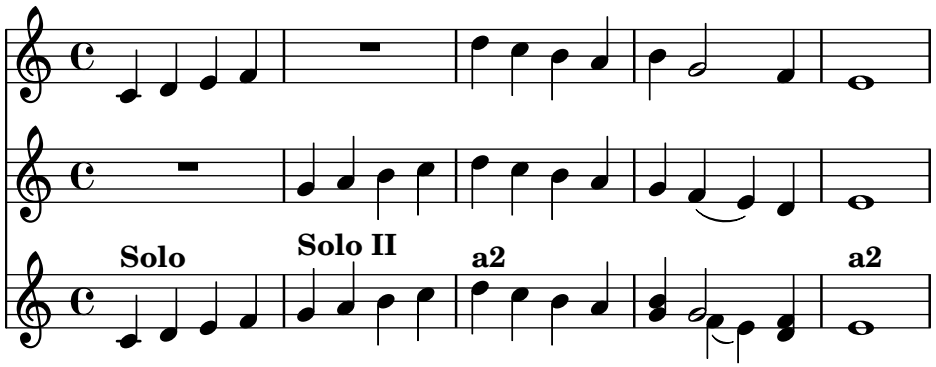
```
\partCombine expression_musicale_1 expression_musicale_2
```

L'exemple suivant illustre les fonctionnalités élémentaires du combinateur de parties : positionner les parties sur une portée, gérer la direction des hampes et de la polyphonie. Les identifiants sont les mêmes pour la combinaison et les parties séparées.

```
instrumentOne = \relative {
  c'4 d e f |
  R1 |
  d'4 c b a |
  b4 g2 f4 |
  e1 |
}

instrumentTwo = \relative {
  R1 |
  g'4 a b c |
  d4 c b a |
  g4 f( e) d |
  e1 |
}

<<
  \new Staff \instrumentOne
  \new Staff \instrumentTwo
  \new Staff \partCombine \instrumentOne \instrumentTwo
>>
```



Les notes de la troisième mesure n'apparaissent qu'une seule fois, alors qu'elles ont été spécifiées deux fois (une fois dans chacune des parties). La direction des hampes et des liaisons de tenue ou de phrasé est gérée automatiquement, selon qu'il s'agisse d'un solo ou d'un unisson. La première partie, dont le contexte s'appellera `one`, aura toujours ses hampes dirigées vers le haut et sera notée « Solo », alors que la deuxième, appelée `two`, aura des hampes vers le bas et sera notée « Solo II ». Les parties à l'unisson seront par défaut estampillées d'un « a2 ».

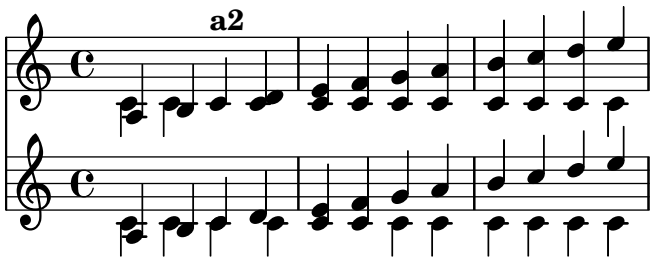
Par défaut, le combinateur fusionnera deux notes de même hauteur en une note *a due*, regroupera en accord les notes de même rythme et dont l'intervalle est inférieur à une neuvième, enfin isolera les notes séparées de plus d'une neuvième (ou si les voix se croisent) dans des voix distinctes. Ceci peut s'adapter à l'aide d'une paire de nombres fournie en argument optionnel

à la commande \partCombine : le premier nombre spécifie l'intervalle à partir duquel les notes seront combinées (0 par défaut) et le second celui à partir duquel les notes seront placées dans des voix séparées. Un second élément de cette paire à zéro obligera le combinateur à séparer les notes dès la seconde ; s'il est à un, elles seront séparées à partir de la tierce, et ainsi de suite.

```
instrumentOne = \relative {
  a4 b c d |
  e f g a |
  b c d e |
}

instrumentTwo = \relative {
  c'4 c c c |
  c c c c |
  c c c c |
}

<<
  \new Staff \partCombine \instrumentOne \instrumentTwo
  \new Staff \partCombine #'(2 . 3) \instrumentOne \instrumentTwo
>>
```



LilyPond interprète dans un contexte Voice chacun des arguments fournis à \partCombine. Si vous travaillez avec des octaves relatives, spécifiez \relative dans chacune des expressions musicales, comme ceci :

```
\partCombine
  \relative ... expression_musicale_1
  \relative ... expression_musicale_2
```

Une section \relative à l'extérieur du \partCombine restera sans effet sur les hauteurs de *expression_musicale_1* ou de *expression_musicale_2*.

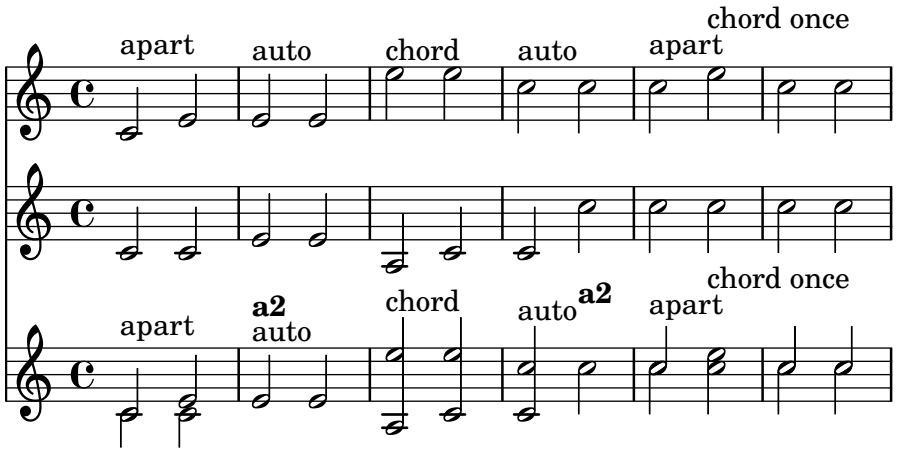
En matière d'édition professionnelle, les voix sont souvent maintenues séparément et sur une durée conséquente, bien que les notes des différentes voix soient les mêmes et pourraient donc être présentées à l'unisson. Dans la mesure où \partCombine considère les notes séparément, combiner des notes en accord ou indiquer un solo ne serait pas optimal en pareil cas. LilyPond dispose alors de certaines commandes qui permettent d'influencer le comportement de la fonction \partCombine. Ces commandes peuvent se voir préfixées d'un \once de sorte à n'affecter que la note qui les suit directement dans l'expression musicale.

- \partCombineApart maintient les notes dans des voix séparées même si elles peuvent se combiner en accord ou en unisson.
- \partCombineChords combine les notes en accords.
- \partCombineUnisono combine les voix en un « unisson ».
- \partCombineSoloI affiche exclusivement la première voix et l'affuble d'un « Solo ».
- \partCombineSoloII affiche exclusivement la deuxième voix et l'affuble d'un « Solo ».

- \partCombineAutomatic annule les effets des dérogations précédentes et active le comportement standard de la fonction \partCombine.

```
instrumentOne = \relative c' {
  \partCombineApart c2^"apart" e |
  \partCombineAutomatic e2^"auto" e |
  \partCombineChords e'2^"chord" e |
  \partCombineAutomatic c2^"auto" c |
  \partCombineApart c2^"apart"
    \once \partCombineChords e^"chord once" |
  c2 c |
}
instrumentTwo = \relative {
  c'2 c |
  e2 e |
  a,2 c |
  c2 c' |
  c2 c |
  c2 c |
}

<<
  \new Staff { \instrumentOne }
  \new Staff { \instrumentTwo }
  \new Staff { \partCombine \instrumentOne \instrumentTwo }
>>
```



Utilisation de \partCombine et paroles

La commande \partCombine n'est pas conçue pour traiter des paroles ; si l'une des voix est explicitement nommée, afin de lui adjoindre des paroles, le combinateur de parties cessera de fonctionner. Toutefois, le recours à un contexte NullVoice permet d'obtenir les effets escomptés – voir Section 9.2.6 [Polyphonie et paroles communes], page 386.

Morceaux choisis

Combinaison de deux parties sur une même portée

L'outil de combinaison de parties (la commande \partCombine) permet d'avoir deux parties différentes sur une même portée. LilyPond ajoute automatiquement des indications textuelles, telles que « solo » ou « a2 ». Si votre intention n'est que de fusionner les parties, sans ajouter de texte, assignez faux à la propriété printPartCombineTexts.

Dans le cas de partitions vocales, et plus particulièrement d'hymnes, ces « solo/a2 » ne sont d'aucune utilité, aussi vaut-il mieux les désactiver. Dans le cas où il y aurait alternance entre *solo* et *tutti*, il vaut mieux faire appel à de la musique polyphonique standard.

Voici trois moyens d'imprimer deux parties sur une même portée : en polyphonie normale, avec \partCombine sans indication supplémentaire, et avec \partCombine commentée.

```
musicUp = \relative c'' {
  \time 4/4
  a4 c4.( g8) a4 |
  g4 e' g,( a8 b) |
  c b a2.
}

musicDown = \relative c'' {
  g4 e4.( d8) c4 |
  r2 g'4( f8 e) |
  d2 \stemDown a
}

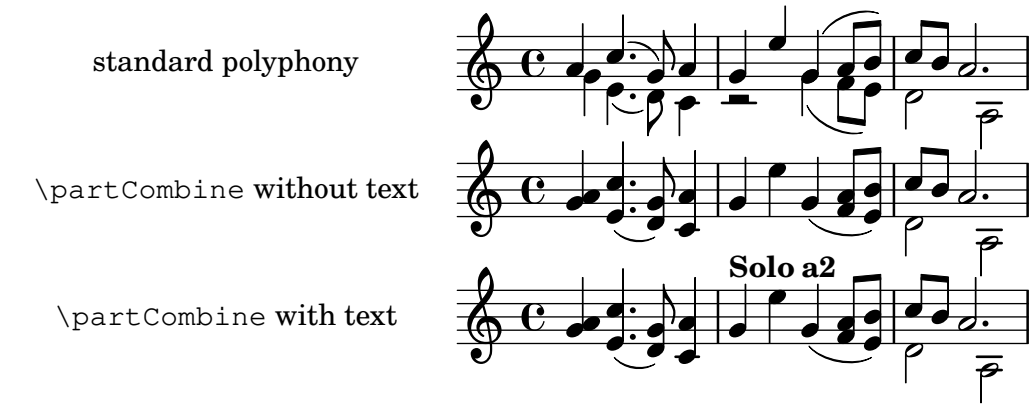
\score {
  <<
    \new Staff \with {
      instrumentName = "standard polyphony"
    } << \musicUp \\ \musicDown >>

    \new Staff \with {
      instrumentName =
        \markup { \typewriter "\\partCombine" without text}
      printPartCombineTexts = ##f
    } \partCombine \musicUp \musicDown

    \new Staff \with {
      instrumentName =
        \markup { \typewriter "\\partCombine" with text}
    } \partCombine \musicUp \musicDown
  >>

  \layout {
    indent = 6.0\cm
    \context {
      \Score
      % Setting this to a large value avoids a bar line at the
      % beginning that would connect the three staves otherwise.
      \override SystemStartBar.collapse-height = 30
    }
```
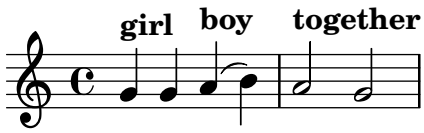
```
  }
}
```



Modification des indications de parties combinées

Lorsque vous regroupez automatiquement des parties, vous pouvez modifier le texte qui sera affiché pour les solos et pour les parties à l'unisson :

```
\new Staff <<
  \set Staff.soloText = "girl"
  \set Staff.soloIIText = "boy"
  \set Staff.aDueText = "together"
  \partCombine
    \relative c'' {
      g4 g r r
      a2 g
    }
    \relative c'' {
      r4 r a( b)
      a2 g
    }
>>
```



Voir aussi

Glossaire musicologique : Section "a due" dans *Glossaire*, Section "partie" dans *Glossaire*.

Manuel de notation : Section 6.3 [Écriture de parties séparées], page 270.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PartCombineMusic" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les différentes commandes `\partCombine...` ne prennent en charge que deux voix. De la même manière, le combinateur n'est pas conçu pour travailler avec des paroles ; il s'arrête dès qu'il est explicitement fait appel à l'une des voix pour y attacher des paroles.

`\partCombine...` ne peut s'inscrire ni dans un bloc `\tuplet` ni dans un bloc `\relative`.

Lorsque printPartCombineTexts est actif et que les deux voix jouent souvent les mêmes notes, le combinateur peut afficher a2 plus d'une fois par mesure.

\partCombine n'examine que l'attaque des notes. Il n'est donc pas en mesure de déterminer si une note attaquée précédemment est encore jouée ou non, ce qui peut engendrer quelques problèmes, entre autres des indications de « Solo » ou « Unisono » incorrectement placées.

\partCombine conserve les objets étendus (liaisons, soufflets, etc.) dans la même voix de sorte à éviter qu'ils soient improprement ou pas du tout imprimés lorsque leur départ ou terminaison est dans une voix différente.

En interne, \partCombine interprète les deux arguments en tant que Voices, dénommées one et two, puis décide de quand les parties seront fusionnées. Par conséquent, si les arguments changent pour d'autres noms de contexte Voice, les événements qu'ils contiendraient seront ignorés.

Dans la mesure où \partCombine fonctionne en deux passes, il faut faire attention à ne pas mélanger les étapes. Par exemple, ce code

```
one = { e''2 \tag #'score f''
            \tag #'part fis'' g''1 }
two = { e''2 d'' g'1 }

\removeWithTag #'score \partCombine \one \two
```

échoue parce que, au premier passage, rien n'indique que l'on va supprimer la musique balisée et, par voie de conséquence, l'information enregistrée est en contradiction avec la musique lors du second passage. Lorsque l'on veut utiliser \partCombine avec du filtrage, des répétitions expansées ou d'autres transformations, il est primordial que ces transformations aient lieu avant, de telle sorte que la musique soit dans sont état définitif avant la première passe, comme par exemple

```
...
\partCombine
  \removeWithTag #'score \one
  \removeWithTag #'score \two
```

Certaines considérations apparaissent aussi dans les chapitres Section 12.1.3 [Tablatures par défaut], page 446, et Section 2.4.1 [Barres de ligature automatiques], page 104.

5.2.6 Saisie de musique en parallèle

On peut écrire plusieurs voix de façon entremêlée. La fonction \parallelMusic prend en charge une liste des variables à créer, ainsi qu'une expression musicale. Le contenu des différentes mesures de l'expression musicale deviennent les valeurs des variables respectives que vous pourrez ensuite utiliser pour imprimer la partition.

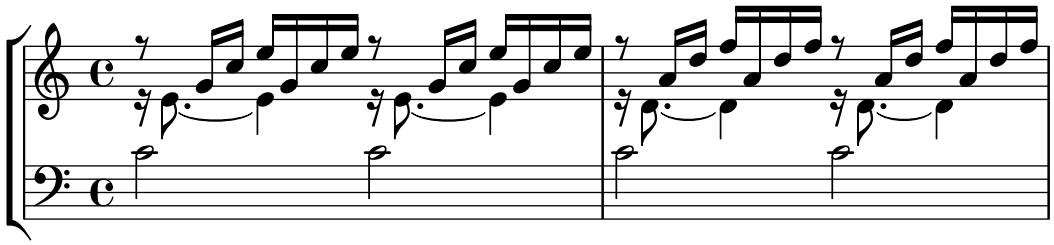
> **Note :** Les contrôles de barre de mesure | sont obligatoires et les mesures doivent être de longueur identique.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Bar 1
  r8 g'16 c'' e'' g' c'' e'' r8 g'16 c'' e'' g' c'' e'' |
  r16 e'8.~   4             r16 e'8.~   4               |
  c'2                       c'2                         |

  % Bar 2
  r8 a'16 d'' f'' a' d'' f'' r8 a'16 d'' f'' a' d'' f'' |
```

```
  r16 d'8.~   4              r16 d'8.~   4              |
  c'2                        c'2                        |

}
\new StaffGroup <<
  \new Staff << \voiceA \\ \voiceB >>
  \new Staff { \clef bass \voiceC }
>>
```

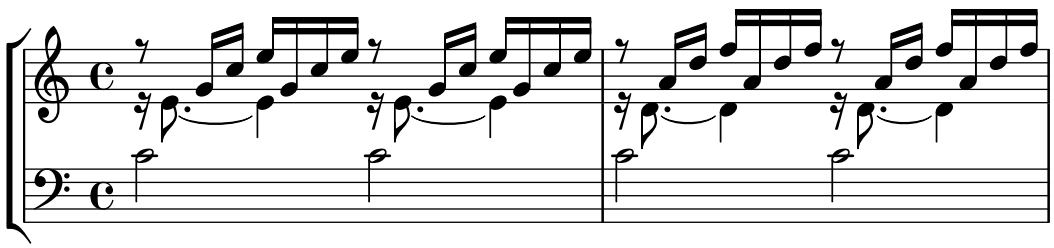


Vous pouvez travailler en mode relatif. Notez cependant que la commande `\relative` n'apparaît pas au sein du bloc `\parallelMusic`. Le calcul des hauteurs relatives s'effectue voix par voix, et non au fil des lignes saisies ; en d'autres termes, les notes de la `voiceA` ignorent tout de celles de la `voiceB`.

```
\parallelMusic voiceA,voiceB,voiceC {
  % Bar 1
  r8 g16 c e g, c e r8 g,16 c e g, c e  |
  r16 e8.~ 4        r16 e8.~  4         |
  c2                c                   |

  % Bar 2
  r8 a,16 d f a, d f r8 a,16 d f a, d f |
  r16 d8.~  4        r16 d8.~  4        |
  c2                 c                  |

 }
\new StaffGroup <<
  \new Staff << \relative c'' \voiceA \\ \relative c' \voiceB >>
  \new Staff \relative c' { \clef bass \voiceC }
>>
```



Ceci fonctionne bien avec la musique pour piano. L'exemple suivant affecte quatre mesures à quatre variables :
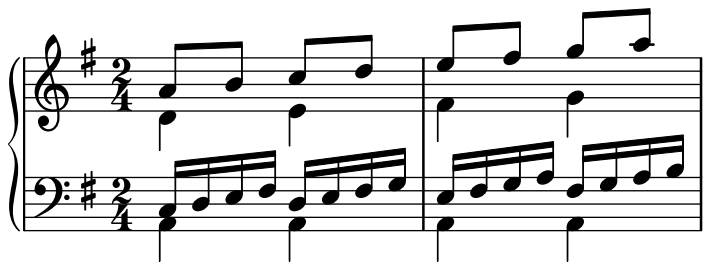
```
global = {
  \key g \major
  \time 2/4
}

\parallelMusic voiceA,voiceB,voiceC,voiceD {
  % Bar 1
  a8    b     c   d     |
  d4          e         |
  c16 d e fis d e fis g  |
  a4          a         |

  % Bar 2
  e8      fis  g     a   |
  fis4         g         |
  e16 fis g  a fis g a b  |
  a4           a         |

  % Bar 3 ...
}

\score {
  \new PianoStaff <<
     \new Staff {
       \global
       <<
         \relative c'' \voiceA
         \\
         \relative c'  \voiceB
       >>
     }
     \new Staff {
       \global \clef bass
       <<
         \relative c \voiceC
         \\
         \relative c \voiceD
       >>
     }
  >>
}
```

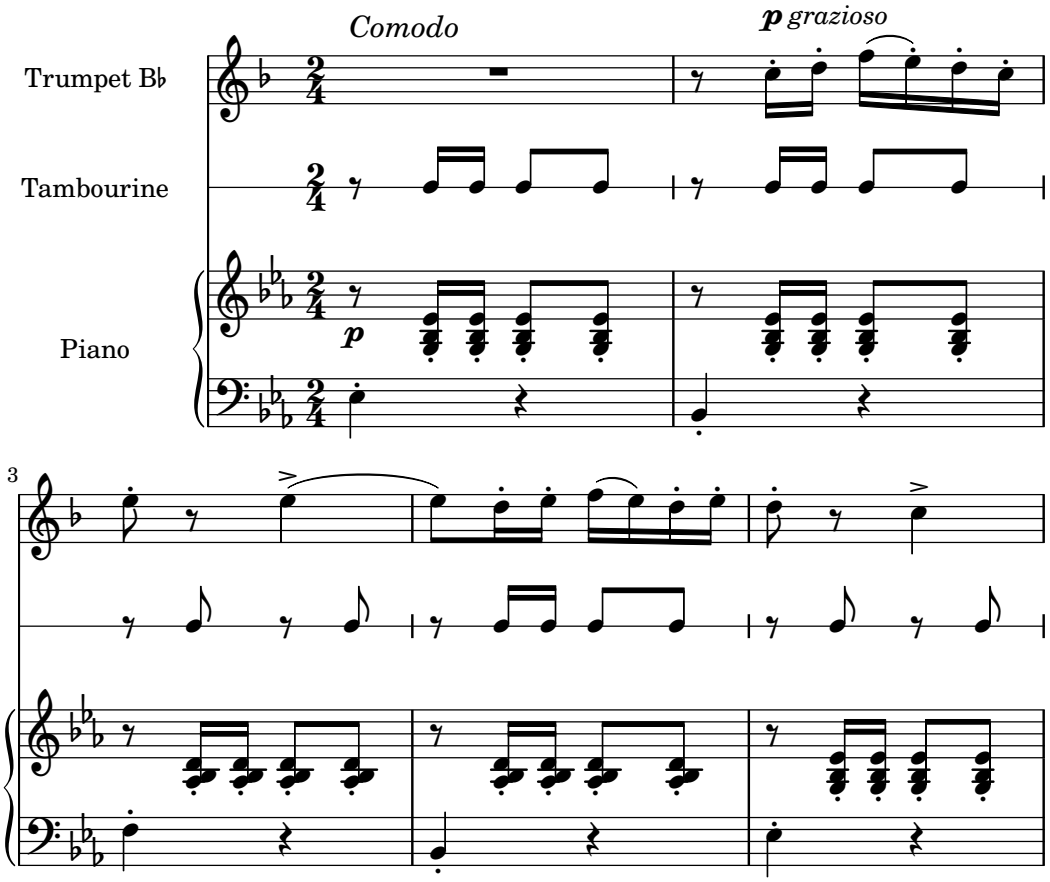


Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Morceaux choisis : Section "Notation simultanée" dans *Morceaux choisis*.

6 Notation sur la portée



Cette section aborde les détails de gravure de la portée, la réalisation de partitions comprenant plusieurs portées et l'ajout d'indications globales d'exécution, présentes sur certaines portées seulement.

6.1 Gravure des portées
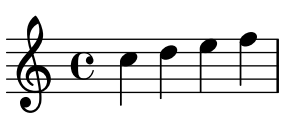
Nous allons voir ici comment créer des portées et comment les regrouper.

6.1.1 Initialisation de nouvelles portées

Les *portées* – en anglais *staff* (*staves* au pluriel) – sont créées à l'aide des commandes `\new` ou `\context`. Pour de plus amples détails, consultez Section 33.2 [Création et référencement d'un contexte], page 763.

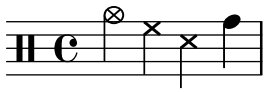
Le contexte de portée standard s'appelle `Staff` :

```
\new Staff \relative { c''4 d e f }
```



Le contexte `DrumStaff` crée une portée à cinq lignes correspondant à une batterie traditionnelle et chacun des instruments est représenté par un symbole spécifique. Les éléments sont saisis en mode batterie, initialisé par la commande `\drummode`, chaque composante étant spécifiée par son nom. Pour de plus amples détails, consultez Section 13.1.5 [Portées de percussion], page 502.

```
\new DrumStaff {
  \drummode { cymc hh ss tomh }
}
```



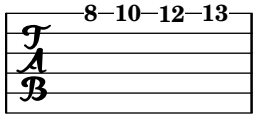
Un RhythmicStaff est composé d'une portée à ligne unique chargée de présenter les valeurs rythmiques saisies. Seules sont imprimées les durées. Pour de plus amples détails, consultez Section 2.3.8 [Gravure de lignes rythmiques], page 101.

```
\new RhythmicStaff { c4 d e f }
```



Un TabStaff crée une portée de tablature correspondant aux six cordes d'une guitare standard. Pour de plus amples détails, consultez Section 12.1.3 [Tablatures par défaut], page 446.

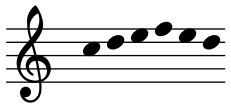
```
\new TabStaff \relative { c''4 d e f }
```



LilyPond dispose aussi de deux contextes dédiés à la musique ancienne : MensuralStaff et VaticanaStaff. Ils sont abordés plus en détails au chapitre Section 17.2.1 [Contextes prédéfinis], page 554.

Le contexte GregorianTranscriptionStaff permet d'obtenir des éditions modernes du grégorien. Les *divisions* apparaîtront, mais pas les barres de mesure.

```
\new GregorianTranscriptionStaff \relative { c''4 d e f e d }
```



Vous pourrez toujours définir d'autres contextes de portée selon vos besoins, en suivant les indications fournies au chapitre Section 33.6 [Définition de nouveaux contextes], page 776.

Voir aussi

Glossaire musicologique : Section “staff” dans *Glossaire*, Section “portées” dans *Glossaire*.

Manuel de notation : Section 17.3.1 [Contextes de musique mensurale], page 556, Section 17.4.1 [Contextes du chant grégorien], page 569, Section 17.2.1 [Contextes prédéfinis], page 554, Section 33.2 [Création et référencement d'un contexte], page 763, Section 33.6 [Définition de nouveaux contextes], page 776, Section 2.3.8 [Gravure de lignes rythmiques], page 101, Section 13.1.5 [Portées de percussion], page 502, Section 6.2.1 [Symbole de la portée], page 258, Section 12.1.3 [Tablatures par défaut], page 446.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

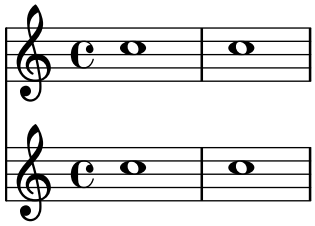
Référence des propriétés internes : Section “Staff” dans *Référence des propriétés internes*, Section “DrumStaff” dans *Référence des propriétés internes*, Section “GregorianTranscriptionStaff” dans *Référence des propriétés internes*, Section “RhythmicStaff” dans *Référence des propriétés internes*, Section “TabStaff” dans *Référence des propriétés internes*, Section “MensuralStaff” dans *Référence des propriétés internes*, Section “VaticanaStaff” dans *Référence des propriétés internes*, Section “StaffSymbol” dans *Référence des propriétés internes*.

6.1.2 Regroupement de portées

LilyPond dispose de différents contextes permettant de regrouper des portées individuelles et d'obtenir ainsi des « systèmes ». Chacun de ces contextes définira le style de regroupement, avec son signe particulier en début de portée et ses règles de gestion des barres de mesure.

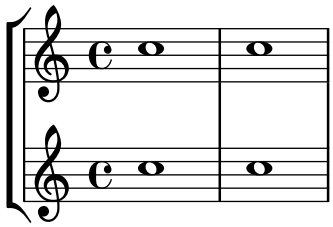
Lorsqu'aucun contexte particulier n'est spécifié, les propriétés suivantes s'appliqueront par défaut : les portées du groupe ne sont pas reliées, hormis par une simple ligne verticale en début de ligne, et les barres de mesure sont indépendantes.

```
<<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



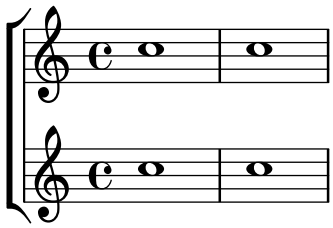
Dans un `StaffGroup`, le groupe de portées est signifié par un crochet, et les barres de mesure sont d'un seul tenant.

```
\new StaffGroup <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



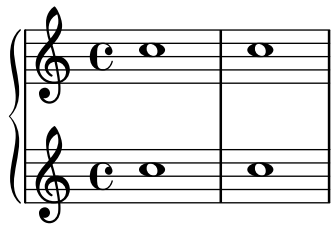
Dans un `ChoirStaff`, le groupe de portées est signifié par un crochet sur la gauche, et les barres de mesure sont individuelles.

```
\new ChoirStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



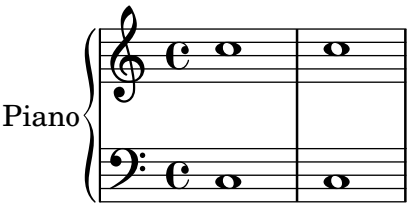
Dans un `GrandStaff`, le groupe de portées est signifié par une accolade sur la gauche, et les barres de mesure sont d'un seul tenant.

```
\new GrandStaff <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { c''1 c }
>>
```



Le contexte PianoStaff est identique au GrandStaff, sauf que ses portées ne peuvent être masquées que conjointement, jamais individuellement – voir Section 6.2.3 [Masquage de portées], page 265.

```
\new PianoStaff \with { instrumentName = "Piano" }
  <<
  \new Staff \relative { c''1 c }
  \new Staff \relative { \clef bass c1 c }
>>
```

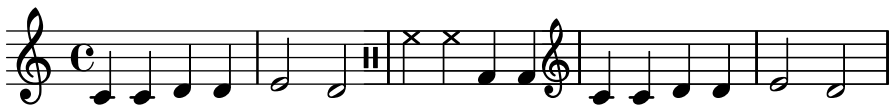


Le contexte OneStaff est un regroupement qui place des portées différentes dans le même alignement vertical. L'exemple suivant montre trois portées partageant le même espace. Ici, le Time_signature_engraver a été déplacé des portées au contexte OneStaff pour empêcher sa répétition à chaque portée.

```
\layout {
  \context {
    \OneStaff
    \consists Time_signature_engraver
  }
  \context {
    \Staff
    \remove Time_signature_engraver
  }
  \context {
    \DrumStaff
    \remove Time_signature_engraver
  }
}

\new OneStaff {
  \new Staff {
    c'4 4 d'4 4 e'2 d'
  }
  \drums {
```

```
    hihat4 hh bassdrum bd
  }
  \new Staff {
    c'4 4 d'4 4 e'2 d'
  }
}
```



Les barres de mesure au début de chaque système adoptent l'un des styles `SystemStartBar`, `SystemStartBrace` ou `SystemStartBracket`. Dans chaque contexte, seul l'un de ces styles est utilisé, et c'est la propriété `systemStartDelimiter` qui détermine lequel. Un quatrième style, `SystemStartSquare`, doit quant à lui être spécifié de manière explicite.

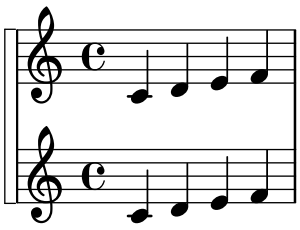
Vous pouvez aussi créer vos propres contextes de regroupement, comme l'explique Section 33.6 [Définition de nouveaux contextes], page 776.

Morceaux choisis

Indication de regroupement de portées par un rectangle

Un regroupement de portées sera indiqué par un simple rectangle – `SystemStartSquare` – en début de ligne dès lors que vous le mentionnerez explicitement au sein d'un contexte `StaffGroup` ou `ChoirStaff`.

```
\score {
  \new StaffGroup { <<
  \set StaffGroup.systemStartDelimiter = #'SystemStartSquare
    \new Staff { c'4 d' e' f' }
    \new Staff { c'4 d' e' f' }
  >> }
}
```
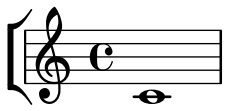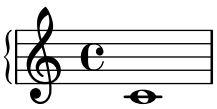


Indicateur de regroupement et portée unique

Lorsque, dans des regroupements de type `ChoirStaff` ou `StaffGroup`, une seule portée est active, aucune indication n'est donnée en début de ligne. Affecter à la propriété `collapse-height` un nombre de lignes inférieur à celui de la portée permet de modifier ce comportement par défaut.

Notez bien que dans le cas des `PianoStaff` et `GrandStaff`, pour lesquels le délimiteur de système est une accolade et non un crochet, il ne s'agit pas de la même propriété – voir le deuxième système de l'exemple.

```
\score {
  \new StaffGroup <<
    % Must be lower than the actual number of staff lines
    \override StaffGroup.SystemStartBracket.collapse-height = 4
    \override Score.SystemStartBar.collapse-height = 4
    \new Staff {
      c'1
    }
  >>
}
\score {
  \new PianoStaff <<
    \override PianoStaff.SystemStartBrace.collapse-height = 4
    \override Score.SystemStartBar.collapse-height = 4
    \new Staff {
      c'1
    }
  >>
}
```





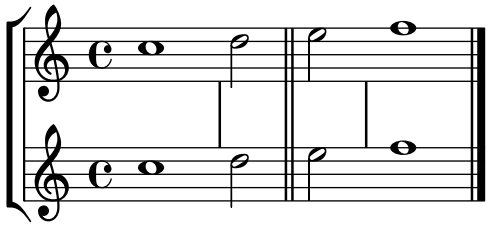
Présentation à l'ancienne (barres de mesure entre les portées)

En musique mensurale, les barres de mesure ne traversent pas les portées. Pour obtenir ce résultat il faudra définir `measureBarType` à `"-span|"` et utiliser un regroupement de portées permettant l'extension des barres entre les portées, tel un `StaffGroup`.

```
\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}

music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}

\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>
```



Voir aussi

Glossaire musicologique : Section “accolade” dans *Glossaire*, Section “crochet” dans *Glossaire*, Section “système” dans *Glossaire*.

Manuel de notation : Section 33.6 [Définition de nouveaux contextes], page 776, Section 6.3.1 [Noms d’instrument], page 270.
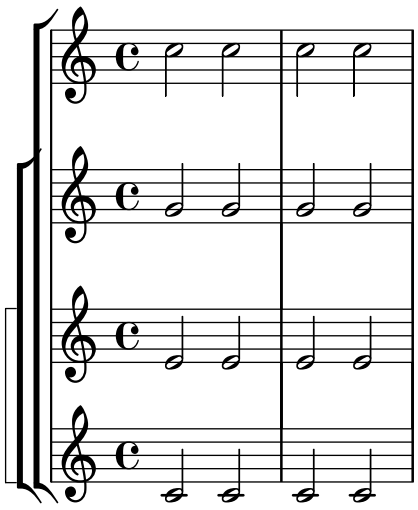
Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Staff” dans *Référence des propriétés internes*, Section “StaffGroup” dans *Référence des propriétés internes*, Section “ChoirStaff” dans *Référence des propriétés internes*, Section “GrandStaff” dans *Référence des propriétés internes*, Section “PianoStaff” dans *Référence des propriétés internes*, Section “SystemStartBar” dans *Référence des propriétés internes*, Section “SystemStartBrace” dans *Référence des propriétés internes*, Section “SystemStartBracket” dans *Référence des propriétés internes*, Section “SystemStartSquare” dans *Référence des propriétés internes*.

6.1.3 Imbrication de regroupements de portées

Les accolades et crochets qui délimitent les systèmes peuvent être imbriqués en profondeur. Chaque niveau inférieur aura son propre délimiteur, en plus de celui du niveau supérieur.

```
\new StaffGroup <<
  \new Staff \relative { c''2 c | c2 c }
  \new StaffGroup <<
    \new Staff \relative { g'2 g | g2 g }
    \new StaffGroup \with {
      systemStartDelimiter = #'SystemStartSquare
    }
    <<
      \new Staff \relative { e'2 e | e2 e }
      \new Staff \relative { c'2 c | c2 c }
    >>
  >>
>>
```



Vous pouvez aussi créer vos propres contextes d'imbrication, comme l'explique Section 33.6 [Définition de nouveaux contextes], page 776.

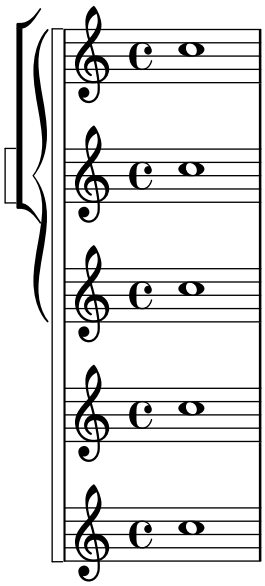
Morceaux choisis

Imbrications de regroupements de portées

La propriété de contexte `systemStartDelimiterHierarchy` permet de créer des regroupements imbriqués complexes. Dans le cadre d'un `StaffGroup`, elle prend en argument la liste alphabétique des sous-groupes à hiérarchiser. Chaque sous-groupe peut être affublé d'un délimiteur particulier. Chacun des regroupements intermédiaires doit être borné par des parenthèses. Bien que des éléments de la liste puissent être omis, le premier délimiteur embrassera toujours l'intégralité des portées. Vous disposez des quatre délimiteurs `SystemStartBar`, `SystemStartBracket`, `SystemStartBrace` et `SystemStartSquare`.

```
\new StaffGroup
\relative c'' <<
  \override StaffGroup.SystemStartSquare.collapse-height = 4
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare
          (SystemStartBrace
            (SystemStartBracket a
              (SystemStartSquare b))
            c)
          d)

  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



Voir aussi

Manuel de notation : Section 33.6 [Définition de nouveaux contextes], page 776, Section 6.3.1 [Noms d'instrument], page 270, Section 6.1.2 [Regroupement de portées], page 250.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

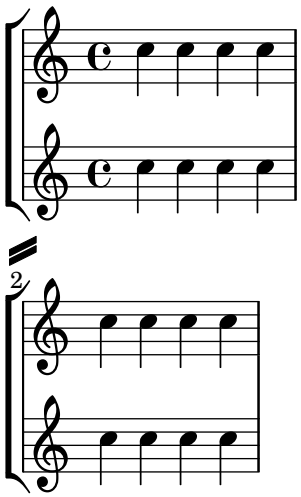
Référence des propriétés internes : Section "StaffGroup" dans *Référence des propriétés internes*, Section "ChoirStaff" dans *Référence des propriétés internes*, Section "SystemStartBar"

dans *Référence des propriétés internes*, Section “SystemStartBrace” dans *Référence des propriétés internes*, Section “SystemStartBracket” dans *Référence des propriétés internes*, Section “SystemStartSquare” dans *Référence des propriétés internes*.

6.1.4 Séparation des systèmes

Le nombre de systèmes peut varier d'une page à l'autre ; vous pouvez, en pareil cas, rendre plus évidente la séparation entre les systèmes en l'indiquant visuellement. Ce « séparateur » est absent par défaut, mais vous pouvez l'activer par une option au sein du bloc `\paper`.

```
\book {
  \score {
    \new StaffGroup <<
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
      \new Staff {
        \relative {
          c''4 c c c
          \break
          c4 c c c
        }
      }
    >>
  }
  \paper {
    system-separator-markup = \slashSeparator
    tagline = ##f
  }
}
```



Voir aussi

Manuel de notation : Chapitre 26 [Mise en forme de la page], page 691.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.
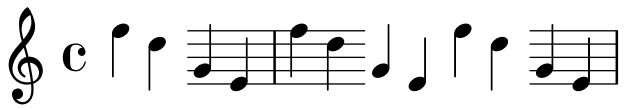
6.2 Modification de portées individuelles

Cette section explique le réglage de la gravure de chaque portée, comme la taille de portée ou le nombre de lignes ; sont aussi décrits la suspension et la reprise de portées, ainsi que les portées d'*ossia*.

6.2.1 Symbole de la portée

Les commandes `\stopStaff` et `\startStaff` permettent respectivement de clôturer et (re)démarrer une portée n'importe où dans une partition.

```
\relative {
  \stopStaff f''4 d \startStaff g, e
  f'4 d \stopStaff g, e
  f'4 d \startStaff g, e
}
```



Commandes prédéfinies
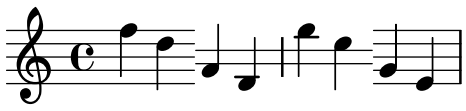
`\startStaff`, `\stopStaff`.

Les notes, nuances, etc. sont regroupées dans un assemblage de lignes horizontales, que l'on nomme la portée (en anglais *staff*, et *staves* au pluriel). Dans LilyPond, ces lignes sont dessinées au moyen d'un objet graphique (*grob*) à part entière, nommé `StaffSymbol` – symbole de portée. Modifier les propriétés d'un `StaffSymbol` changera l'apparence de la portée, dès lors qu'elles auront été définies avant de créer la portée en question.

Vous pouvez modifier le nombre de lignes d'une portée :
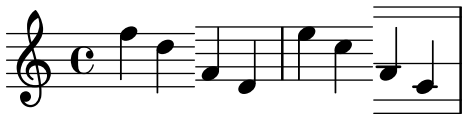
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-count = 2
  \startStaff g, e |

  f'4 d \stopStaff
  \revert Staff.StaffSymbol.line-count
  \startStaff g, e |
}
```



Le positionnement de chacune des lignes de la portée est modifiable. Une liste de nombres détermine le positionnement de chaque ligne. Le `0` correspond à la ligne médiane d'une portée habituelle, pour laquelle la liste est donc `(-4 -2 0 2 4)`. Une ligne sera donc imprimée pour chaque valeur exprimée ; le nombre de lignes, ainsi que leur position dans la portée, peut donc se modifier à l'aide d'une seule commande. C'est ainsi que la propriété `line-count` est ignorée dès lors qu'est activée `line-positions`
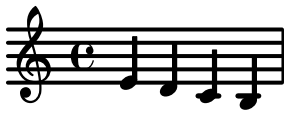
```
\relative {
  f''4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(1 3 5 -1 -3)
  \startStaff g, e |
  f'4 d \stopStaff
  \override Staff.StaffSymbol.line-positions = #'(8 6.5 -6 -8 -0.5)
  \startStaff g, e |
}
```



Afin de préserver l'orientation habituelle des hampes – ascendantes dans la partie inférieure de la portée, descendantes dans la partie supérieure – la ligne (ou l'interligne) centrale de la portée personnalisée devra être en phase avec avec la ligne médiane d'une portée classique (0). La position de la clef et celle du do médium demanderont parfois un ajustement afin d'être en phase avec cette nouvelle portée. Pour plus d'explications, reportez-vous aux exemples du chapitre Section 1.3.1 [Clefs], page 20.
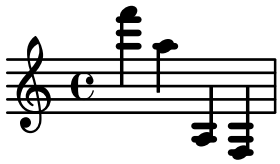
Lorsque vous modifierez l'épaisseur des lignes, gardez à l'esprit que les lignes supplémentaires et les hampes seront aussi modifiées.

```
\new Staff \with {
  \override StaffSymbol.thickness = 3
}
{ e4 d c b }
```



L'épaisseur des lignes supplémentaires (*ledger lines*) peut être déterminée indépendamment des lignes de la portée.

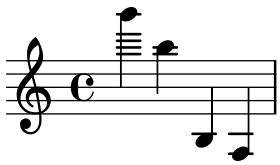
```
\new Staff \with {
  \override StaffSymbol.thickness = 2
  \override StaffSymbol.ledger-line-thickness = #'(0.5 . 0.4)
} \relative {
  f'''4 a, a,, f
}
```



La première valeur est multipliée par l'épaisseur de ligne de portée, la seconde par l'espace d'interligne ; ces deux valeurs sont alors combinées pour donner la nouvelle épaisseur des lignes supplémentaires.
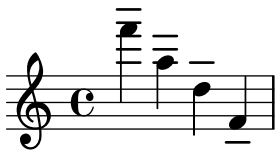
L'emplacement des lignes supplémentaires est réglable :

```
\new Staff \with {
  \override StaffSymbol.ledger-positions = #'(-3 -2 -1 2 5 6)
} \relative {
  f'''4 a, a,, f
}
```
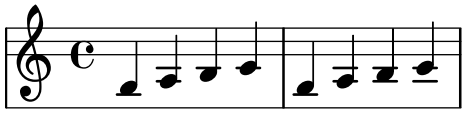


Vous pouvez faire apparaître des lignes supplémentaires additionnelles au-dessus ou en dessous des têtes de note selon leur positionnement relatif aux autres notes, qu'elles aient ou non elles-mêmes des lignes supplémentaires.

```
\new Staff \with {
  \override StaffSymbol.ledger-extra = 4
} \relative {
  f'''4 a, d, f,
}
```



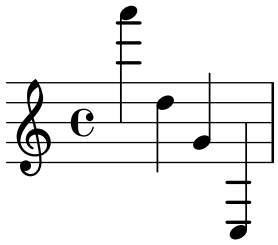
Des lignes supplémentaires peuvent apparaître y compris au sein d'une portée, notamment lorsque vous l'avez personnalisée. L'exemple suivant illustre deux cas de figure quant au positionnement des lignes supplémentaires selon que la propriété `legder-position` est définie explicitement ou non. La présence du `\stopStaff` est ici rendue nécessaire pour annuler les effets de la commande `\override` qui s'applique à l'intégralité du `StaffSymbol`.

```
\override Staff.StaffSymbol.line-positions =   #'(-8 0 2 4)
d4 e f g
\stopStaff
\startStaff
\override Staff.StaffSymbol.ledger-positions = #'(-8 -6 (-4 -2) 0)
d4 e f g
```
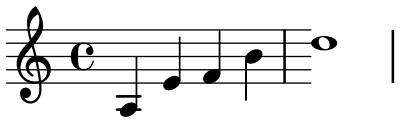


Modifier l'équidistance des lignes de la portée affectera aussi les lignes supplémentaires.

```
\new Staff \with {
  \override StaffSymbol.staff-space = 1.5
} \relative {
  f'''4 d, g, e,
}
```



La largeur d'une portée, exprimée en espace de portée, peut être figée. L'espacement des objets inclus dans cette portée ne sera en rien affecté par ce réglage.

```
\new Staff \with {
  \override StaffSymbol.width = 23
}
\relative { a4 e' f b | d1 }
```
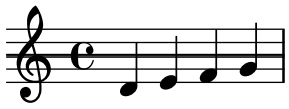


Morceaux choisis

Empâtement de certaines lignes d'une portée

Vous pourriez avoir envie, dans un but pédagogique, de rendre certaines lignes d'une portée plus épaisses que les autres, comme la ligne médiane, ou bien pour mettre en exergue la ligne portant la clef de sol. Il suffit pour cela d'ajouter une ligne qui sera accolée à celle qui doit être mise en évidence, grâce à la propriété `line-positions` de l'objet `StaffSymbol`.

```
{
  \override Staff.StaffSymbol.line-positions =
    #'(-4 -2 -0.2 0 0.2 2 4)
  d'4 e' f' g'
}
```



Voir aussi

Glossaire musicologique : Section "ligne" dans *Glossaire*, Section "ligne supplémentaire" dans *Glossaire*, Section "portée" dans *Glossaire*.

Manuel de notation : Section 1.3.1 [Clefs], page 20.
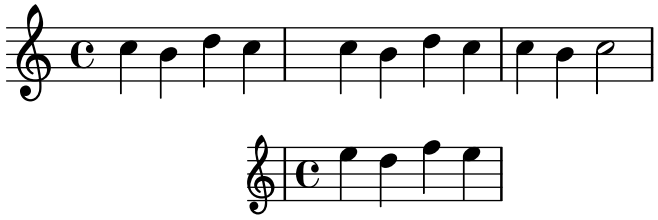
Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "StaffSymbol" dans *Référence des propriétés internes*, Section "staff-symbol-interface" dans *Référence des propriétés internes*.

6.2.2 Portées d'ossia

Une portée d'*ossia* – ou de variante – s'obtient en créant, à l'endroit approprié, une nouvelle portée simultanée :

```
\new Staff \relative {
  c''4 b d c
  <<
    { c4 b d c }
    \new Staff { e4 d f e }
  >>
  c4 b c2
}
```
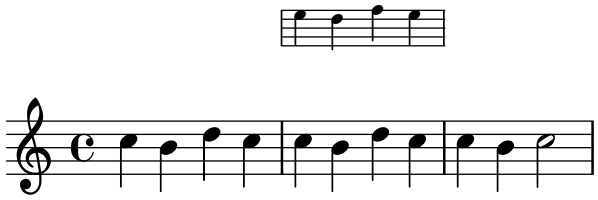


L'exemple ci-dessus n'est probablement pas ce qui vous conviendra le plus. Afin que cette ossia se place au-dessus de la portée à laquelle elle se réfère, étant par ailleurs dépourvue de métrique et de clef, et d'une taille légèrement inférieure, vous devrez avoir recours à quelques retouches. Le manuel d'initiation aborde une technique particulière pour obtenir ce résultat, au chapitre Section "Expressions musicales imbriquées" dans *Manuel d'initiation*.

L'exemple qui suit utilise, pour aligner la portée d'ossia, la propriété `alignAboveContext`. Cette méthode est tout à fait appropriée lorsqu'il y a un nombre restreint d'ossias.

```
\new Staff = "main" \relative {
  c''4 b d c
  <<
    { c4 b d c }

    \new Staff \with {
      \remove Time_signature_engraver
      alignAboveContext = "main"
      \magnifyStaff #2/3
      firstClef = ##f
    }
    { e4 d f e }
  >>
  c4 b c2
}
```

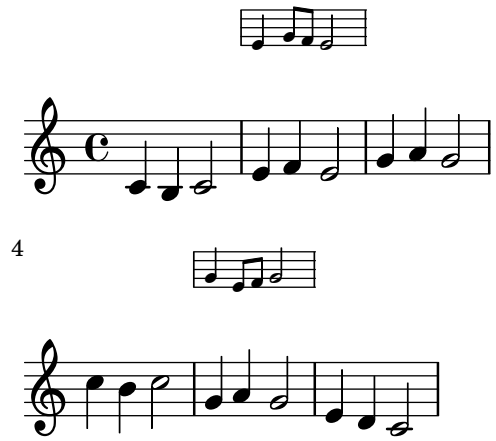


Dans le cas où de nombreux et courts fragments d'ossia affecteraient une même portée, il est judicieux de créer un contexte `Staff` vide auquel sera attribué un *identifiant*. Il suffira alors, pour entamer un fragment d'ossia, de « faire appel » à ce contexte grâce aux commandes `\startStaff`

et \stopStaff. Vous vous rendrez compte à l'utilisation des avantages que procure cette façon de procéder, bien plus que dans l'exemple suivant.
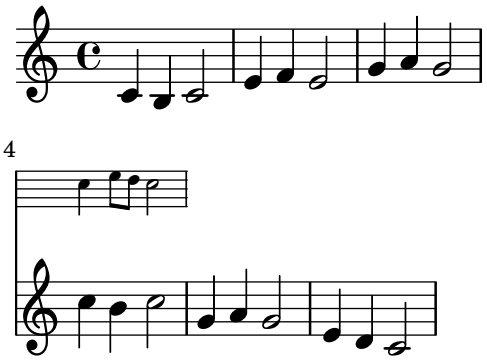
```
<<
  \new Staff = "ossia" \with {
    \remove Time_signature_engraver
    \hide Clef
    \magnifyStaff #2/3
  }
  { \stopStaff s1*6 }

  \new Staff \relative {
    c'4 b c2
    <<
      { e4 f e2 }
      \context Staff = "ossia" {
        \startStaff e4 g8 f e2 \stopStaff
      }
    >>
    g4 a g2 \break
    c4 b c2
    <<
      { g4 a g2 }
      \context Staff = "ossia" {
        \startStaff g4 e8 f g2 \stopStaff
      }
    >>
    e4 d c2
  }
>>
```



Vous pourriez aussi recourir à la commande \RemoveAllEmptyStaves pour créer votre portée d'ossia. Cependant, cette méthode reste limitée à l'apparition de ces ossias en début de ligne. Pour plus d'information au sujet de la commande \RemoveAllEmptyStaves, reportez-vous au chapitre Section 6.2.3 [Masquage de portées], page 265.

```
<<
  \new Staff = "ossia" \with {
    \remove Time_signature_engraver
    \hide Clef
    \magnifyStaff #2/3
    \RemoveAllEmptyStaves
  } \relative {
    R1*3
    c''4 e8 d c2
  }
  \new Staff \relative {
    c'4 b c2
    e4 f e2
    g4 a g2 \break
    c4 b c2
    g4 a g2
    e4 d c2
  }
>>
```



Morceaux choisis

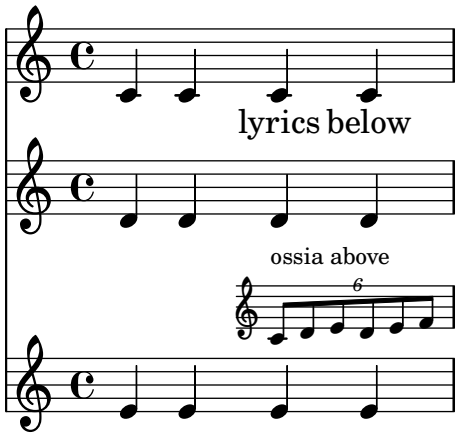
Positionnement d'une ossia et des paroles

Cet exemple illustre la manière de positionner une portée d'ossia et des paroles à l'aide des propriétés de contexte `alignBelowContext` et `alignAboveContext`.

```
\relative c' <<
  \new Staff = "1" { c4 c c c }
  \new Staff = "2" { d4 d d d }
  \new Staff = "3" { e4 e e e }

  { \skip 2
    <<
      \lyrics {
        \set alignBelowContext = "1"
        lyrics4 below
      }
      \new Staff \with {
        alignAboveContext = "3"
        fontSize = -2
        \override StaffSymbol.staff-space = #(magstep -2)
```

```
        \remove "Time_signature_engraver"
        \override VerticalAxisGroup.staff-staff-spacing =
           #'((minimum-distance . 0)
              (basic-distance . 0)
              (padding . 1))
      } {
        \tuplet 6/4 {
          \override TextScript.padding = 2
          c8[^"ossia above" d e d e f]
        }
      }
    >>
  }
>>
```



Voir aussi

Glossaire musicologique : Section “ossia” dans *Glossaire*, Section “portée” dans *Glossaire*, Section “Frenched staff” dans *Glossaire*.

Manuel d’initiation : Section “Expressions musicales imbriquées” dans *Manuel d’initiation*, Section “Longueur et épaisseur des objets” dans *Manuel d’initiation*, Section “Taille des objets” dans *Manuel d’initiation*.

Manuel de notation : Section 6.2.3 [Masquage de portées], page 265.

Morceaux choisis : Section “Notation sur la portée” dans *Morceaux choisis*.

Référence des propriétés internes : Section “StaffSymbol” dans *Référence des propriétés internes*.

6.2.3 Masquage de portées

Désactiver le graveur `Staff_symbol_engraver` dans un contexte `Staff` permet de masquer des lignes. La commande `\stopStaff` aura le même effet.

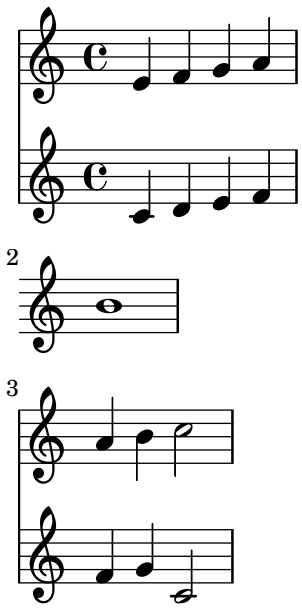
```
\new Staff \with {
  \remove Staff_symbol_engraver
}
\relative { a''8 f e16 d c b a2 }
```



L'instruction `\RemoveEmptyStaves` placée dans un bloc `\layout` ou dans une clause `\with` affectant une portée particulière, aura pour effet de masquer toute portée qui ne contient rien. Dans les partitions d'orchestre, les portées qui n'ont que des silences sont habituellement masquées afin de gagner de la place. Ce style d'édition s'appelle en anglais « Frenched Score ». Cette fonctionnalité masque et supprime toutes les portées vides d'une partition, hormis celles du premier système. Le premier système sera lui aussi concerné dès lors que sera utilisée l'instruction `\RemoveAllEmptyStaves`.

```
\layout {
  \context {
    \Staff
    \RemoveEmptyStaves
  }
}

\relative <<
  \new Staff {
    e'4 f g a \break
    b1 \break
    a4 b c2
  }
  \new Staff {
    c,4 d e f \break
    R1 \break
    f4 g c,2
  }
>>
```



Une portée est considérée comme vide dès lors qu'elle ne contient que des silences multimesures, des silences visibles ou invisibles (ou d'espacement – les *\skip*) ou bien une combinaison de ces éléments. **Tous** les autres objets musicaux, ceux qui font qu'une portée ne sera pas considérée vide, sont répertoriés dans la propriété de contexte `keepAliveInterfaces` telle que définie dans le fichier `ly/engraver-init.ly`.
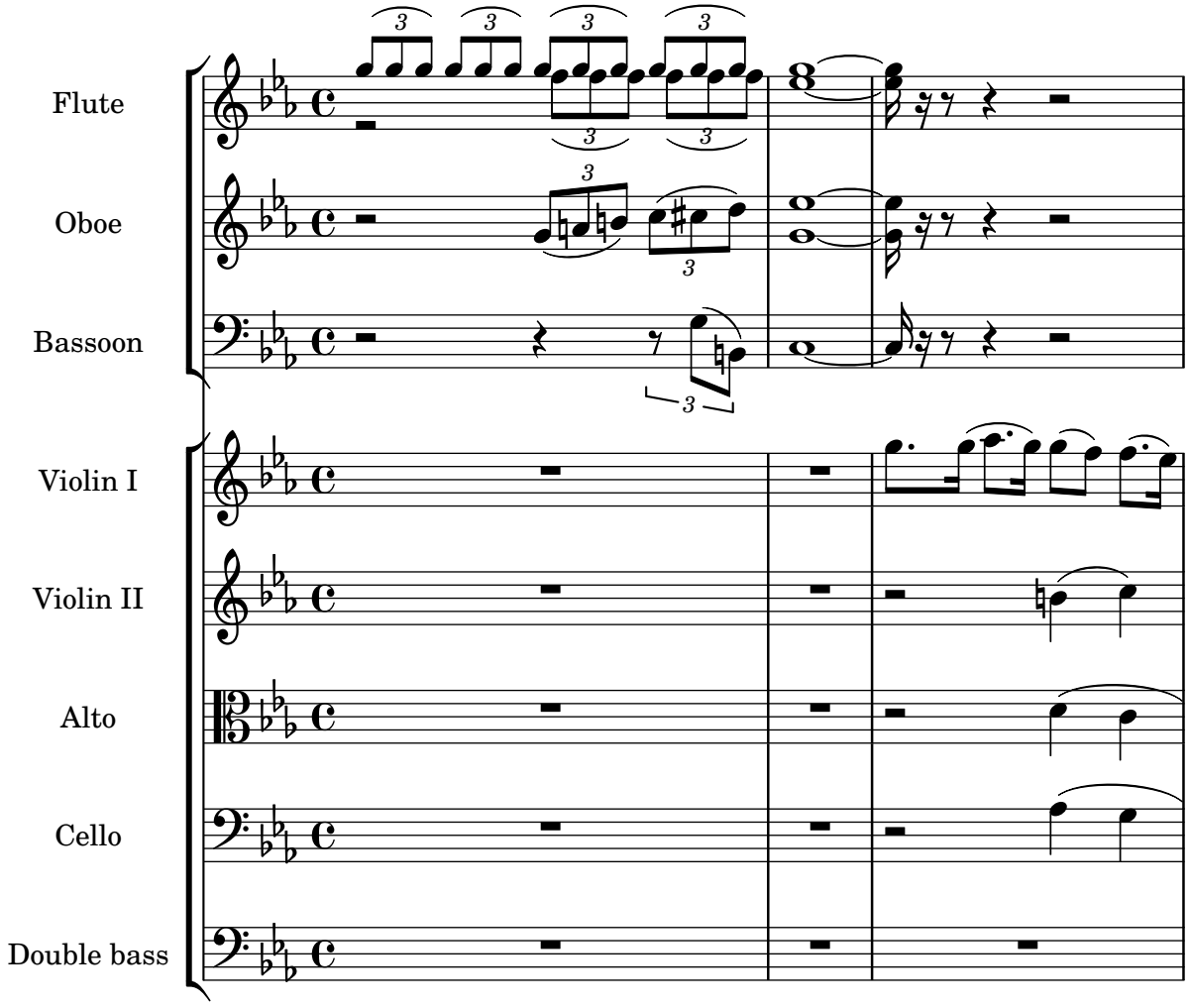
`\RemoveEmptyStaves` et `\RemoveAllEmptyStaves` sont tous deux des raccourcis prédéfinis qui déterminent les propriétés telles que `remove-empty` et `remove-first`, attachées à l'objet
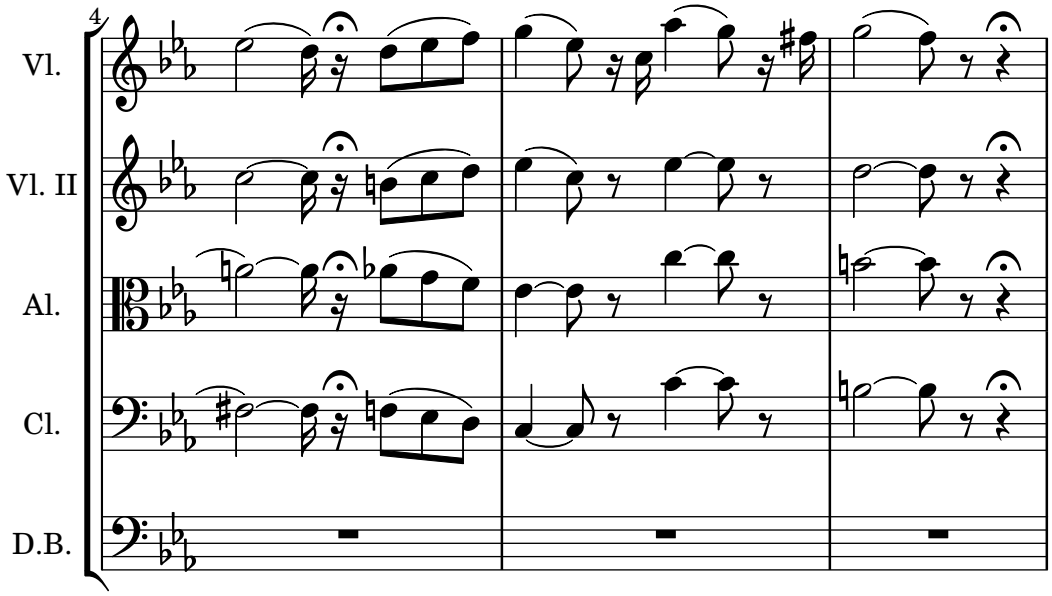
`VerticalAxisGroup`, comme indiqué dans Section B.22 [Identificateurs de modification de contexte], page 979.

Le `Keep_alive_together_engraver` permet de masquer l'intégralité d'un regroupement de portées plutôt que des portées individuelles. Il est normalement attaché au contexte `PianoStaff` : un système pianistique sera masqué uniquement dans le cas où les deux portées qui le composent sont vides. De la même manière, il est d'usage pour un conducteur d'orchestre de masquer les regroupements de portées plutôt que de manière individuelle. Ceci s'obtient en ajoutant le `Keep_alive_together_engraver` au regroupement de portées considéré, comme indiqué dans Section 33.4 [Modification des greffons de contexte], page 769, – voir Section 6.1.2 [Regroupement de portées], page 250, pour leur dénomination.

```
\layout {
  \context {
    \StaffGroup
    \RemoveEmptyStaves
    \consists Keep_alive_together_engraver
  }
}
```

Dans l'exemple suivant, les portées des instruments à vent sont absentes du deuxième système, alors que celle de la contrebasse demeure présente puisqu'elle est rattachée au regroupement des cordes qui, elles, continuent de jouer.





En interne, le `Keep_alive_together_engraver` recourt à la propriété `remove-layer` du `VerticalAxisGroup` d'une portée pour décider de son impression lorsqu'elle est considérée vide. Cette propriété peut aussi se définir manuellement, auquel cas elle agit sous forme d'index de priorité : des valeurs proches de zéro seront prioritaires sur celles plus élevées, ce qui aura pour conséquence de masquer d'abord les portées dont le `remove-layer` est plus grand.

Ceci se révèle tout particulièrement utile pour une « division » de portée, lorsque plusieurs parties indépendantes – voir Section 6.3 [Écriture de parties séparées], page 270, – doivent momentanément se répartir sur plusieurs portées. Dans l'exemple qui suit, deux parties sont réparties sur *trois* portées ; ces trois portées n'apparaissent toutefois jamais simultanément :

- au premier système, une seule d'entre elles est affichée, dans la mesure où la propriété `keepAliveInterfaces` a été réglée sur une liste vide – de fait, les deux autres portées sont considérées comme vides et donc masquées, en dépit de ce qu'elles pourraient contenir ;
- lorsque cette propriété est désactivée – et retrouve donc ses réglages par défaut – elle n'empêche plus les deux autres portées d'être affichées. Cependant, et parce que le réglage de leur `remove-layer` est inférieur à celui de la portée unique, ces deux portées seront alors imprimées à la place.

De telles substitutions ne s'appliquent pas seulement aux notes, accords ou autres éléments musicaux intervenant dès après le nouveau réglage, mais à l'intégralité du système où il est mentionné.

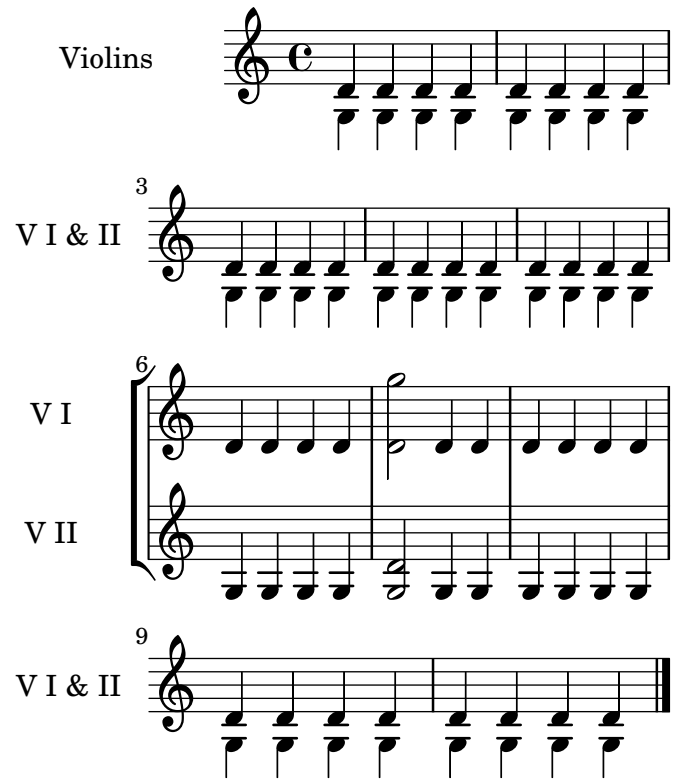
```
\layout {
  short-indent = 2\cm
  indent = 3\cm
  \context {
    \Staff
    keepAliveInterfaces = #'()
  }
}

violI = {
  \repeat unfold 24 { d'4 }
  \once \unset Staff.keepAliveInterfaces
  <d' g''>2
  \*14 { d'4 }
```

```
  \bar "|."
}

violII = {
  \repeat unfold 24 { g4 }
  <g d'>2
  \*14 { g4 }
  \bar "|."
}

\new StaffGroup \with { \consists Keep_alive_together_engraver } <<
  \new Staff \with {
    instrumentName = "Violins"
    shortInstrumentName = "V I & II"
    \override VerticalAxisGroup.remove-layer = 2
  } <<  \violI \\ \violII  >>
  \new Staff \with {
    instrumentName = "Violin I"
    shortInstrumentName = "V I"
    \RemoveAllEmptyStaves
    \override VerticalAxisGroup.remove-layer = 1
  } \violI
  \new Staff \with {
    instrumentName = "Violin II"
    shortInstrumentName = "V II"
    \RemoveAllEmptyStaves
    \override VerticalAxisGroup.remove-layer = 1
  } \violII
>>
```



`\RemoveAllEmptyStaves` permet aussi de gérer des fragments d'ossia attachés à une portée. Pour plus de détails, voir Section 6.2.2 [Portées d'ossia], page 262.

Commandes prédéfinies

`\RemoveEmptyStaves`, `\RemoveAllEmptyStaves`,

Voir aussi

Glossaire musicologique : Section "Frenched staff" dans *Glossaire*.

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 7.1.4 [Dictée à trous], page 297, Section B.22 [Identificateurs de modification de contexte], page 979, Section 33.4 [Modification des greffons de contexte], page 769, Section 33.5 [Modification des réglages par défaut d'un contexte], page 771, Section 6.2.2 [Portées d'ossia], page 262, Section 6.1.2 [Regroupement de portées], page 250, Section 2.2.2 [Silences invisibles], page 71, Section 6.2.1 [Symbole de la portée], page 258, Section 36.7 [Visibilité des objets], page 812.

Fichiers d'initialisation : `ly/engraver-init.ly`.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Axis_group_engraver" dans *Référence des propriétés internes*, Section "ChordNames" dans *Référence des propriétés internes*, Section "FiguredBass" dans *Référence des propriétés internes*, Section "Keep_alive_together_engraver" dans *Référence des propriétés internes*, Section "Lyrics" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*, Section "Staff_symbol_engraver" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Supprimer le `Staff_symbol_engraver` aura pour effet de masquer les barres de mesure. Forcer leur visibilité peut entraîner des problèmes de formatage. En pareil cas il vaut mieux, au lieu de supprimer le graveur, recourir aux dérogations suivantes :

```
\omit StaffSymbol
\override NoteHead.no-ledgers = ##t
```

En ce qui concerne les désagréments et messages liés à l'utilisation de l'instruction `\RemoveEmptyStaves`, consultez Section 33.5 [Modification des réglages par défaut d'un contexte], page 771.

6.3 Écriture de parties séparées
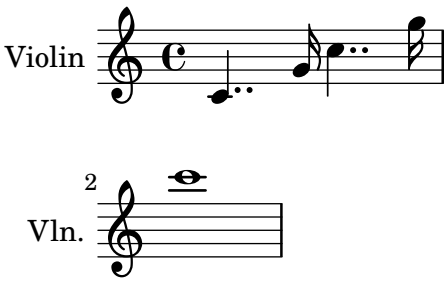
Nous allons voir, au fil des lignes qui suivent, comment préparer un « matériel » pour orchestre ou ensemble musical, ce qui requiert souvent d'insérer des noms d'instrument dans une partition. Nous aborderons aussi la citation d'autres voix, et comment la mettre en forme, ainsi que le moyen de contracter une succession de mesures vides dans une partition de pupitre.

Par ailleurs, une méthode pour gérer la division d'un pupitre est abordée à la rubrique Section 6.2.3 [Masquage de portées], page 265.

6.3.1 Noms d'instrument
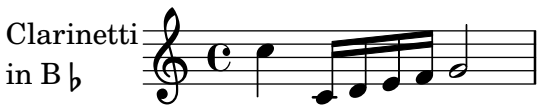
Dans un conducteur, les noms d'instrument sont portés en regard de chacune des portées, qu'il s'agisse d'un contexte `Staff`, `PianoStaff`, `StaffGroup`, `GrandStaff` ou `ChoirStaff` . La première ligne affichera la valeur de `instrumentName`, et les suivantes celle de `shortInstrumentName`.

```
\new Staff \with {
  instrumentName = "Violin "
  shortInstrumentName = "Vln. "
} \relative {
  c'4.. g'16 c4.. g'16 \break | c1 |
}
```
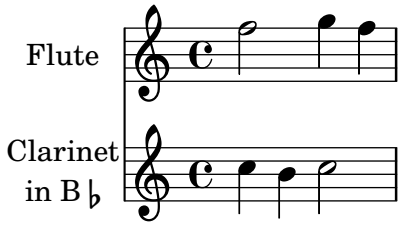


Le recours à la commande \markup permet de construire des noms d'instrument particuliers, tels que

```
\new Staff \with {
  instrumentName = \markup {
    \column { "Clarinetti"
      \line { "in B" \smaller \flat }
    }
  }
} \relative {
  c''4 c,16 d e f g2
}
```



Lorsque plusieurs contextes de portée sont regroupés, les noms d'instrument, que ce soit sous leur forme développée ou abrégée, sont par défaut centrés. Si l'un d'entre eux est libellé sur plusieurs lignes, il faudra recourir à l'instruction \center-column :

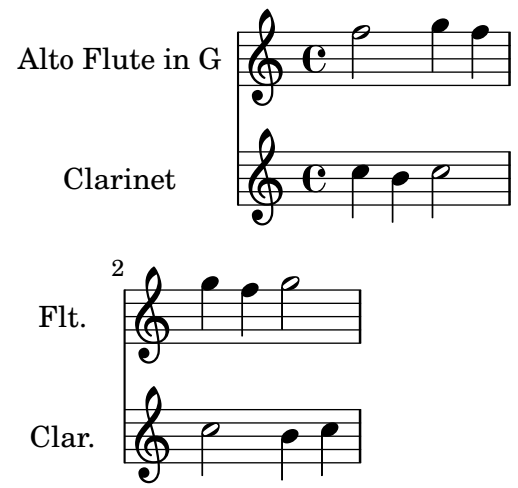
```
<<
  \new Staff \with {
    instrumentName = "Flute"
  }
  { f2 g4 f }
  \new Staff \with {
    instrumentName = \markup {
      \center-column { "Clarinet"
      \line { "in B" \smaller \flat }
      }
    }
  }
  { c4 b c2 }
>>
```



Lorsque le nom d'un instrument est relativement long, il est judicieux d'augmenter les retraits – *indent* – au sein du bloc `\layout` à l'aide des commandes `indent` et `short-indent`. Pour plus de plus amples détails sur ces réglages, reportez-vous au chapitre Section 26.5.3 [Variables d'indentation et de décalage], page 699.

```
<<
  \new Staff \with {
    instrumentName = "Alto Flute in G"
    shortInstrumentName = "Flt."
  } \relative {
    f''2 g4 f \break
    g4 f g2
  }
  \new Staff \with {
    instrumentName = "Clarinet"
    shortInstrumentName = "Clar."
  } \relative {
    c''4 b c2 \break
    c2 b4 c
  }
>>

\layout {
  indent = 3.0\cm
  short-indent = 1.5\cm
}
```



Des noms d'instrument peuvent s'utiliser dans d'autres contextes, tels que `ChordNames` ou `FiguredBass`, dès lors qu'on leur adjoint le graveur `Instrument_name_engraver`. Pour de plus amples informations sur la manière d'activer ou désactiver un graveur, voir Section 33.4 [Modification des greffons de contexte], page 769.

Le nom abrégé d'un instrument (shortInstrumentName) peut changer en cours de morceau, en même temps que les autres réglages propres au nouvel instrument. Notez cependant que la valeur de instrumentName ne s'affichera que sur la première portée :

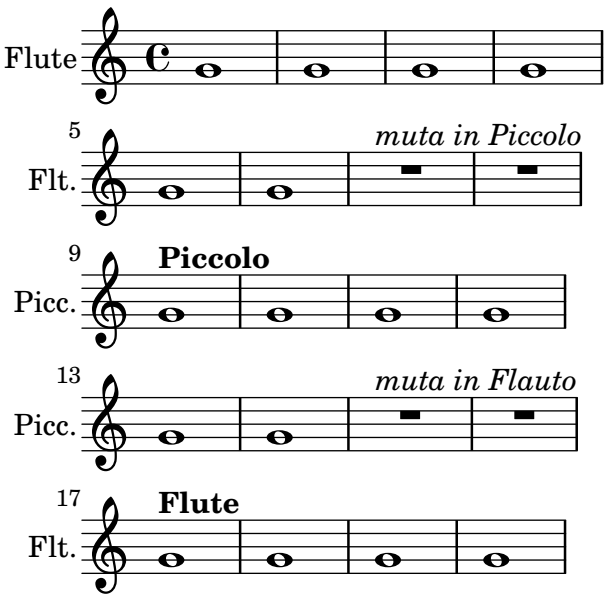
```
prepPiccolo = <>^\markup \italic { muta in Piccolo }

prepFlute = <>^\markup \italic { muta in Flauto }

setPiccolo = {
  <>^\markup \bold { Piccolo }
  \transposition c''
}

setFlute = {
  <>^\markup \bold { Flute }
  \transposition c'
}

\new Staff \with {
  instrumentName = "Flute"
  shortInstrumentName = "Flt."
}
\relative {
  g'1 g g g \break
  g1 g \prepPiccolo R R \break
  \set Staff.instrumentName = "Piccolo"
  \set Staff.shortInstrumentName = "Picc."
  \setPiccolo
  g1 g g g \break
  g1 g \prepFlute R R \break
  \set Staff.instrumentName = "Flute"
  \set Staff.shortInstrumentName = "Flt."
  \setFlute
  g1 g g g
}
```



Voir aussi

Manuel de notation : Section 33.4 [Modification des greffons de contexte], page 769, Section 26.5.3 [Variables d'indentation et de décalage], page 699.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "InstrumentName" dans *Référence des propriétés internes*, Section "PianoStaff" dans *Référence des propriétés internes*, Section "Staff" dans *Référence des propriétés internes*.

6.3.2 Citation d'autres voix

Il est assez courant qu'une voix soit doublée par une autre. Par exemple, les premiers et seconds violons peuvent jouer les mêmes notes durant un moment. LilyPond gère parfaitement ces situations où une voix est la *réplique* d'une autre, sans devoir ressaisir la musique en question.

L'instruction `\addQuote`, placée au niveau le plus haut – c'est à dire en dehors de tout bloc de musique – définit la musique dont il sera possible de répliquer des fragments.
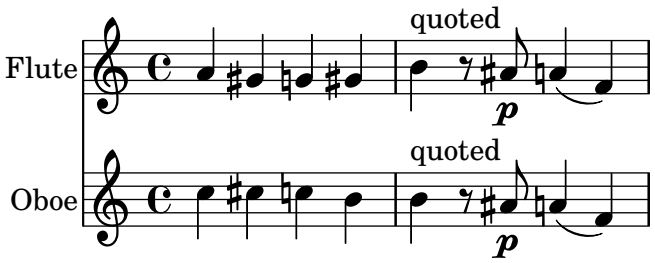
Au cours d'une partie, des extraits de répliques peuvent être cités en utilisant la commande `\quoteDuring`. Cette commande prend deux arguments : le nom de la voix reproduite, tel que défini par `\addQuote`, et une expression musicale qui indique la durée de cette citation.

```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { s1 }
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```



Si l'expression musicale utilisée pour `\quoteDuring` contenait autre chose que du silence, il en résulterait une situation polyphonique, ce qui n'est pas le but recherché :
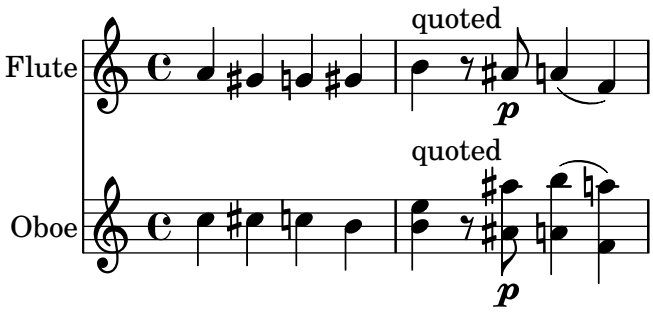
```
fluteNotes = \relative {
  a'4 gis g gis | b4^"quoted" r8 ais\p a4( f)
 }

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "flute" { e4 r8 ais b4 a }
```

```
}

\addQuote "flute" { \fluteNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```



Lorsqu'une commande \unfoldRepeats est requise dans une expression musicale de telle sorte que la musique soit réimprimée par \quoteDuring, l'instruction \addQuote doit elle-même contenir la commande \unfoldRepeats :

```
fluteNotes = \relative {
  \repeat volta 2 { a'4 gis g gis }
}

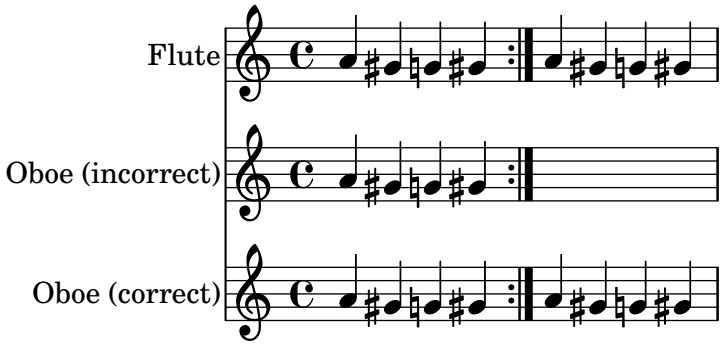
oboeNotesDW = \relative {
  \repeat volta 2 \quoteDuring "incorrect" { s1 }
}

oboeNotesW = \relative {
  \repeat volta 2 \quoteDuring "correct" { s1 }
}


\addQuote "incorrect" { \fluteNotes }

\addQuote "correct" { \unfoldRepeats \fluteNotes }

\score {
  \unfoldRepeats
  <<
    \new Staff \with { instrumentName = "Flute" }
    \fluteNotes
    \new Staff \with { instrumentName = "Oboe (incorrect)" }
    \oboeNotesDW
    \new Staff \with { instrumentName = "Oboe (correct)" }
    \oboeNotesW
  >>
}
```
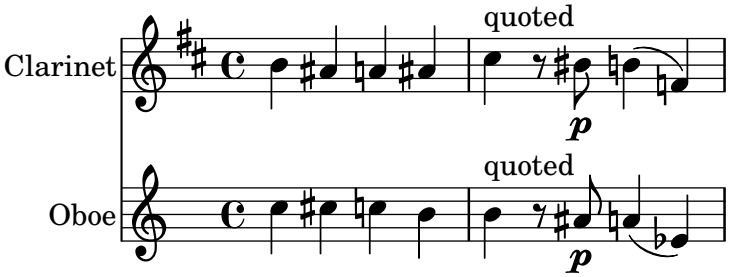


L'instruction `\quoteDuring` prendra en compte les réglages d'une commande `\transposition`, qu'elle apparaisse au niveau de la voix répliquée ou dans celle qui réplique.

```
clarinetNotes = \relative c'' {
  \transposition bes
  \key d \major
  b4 ais a ais | cis4^"quoted" r8 bis\p b4( f)
}

oboeNotes = \relative {
  c''4 cis c b \quoteDuring "clarinet" { s1 }
 }
\addQuote "clarinet" { \clarinetNotes }

\score {
  <<
    \new Staff \with { instrumentName = "Clarinet" } \clarinetNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
}
```
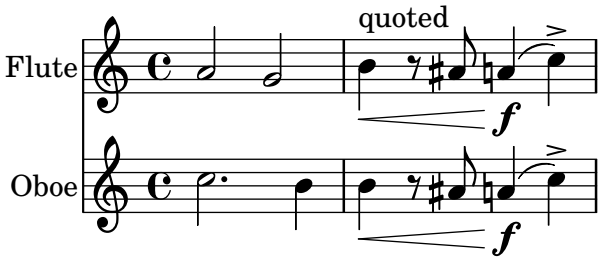


LilyPond répliquera, par défaut, tous les éléments – articulations, nuances, *markups*, etc. La propriété `quotedEventTypes` permet de définir plus précisément quels éléments de la voix originelle seront reproduits.

```
fluteNotes = \relative {
  a'2 g2 |
  b4\<^"quoted" r8 ais a4\f( c->)
 }

oboeNotes = \relative {
  c''2. b4 |
  \quoteDuring "flute" { s1 }
}
```

```
\addQuote "flute" { \fluteNotes }

\score {
  <<
    \set Score.quotedEventTypes = #'(note-event articulation-event
                                     crescendo-event rest-event
                                     slur-event dynamic-event)
    \new Staff \with { instrumentName = "Flute" } \fluteNotes
    \new Staff \with { instrumentName = "Oboe" } \oboeNotes
  >>
 }
```



Les citations peuvent être « balisées » par un nom particulier afin de les utiliser de différentes manières. Pour de plus amples détails à ce propos, consultez le chapitre Section 22.2.2 [Utilisation de balises], page 647.

Voir aussi

Manuel de notation : Section 1.3.4 [Instruments transpositeurs], page 30, Section 22.2.2 [Utilisation de balises], page 647.

Fichiers d'initialisation : `scm/define-event-classes.scm`.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Music classes" dans *Référence des propriétés internes*, Section "QuoteMusic" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Seul le contenu de la première `Voice` rencontrée dans la partie marquée d'une commande `\addQuote` pourra être retenu. Par voie de conséquence, *musique* ne saurait comprendre de `\new` ou une instance `\context Voice` qui la ferait passer à une autre voix.

Citer des notes d'ornement ne fonctionne pas, et peut même entraîner un blocage de LilyPond.
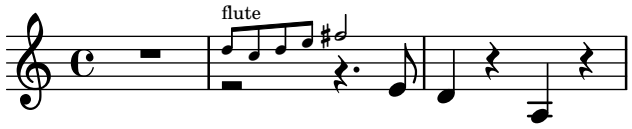
Citer des triolets imbriqués peut entraîner un résultat de piètre qualité.

6.3.3 Mise en forme d'une citation

Le moyen le plus simple pour mettre en forme des notes provenant d'une autre voix consiste à déclarer explicitement un contexte `CueVoice` au sein de la voix où elle apparaît.
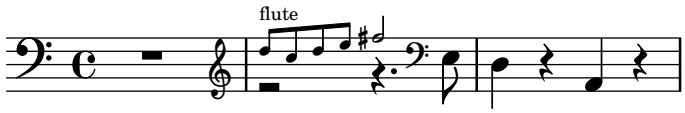
```
\relative {
  R1
  <<
    { e'2\rest r4. e8 }
    \new CueVoice {
      \stemUp d'8^"flute" c d e fis2
    }
```

```
  >>
  d,4 r a r
}
```
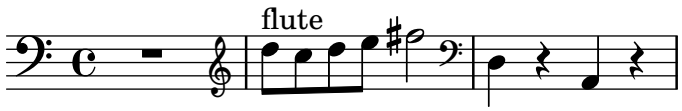


L'instruction \cueClef, utilisée conjointement à un contexte CueVoice explicite permet d'indiquer la clef, dans une taille réduite, propre à la voix citée. Le retour à la clef d'origine s'effectue à l'aide de l'instruction \cueClefUnset.

```
\relative {
  \clef "bass"
  R1
  <<
    { e'2\rest r4. \cueClefUnset e,8 }
    \new CueVoice {
      \cueClef "treble" \stemUp d''8^"flute" c d e fis2
    }
  >>
  d,,4 r a r
}
```



Notez que les deux instructions \cueClef et \cueClefUnset sont disponibles si nécessaire en dehors d'un CueVoice.

```
\relative {
  \clef "bass"
  R1
  \cueClef "treble"
  d''8^"flute" c d e fis2
  \cueClefUnset
  d,,4 r a r
}
```



Lorsque la situation est plus complexe, instrument transpositeur ou citations de plusieurs sources, vous disposez des instructions \cueDuring et \cueDuringWithClef, versions spécifiques de la commande \quoteDuring – voir la rubrique précédente (Section 6.3.2 [Citation d'autres voix], page 274).

Leur syntaxe est :

\cueDuring *origine position musique*

et

\cueDuringWithClef *origine position #clef musique*

Des mesures issues de la partie d'*origine* seront recopiées dans un contexte de CueVoice et synchronisées avec *musique* – habituellement un silence. L'apparition des petites notes initialise

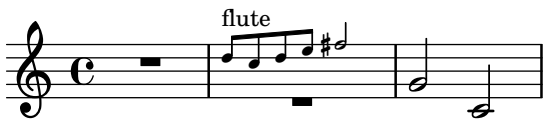
une polyphonie temporaire pour la portée concernée. L'argument *position* détermine si ces petites notes seront attachées à la première ou à la seconde voix – `UP` pour la première, `DOWN` pour la seconde.

```
fluteNotes = \relative {
  r2. c''4 | d8 c d e fis2 | g2 d |
}

oboeNotes = \relative c'' {
  R1
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  g2 c,
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \oboeNotes
}
```
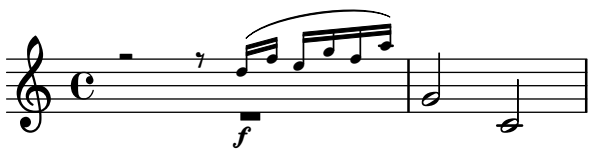


La propriété `quotedCueEventTypes` permet de définir précisément quels éléments de la voix originelle seront reproduits. Sa valeur par défaut est `'(note-event rest-event tie-event beam-event tuplet-span-event)`. LilyPond reproduira donc les notes, silences, liaisons de prolongation, ligatures et n-olets, mais pas les articulations, annotations ni nuances.

> **Note :** Dans l'exemple précédent, il était nécessaire de déclarer explicitement le contexte `Voice`, sinon l'intégralité de l'expression musicale se serait retrouvée dans le contexte `CueVoice`.

```
oboeNotes = \relative {
  r2 r8 d''16(\f f e g f a)
  g8 g16 g g2.
}
\addQuote "oboe" { \oboeNotes }

\new Voice \relative c'' {
  \set Score.quotedCueEventTypes = #'(note-event rest-event tie-event
                                      beam-event tuplet-span-event
                                      dynamic-event slur-event)
  \cueDuring "oboe" #UP { R1 }
  g2 c,
}
```
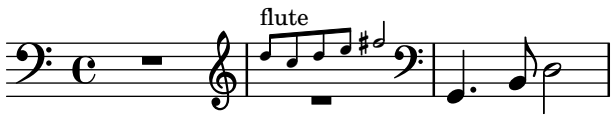


Le nom de l'instrument qui est répliqué peut s'indiquer à l'aide d'un *markup*. Par ailleurs, si la citation nécessite l'apparition d'une clef différente, celle-ci devra être introduite manuellement, tout comme l'originale qui devra être rappelée en fin de citation.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  \clef treble
  <>^\markup \tiny { flute }
  \cueDuring "flute" #UP { R1 }
  \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```
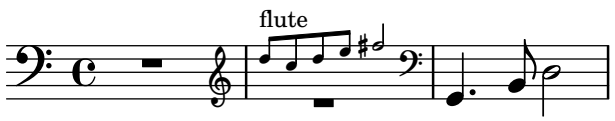


L'instruction `\cueDuringWithClef` se chargera quant à elle, et grâce à un argument supplémentaire, de gérer le changement de clef nécessaire à la citation et le retour à la clef originelle.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}

bassoonNotes = \relative c {
  \clef bass
  R1
  <>^\markup { \tiny "flute" }
  \cueDuringWithClef "flute" #UP "treble" { R1 }
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}
```



L'instruction \cueDuring, à l'instar de la commande \quoteDuring, tient compte des instruments transpositeurs. La citation s'effectue aux hauteurs correspondant à l'instrument où elles apparaissent.

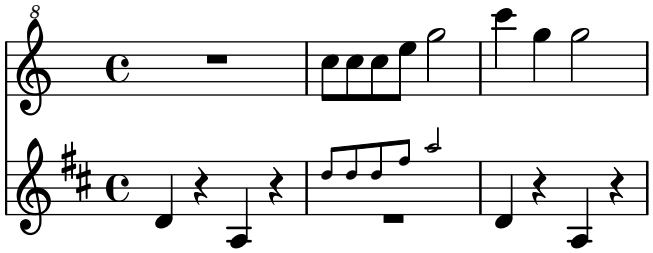
L'instruction \transposedCueDuring est particulièrement adaptée pour des instrument ayant une tessiture éloignée, comme dans le cas d'un piccolo cité dans une partie de clarinette basse. Sa syntaxe est identique à celle de \cueDuring, à ceci près qu'elle nécessite un argument supplémentaire afin de spécifier la transposition à effectuer en hauteur absolue.

```
piccoloNotes = \relative {
  \clef "treble^8"
  R1
  c'''8 c c e g2
  c4 g g2
}

bassClarinetNotes = \relative c' {
  \key d \major
  \transposition bes,
  d4 r a r
  \transposedCueDuring "piccolo" #UP d { R1 }
  d4 r a r
}

\addQuote "piccolo" { \piccoloNotes }

<<
  \new Staff \piccoloNotes
  \new Staff \bassClarinetNotes
>>
```



La commande killCues permet de supprimer les notes d'une citation. Ceci est utile lorsque cette citation n'est pas imprimée dans le conducteur entre autres. killCues supprimera les notes et autres événements pris en charge par \cueDuring. Pour les autres annotations telles que changement de clef ou instrument concerné, faites appel à des balises – voir Section 22.2.2 [Utilisation de balises], page 647, à ce sujet.

```
fluteNotes = \relative {
  r2. c''4 d8 c d e fis2 g2 d2
}
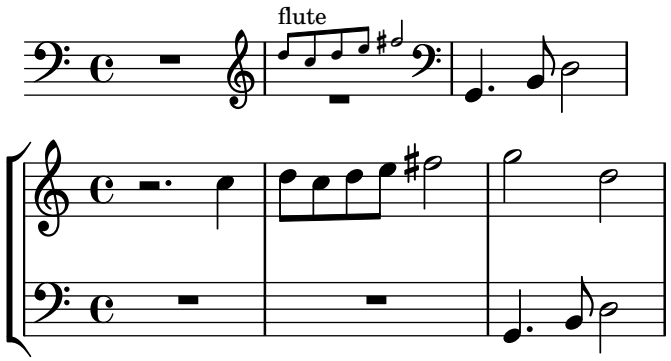
bassoonNotes = \relative c {
  \clef bass
  R1
  \tag #'part {
    \clef treble
    <>^\markup \tiny { flute }
```

```
  }
  \cueDuring "flute" #UP { R1 }
  \tag #'part \clef bass
  g4. b8 d2
}

\addQuote "flute" { \fluteNotes }

\new Staff {
  \bassoonNotes
}

\new StaffGroup <<
  \new Staff {
    \fluteNotes
  }
  \new Staff {
    \removeWithTag #'part { \killCues { \bassoonNotes } }
  }
>>
```



Voir aussi

Manuel de notation : Section 6.3.2 [Citation d'autres voix], page 274, Section 9.6.3 [Citation-repère], page 407, Section 1.3.1 [Clefs], page 20, Section 1.3.4 [Instruments transpositeurs], page 30, Section 6.3.1 [Noms d'instrument], page 270, Section 22.2.2 [Utilisation de balises], page 647.

Morceaux choisis : Section "Notation sur la portée" dans *Morceaux choisis*.

Référence des propriétés internes : Section "CueVoice" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La commande `\cueDuring` ne sait pas gérer les collisions de silence entre les contextes `Voice` et `CueVoice`.

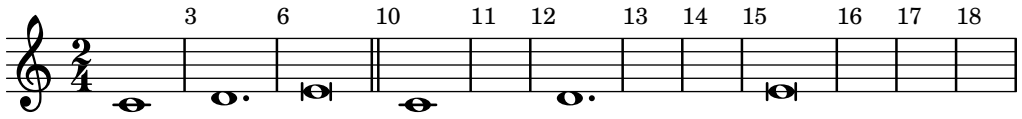
Dans le cadre d'un `\cueDuringWithClef` ou d'un `\transposedCueDuring`, l'argument supplémentaire doit intervenir après l'origine et la position.

6.3.4 Compression de mesures vides

Toutes les mesures sont par défaut imprimées, même si elles sont vides ; ceci peut se produire dans le cas où un événement rythmique (note, silence ou saut) dure au point de s'étaler sur plusieurs mesures. Ce comportement peut se modifier de sorte que les mesures vides sont compressées en

une seule mesure comme ici (la deuxième partie de cet exemple, où les mesures sont expansées, retrouve en fait le comportement par défaut) :

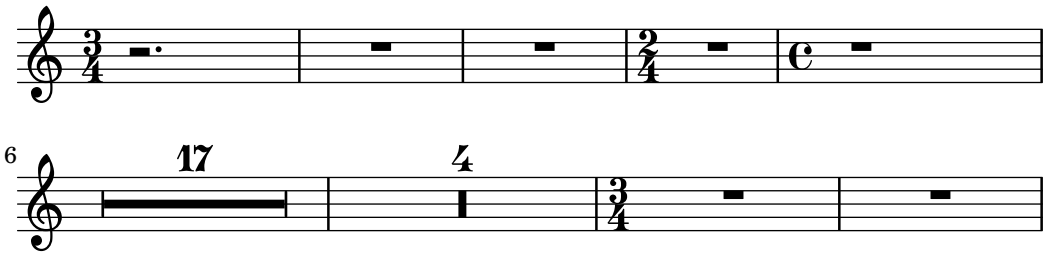
```
\override Score.BarNumber.break-visibility = ##(#f #t #t)
\time 2/4
\compressEmptyMeasures
c'1 d'1. e'\breve
\bar "||"
\expandEmptyMeasures
c'1 d'1. e'\breve
```



Bien que la syntaxe de cette notation soit correcte, elle peut être déroutante d'un point de vue musical, ce qu'illustre l'exemple ci-dessus. C'est pourquoi l'impression des numéros de mesure devient nécessaire, en suivant la syntaxe indiquée dans Section 36.7.4 [Utilisation de break-visibility], page 815.

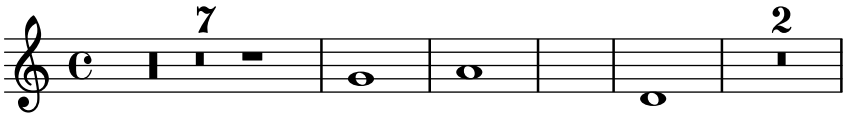
Une telle notation peut toutefois s'avérer utile lorsqu'elle s'applique à des Section 2.2.3 [Silences valant une mesure], page 73. Un silence de plusieurs mesures sera alors affiché sous la forme d'une seule mesure contenant un symbole de silence multimesure surplombé du nombre de mesures de silence :

```
% Comportement par défaut
\time 3/4 r2. | R2.*2 |
\time 2/4 R2 |
\time 4/4
% Mesures de silence comprimées en une seule mesure
\compressEmptyMeasures
r1 | R1*17 | R1*4 |
\expandEmptyMeasures
% Mesures de silence à nouveau expansées
\time 3/4
R2.*2 |
```



Contrairement à \compressEmptyMeasures, la fonction musicale \compressMMRests s'applique exclusivement aux silences tout en laissant les autres événements expansés. En sa qualité de fonction, et non de réglage de propriété, sa syntaxe diffère quelque peu, en ceci qu'elle doit être suivie d'une expression musicale :

```
\compressMMRests {
  % Les silences sont comprimés...
  R1*7
  % ... mais les notes peuvent s'étaler sur plusieurs mesures.
  g'1 a'1*2 d'1
  R1*2
}
```



Toutes les commandes décrites dans ces paragraphes reposent en fait sur la propriété interne `skipBars`, qui se définit au sein du contexte `Score` comme indiqué dans Section 35.2 [Les commandes \set et \unset], page 786.

Commandes prédéfinies

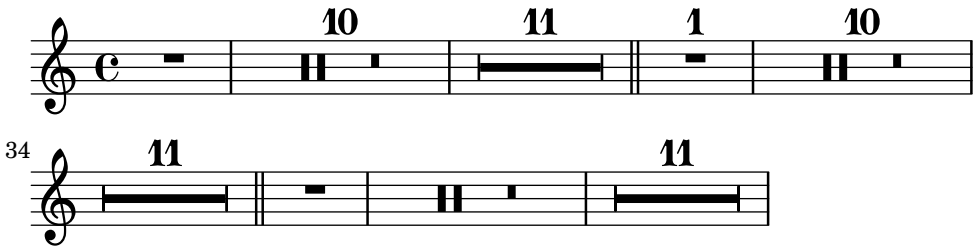
`\compressEmptyMeasures`, `\expandEmptyMeasures`, `\compressMMRests`.

Morceaux choisis

Dénombrer une unique mesure de silence

Les silences multimesures affichent leur longueur sauf s'il n'y a qu'une seule mesure. Ceci peut se modifier en réglant `restNumberThreshold`.

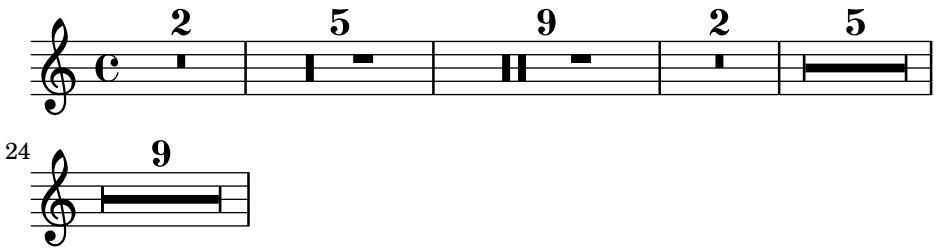
```
{
  \compressEmptyMeasures
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 0
  R1 R1*10 R1*11 \bar "||"
  \set restNumberThreshold = 10
  R1 R1*10 R1*11
}
```



Modifier l'apparence d'un silence multimesure

Dans le cas où ce silence dure moins de dix mesures, LilyPond imprime sur la portée des « silences d'église » – *Kirchenpause* en allemand – et qui sont une simple suite de rectangles. Par défaut fixée à 10, la propriété `expand-limit` peut se régler afin d'obtenir un silence unique.

```
\relative c'' {
  \compressMMRests {
    R1*2 | R1*5 | R1*9
    \override MultiMeasureRest.expand-limit = 3
    R1*2 | R1*5 | R1*9
  }
}
```

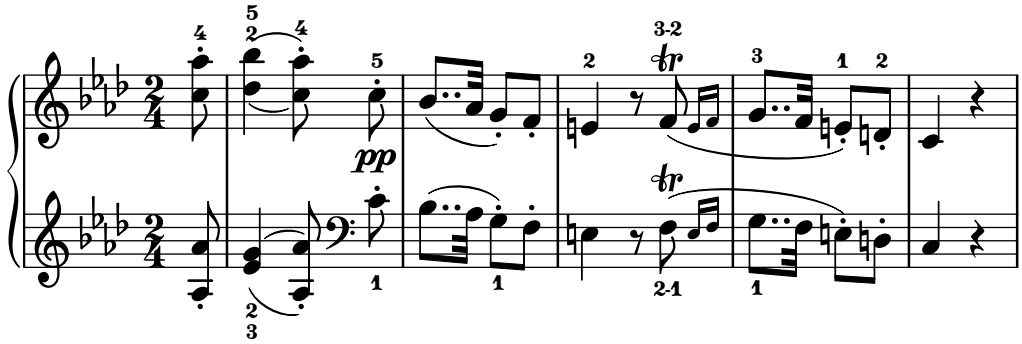


Voir aussi

Manuel de notation : Section 35.2 [Les commandes `\set` et `\unset`], page 786, Section 2.2.3 [Silences valant une mesure], page 73, Section 36.7.4 [Utilisation de `break-visibility`], page 815.

Référence des propriétés internes : Section “MultiMeasureRest” dans *Référence des propriétés internes*, Section “MultiMeasureRestNumber” dans *Référence des propriétés internes*, Section “MultiMeasureRestScript” dans *Référence des propriétés internes*, Section “MultiMeasureRestText” dans *Référence des propriétés internes*.

7 Annotations éditoriales



Ce chapitre traite de la manière de modifier l'apparence des notes dans un but pédagogique ou d'analyse.

7.1 Dans la portée
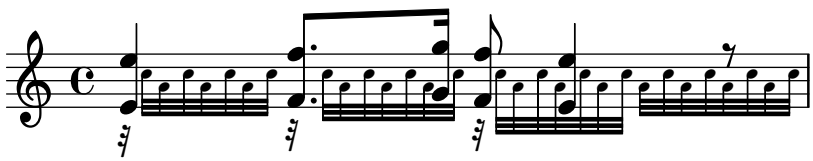
Nous allons voir ici comment mettre en exergue certains éléments à l'intérieur même de la portée.

7.1.1 Indication de la taille de fonte musicale

> **Note :**
> Pour les tailles de police textuelle, voir Section 8.2.2 [Sélection de la fonte et de la taille], page 333.
> Pour la taille des portées, voir Section 27.2 [Définition de la taille de portée], page 707.
> Pour les petites notes, voir Section 6.3.3 [Mise en forme d'une citation], page 277.
> Pour les variantes, voir Section 6.2.2 [Portées d'ossia], page 262.

Le plus sûr moyen de régler la taille des éléments de notation sans modifier la taille de la portée consiste à utiliser la commande `\magnifyMusic` :

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```

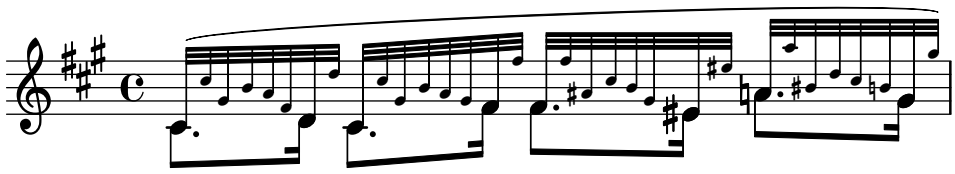


La présence d'un \override dans cet exemple permet de contourner une bogue – voir « Problèmes connus et avertissements » en fin de section.

Dans le cas de fusion d'une tête de note normale avec une autre de taille inférieure, la taille de la petite note peut nécessiter une réinitialisation – à l'aide d'un \once \normalsize – de telle sorte que les hampes et altérations s'alignent correctement :

```
\new Staff <<
  \key fis \minor
  \mergeDifferentlyDottedOn
  \new Voice \relative {
    \voiceOne
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment =
        #(* 1.2 0.63)

      \once \normalsize cis'32( cis' gis b a fis
        \once \normalsize d d'
      \once \normalsize cis, cis' gis b a gis
        \once \normalsize fis fis'
      \once \normalsize fis, fis' ais, cis b gis
        \once \normalsize eis eis'
      \once \normalsize a, a' bis, d cis b
        \once \normalsize gis gis')
    }
  }
  \new Voice \relative {
    \voiceTwo
    cis'8. d16 cis8. fis16 fis8. eis16 a8. gis16
  }
>>
```
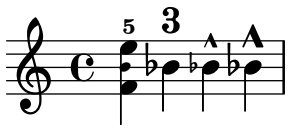


La commande \magnifyMusic n'est pas conçue pour gérer les citations, notes d'ornement ou portées d'ossia – des moyens spécifiques sont déjà disponibles en la matière. Elle est par contre tout à fait adaptée dans le cas d'un instrument particulier disposant de sa propre portée là où des notes d'ornement seraient inappropriées, comme pour une pseudo-cadence ou les exemples ci-dessus. L'attribution d'une valeur de 0,63 à \magnifyMusic duplique les dimensions d'un contexte CueVoice.

> **Note :** La commande \magnifyMusic n'est pas censée intervenir en complément d'une modification de la taille de portée – voir Section 27.2 [Définition de la taille de portée], page 707.

Redimmensionnement individuel d'objets de rendu

L'ajustement de la propriété font-size à l'aide des commandes \tweak ou \override permet de retailler un objet de rendu particulier :

```
\relative {
  % resize a note head
  <f' \tweak font-size -4 b e>-5
  % resize a fingering
  bes-\tweak font-size 0 -3
  % resize an accidental
  \once \override Accidental.font-size = -4 bes!-^
  % resize an articulation
  \once \override Script.font-size = 4 bes!-^
}
```
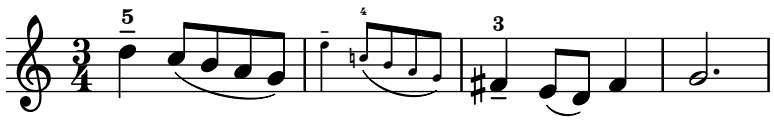


La valeur par défaut de `font-size` est mentionnée, pour chaque objet de rendu, dans la référence des propriétés internes. La propriété `font-size` ne peut intervenir que pour les objets qui utilisent des polices, autrement dit ceux qui disposent de l'interface de rendu `font-interface`. L'absence de `font-size` dans la liste des « réglages par défaut » d'un objet signifie qu'elle est de 0 – voir Section "All layout objects" dans *Référence des propriétés internes (Tous les objets de rendu)*.

La propriété `fontSize`

La propriété `fontSize` d'un contexte a pour effet de définir la taille proportionnelle de tout élément de notation basé sur un glyphe dans ce contexte.[1]

```
\relative {
  \time 3/4
  d''4---5 c8( b a g) |
  \set fontSize = -6
  e'4-- c!8-4( b a g) |
  \set fontSize = 0
  fis4---3 e8( d) fis4 |
  g2.
}
```



La valeur de `fontSize` est un nombre indiquant la taille relative par rapport à la hauteur standard de la portée en cours. La valeur par défaut de `fontSize` est de 0. Un pas de six aboutit exactement au doublement de la taille ; un pas de moins six la réduit de moitié. Chaque incrément correspond à une augmentation d'environ 12 % de la taille de la police.

La fonction Scheme `magnification->font-size` permet de s'affranchir de l'échelle logarithmique peu intuitive des unités de la propriété `font-size`. Ainsi, l'ajustement à 75 % de la notation musicale par rapport à la taille de la portée peut se libeller :

```
\set fontSize = #(magnification->font-size 0.75)
```

[1] Notez bien les mots « basé sur un glyphe » – une hampe, par exemple, n'est pas un glyphe mais est dessinée par LilyPond à l'aide de lignes et de courbes, en conséquence de quoi elle n'est pas affectée. Il en va de même pour d'autres objets tels que liaisons et ligatures.

La fonction Scheme `magstep` quant à elle effectue l'inverse : elle convertit le nombre affecté à `font-size` en facteur d'échelle.
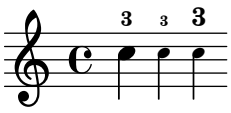
La propriété `fontSize` affecte uniquement les éléments de notation reposant sur des glyphes, tels les têtes de note, altérations, scripts, etc. Elle n'aura aucun effet sur la taille de la portée, la hauteur des hampes ou la longueur des ligatures ni sur l'espacement horizontal. L'échelonnement des hampes, ligature et espacement horizontal, couplé à la taille de la notation (sans modification de la taille de la portée), s'obtient à l'aide de la commande `\magnifyMusic` comme nous venons de le voir. La modification de l'ensemble, y compris la taille de portée, est abordé dans Section 27.2 [Définition de la taille de portée], page 707.

Dès lors que la **propriété de contexte** `fontSize` est définie, sa valeur est ajoutée individuellement à la valeur de la **propriété de** *grob* `font-size` des objets de rendu. Ceci peut être source de confusion lorsque des propriétés `font-size` individuelles sont réglées alors que `fontSize` est déjà fixé :

```
% the default font-size for NoteHead is 0
% the default font-size for Fingering is -5
c''4-3

\set fontSize = -3
% the effective font size for NoteHead is now -3
% the effective font size for Fingering is now -8
c''4-3

\override Fingering.font-size = 0
% the effective font size for Fingering is now -3
c''4-3
```
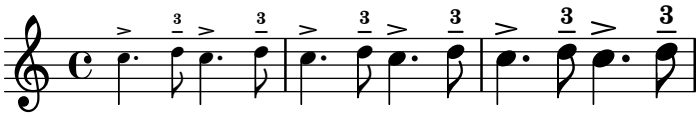


LilyPond dispose d'un certain nombre de raccourcis :

Commande	Équivalent	Taille relative
`\teeny`	`\set fontSize = -3`	71 %
`\tiny`	`\set fontSize = -2`	79 %
`\small`	`\set fontSize = -1`	89 %
`\normalsize`	`\set fontSize = 0`	100 %
`\large`	`\set fontSize = 1`	112 %
`\huge`	`\set fontSize = 2`	126 %

```
\relative c'' {
  \teeny
  c4.-> d8---3
  \tiny
  c4.-> d8---3
  \small
  c4.-> d8---3
  \normalsize
  c4.-> d8---3
  \large
  c4.-> d8---3
  \huge
  c4.-> d8---3
```

```
}
```



Pour changer la taille des caractères, LilyPond met à l'échelle la fonte dont la taille est la plus proche de la taille voulue. La taille standard (pour laquelle `font-size = 0`) dépend de la hauteur de la portée. À une portée de 20 points correspond une police de 11 points.

Commandes prédéfinies

`\magnifyMusic`, `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`.

Voir aussi

Manuel de notation : Section 27.2 [Définition de la taille de portée], page 707, Section 6.3.3 [Mise en forme d'une citation], page 277, Section 6.2.2 [Portées d'ossia], page 262, Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

Fichiers d'initialisation : `ly/music-functions-init.ly`, `ly/property-init.ly`.

Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "font-interface" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Deux bogues actuellement répertoriés font obstacle à un espacement horizontal correct avec `\magnifyMusic`. La seule façon de les contourner n'est cependant pas garantie dans toutes les situations. Dans l'exemple suivant, il vous faudra adapter la valeur de la variable *mag*. Vous pourriez aussi tenter de supprimer une ou les deux commandes `\newSpacingSection`, ou les commandes `\override` et `\revert` :
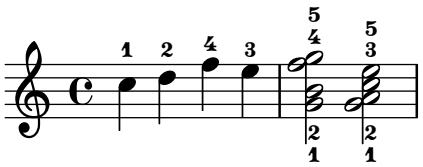
```
\magnifyMusic mag {
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #(* 1.2 mag)
  [musique]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
}
```

7.1.2 Doigtés

LilyPond dispose de deux graveurs chargés de gérer les instructions de doigté. Dans les deux cas, les doigtés peuvent s'indiquer par un '*note-chiffre_du_doigt*'.
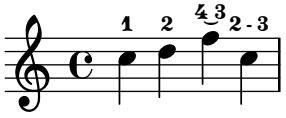
- Le premier graveur est le `Fingering_engraver`, qui se charge des instructions de doigté en dehors des constructions d'accord – autrement dit en dehors d'un `<...>`. Les doigtés apparaîtront selon leur ordre d'apparition dans le code saisi, et les *markups* de doigté sont toujours empilés verticalement en surplomb ou au-dessous de la note ou de l'accord.

```
\relative {
  c''4-1 d-2 f-4 e-3 |
  <g, b f' g>2_2_1^4^5 <g a c e>_2_1^3^5
}
```



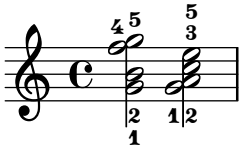
Lorsqu'un doigté doit s'indiquer sous forme textuelle, il est préférable d'utiliser la commande `\finger`.

```
\relative {
  c''4-1 d-2 f\finger \markup \tied-lyric "4~3" c\finger "2 - 3"
}
```
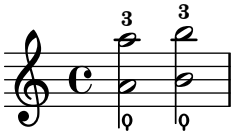


- Le second graveur est le `New_fingering_engraver` qui gère les instructions de doigté, d'articulation et d'harmonique au sein des accords – à l'intérieur d'un `<...>`.

```
\relative {
  <g'-1 b-2 f'-4 g-5>2 <e'-5 c-3 a-2 g-1>
}
```



La commande `\thumb` peut être utilisée pour indiquer, par exemple dans une partition de violoncelle, si une note doit être jouée avec le pouce (*thumb* en anglais).

```
\relative { <a'_\thumb a'-3>2 <b_\thumb b'-3> }
```



Les indications de doigtés peuvent se placer au-dessus ou en dessous de la portée – voir Section 36.1 [Direction et positionnement], page 801, à ce sujet.

Les extraits ci-après indique comment contrôler le positionnement des doigtés.

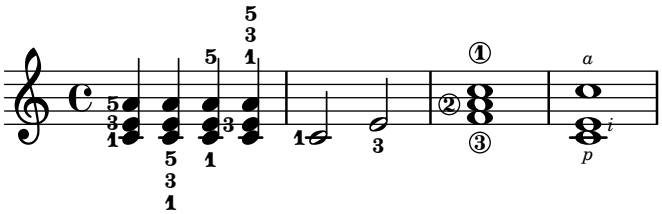
Morceaux choisis

Contrôler la position des doigtés dans un accord

Le positionnement des doigtés peut être contrôlé de manière très précise à l'aide de la propriété `fingeringOrientation`. Afin que l'orientation soit prise en compte, il est nécessaire d'utiliser une syntaxe d'accord `<...>`, même s'il ne s'agit que d'une seule note. Le positionnement des numéros de corde et doigtés main droite se règle de manière analogue à l'aide des propriétés `stringNumberOrientation` et `strokeFingerOrientation`.

Ces propriétés peuvent prendre une liste de trois valeurs. Elles contrôlent si les informations doivent être placées au-dessus (en présence d'un `up`) ou au-dessous (en présence d'un `down`), à gauche (en présence d'un `left` ou à droite (en présence d'un `right`). Par contre, si un positionnement n'est pas mentionné, aucun doigté n'y figurera. LilyPond tient compte de ces contraintes et trouvera le meilleur emplacement pour le doigté des notes de l'accord qui suit. Bien entendu, `left` et `right` sont exclusifs l'un de l'autre – les doigtés ne peuvent se placer que d'un seul côté.
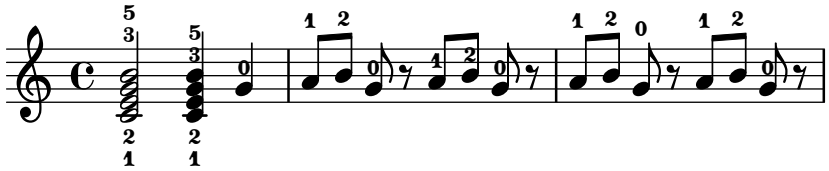
```
\relative c' {
  \set fingeringOrientations = #'(left)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(down right up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(up)
  <c-1 e-3 a-5>4
  \set fingeringOrientations = #'(left)
  <c-1>2
  \set fingeringOrientations = #'(down)
  <e-3>2
  \set stringNumberOrientations = #'(up left down)
  <f\3 a\2 c\1>1
  \set strokeFingerOrientations = #'(down right up)
  <c\rightHandFinger 1 e\rightHandFinger 2 c'\rightHandFinger 4 >
}
```



Impression des doigtés à l'intérieur de la portée

L'empilement des indications de doigté se fait par défaut à l'extérieur de la portée. Il est néanmoins possible d'annuler ce comportement. Une attention particulière doit toutefois être portée dans les cas où doigté et hampe vont dans la même direction : les indications de doigté n'évitent les hampe qu'en présence de ligature. Ce réglage peut s'adapter pour éviter toutes les hampes ou aucune d'elles. L'exemple suivant illustre ces deux options, ainsi que la manière de revenir au comportement par défaut.

```
\relative c' {
  <c-1 e-2 g-3 b-5>2
  \override Fingering.staff-padding = #'()
  <c-1 e-2 g-3 b-5>4 g'-0
  a8[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##f
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = ##t
  a[-1 b]-2 g-0 r
  \override Fingering.add-stem-support = #only-if-beamed
  a[-1 b]-2 g-0 r
}
```



Voir aussi

Manuel de notation : Section 36.1 [Direction et positionnement], page 801.

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Fingering” dans *Référence des propriétés internes*, Section “Fingering_engraver” dans *Référence des propriétés internes*, Section “fingering-event” dans *Référence des propriétés internes*, Section “FingeringEvent” dans *Référence des propriétés internes*, Section “New_fingering_engraver” dans *Référence des propriétés internes*.

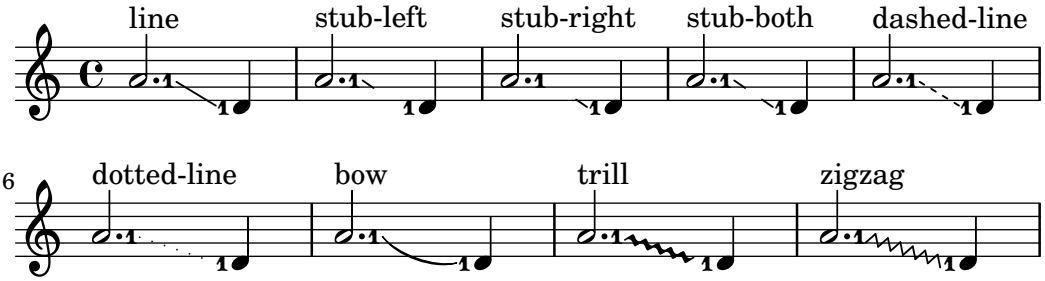
7.1.3 Glissement de doigt

En matière d’instrument à cordes, un doigt que l’on fait glisser s’indique souvent à l’aide d’un trait reliant le même doigt utilisé pour plusieurs notes jouées à des positions différentes sur une même corde. Cette ligne s’entame par un \glide placé avant l’instruction de doigté et se terminera à l’occurrence suivante du même doigt. Cette ligne peut adopter différents aspects.

```
mus = {
  \set fingeringOrientations = #'(right)
  <a'\glide-1>2.
  \set fingeringOrientations = #'(left)
  <d'-1>4
}

{
  <>^"line"
  \mus
  <>^"stub-left"
  \override FingerGlideSpanner.style = #'stub-left
  \mus
  <>^"stub-right"
  \override FingerGlideSpanner.style = #'stub-right
  \mus
  <>^"stub-both"
  \override FingerGlideSpanner.style = #'stub-both
```

```
  \mus
  <>^"dashed-line"
  \override FingerGlideSpanner.style = #'dashed-line
  \mus
  \break
  <>^"dotted-line"
  \override FingerGlideSpanner.style = #'dotted-line
  \mus
  <>^"bow"
  \override FingerGlideSpanner.style = #'bow
  \mus
  <>^"trill"
  \override FingerGlideSpanner.style = #'trill
  \mus
  <>^"zigzag"
  \override FingerGlideSpanner.style = #'zigzag
  \mus
}
```

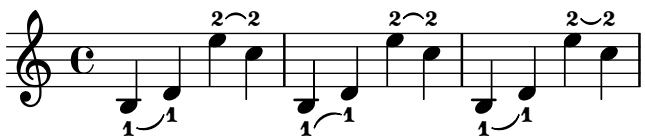


Dès lors que le `style` est défini à `'bow`, le positionnement de l'arc s'ajuste à l'aide des modificateurs d'orientation.

```
{
  \override FingerGlideSpanner.style = #'bow
  \set fingeringOrientations = #'(down)
  <b\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''\glide-2> <c''-2>

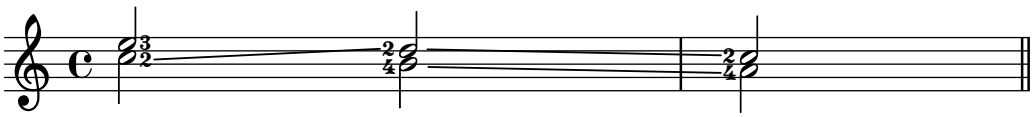
  \set fingeringOrientations = #'(down)
  <b^\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''^\glide-2> <c''-2>

  \set fingeringOrientations = #'(down)
  <b_\glide-1>4 <d'-1>
  \set fingeringOrientations = #'(up)
  <e''_\glide-2> <c''-2>
}
```



Dès lors que le Finger_glide_engraver est déplacé dans le contexte Staff, les objets Fingering de différents contextes Voice pourront être reliés.

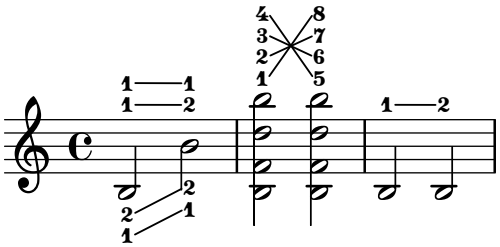
```
\score {
  \new Staff <<
    \new Voice {
      \voiceOne
      \set fingeringOrientations = #'(right)
      <e''-3>2
      \set fingeringOrientations = #'(left)
      <d''-\tweak bound-details.left.padding 2.5 \glide-2>
      <c''-2>
      \bar "||"
    }
    \new Voice {
      \voiceTwo
      \set fingeringOrientations = #'(right)
      <c''\glide-2>
      \set fingeringOrientations = #'(left)
      <b'-\tweak bound-details.left.padding 2.5 \glide-4>
      <a'-4>
    }
  >>
  \layout {
    ragged-right = ##f
    \context {
      \Voice
      \remove Finger_glide_engraver
    }
    \context {
      \Staff
      \consists Finger_glide_engraver
    }
  }
}
```



La connection entre différents doigtés ou entre plusieurs instances d'un même doigt s'obtient en réglant la propriété id à l'aide d'un \= qui prend en argument un entier positif ou un symbole, ou bien en appliquant un \tweak à la propriété text.

```
{
  b2 \glide \= #'foo ^1
     \glide \= #'bar ^1
     \glide _2
     \glide _1
  b' \= #'foo ^2
     \= #'bar ^1
     _2
     _1
```

```
  \set fingeringOrientations = #'(up)
  <
   b\glide \=1 -1
   f'\glide \=2 -2
   d''\glide \=3 -3
   b''\glide \=4 -4
  >
  <
   b\=4 -5
   f'\=3 -6
   d''\=2 -7
   b''\=1 -8
  >
  b\glide -1 b\tweak text "2" -1
}
```

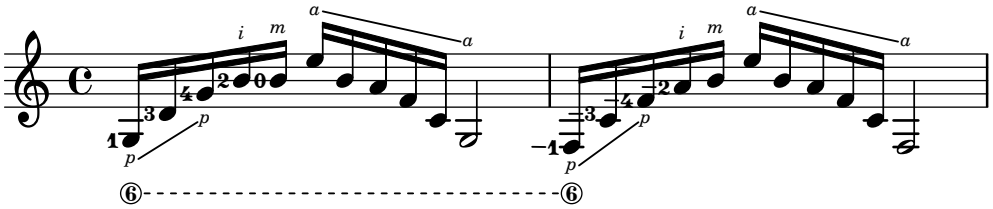


Le FingerGlideSpanner peut aussi relier des numéros de cordes, indiquant ainsi de jouer sur la même corde, ou bien un glissement de doigts indiquant un *Arrastre*.

```
{
  \override StringNumber.staff-padding = 7
  \omit TupletNumber
  \set fingeringOrientations = #'(left)
  \tuplet 5/4 4 {
    \set strokeFingerOrientations = #'(down)
    <
     g-\tweak style #'stub-right \glide-1
       \glide \rightHandFinger #1
       -\tweak style #'dashed-line \glide _\6
    >16
    <d'-\tweak style #'stub-right \glide -3 >
    <g'-\tweak style #'stub-right \glide -4 \rightHandFinger #1 >
    \set strokeFingerOrientations = #'(up)
    <b'-\tweak style #'stub-right \glide -2 \rightHandFinger #2 >
    <b'-0\rightHandFinger #3 >
    e''\glide \rightHandFinger #4
    b' a' f' c'
  }
  g2\rightHandFinger #4

  \tuplet 5/4 4 {
    \set strokeFingerOrientations = #'(down)
    <f-1 \glide \rightHandFinger #1 _\6 >16
    % Raise a bit, otherwise the stub-line would be hidden
    % by the ledger line.
```

```
    <c'\tweak Y-offset #0.5 -3>
    <f' -4 \rightHandFinger #1 >
    \set strokeFingerOrientations = #'(up)
    <a'-2\rightHandFinger #2 >
    b'\rightHandFinger #3
    e''\glide \rightHandFinger #4
    b' a' f' c'
  }
  f2\rightHandFinger #4
}
```



Voir aussi

Glossaire musicologique : Section “arrastre” dans *Glossaire*.

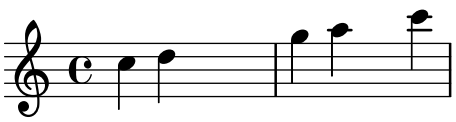
Manuel de notation : Section 36.1 [Direction et positionnement], page 801.

Manuel des références internes : Section “FingerGlideEvent” dans *Référence des propriétés internes*, Section “finger-glide-event” dans *Référence des propriétés internes*, Section “Finger_glide_engraver” dans *Référence des propriétés internes*, Section “finger-glide-interface” dans *Référence des propriétés internes*, Section “FingerGlideSpanner” dans *Référence des propriétés internes*.

7.1.4 Dictée à trous
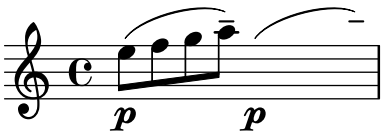
Les notes masquées – ou invisibles ou encore transparentes – sont utiles dans le cadre d’exercices de théorie ou de composition.

```
\relative {
  c''4 d
  \hideNotes
  e4 f
  \unHideNotes
  g a
  \hideNotes
  b
  \unHideNotes
  c
}
```



Têtes de note, hampes, crochets et silences sont invisibles. Une ligature sera invisible si elle démarre sur une note invisible. Les objets de notation attachés à une note invisible ne seront pas masqués pour autant.

```
\relative c'' {
  e8(\p f g a)--
  \hideNotes
  e8(\p f g a)--
}
```



Commandes prédéfinies

`\hideNotes`, `\unHideNotes`.

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 6.2.3 [Masquage de portées], page 265, Section 2.2.2 [Silences invisibles], page 71, Section 36.7 [Visibilité des objets], page 812.

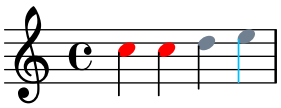
Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Note_spacing_engraver" dans *Référence des propriétés internes*, Section "NoteSpacing" dans *Référence des propriétés internes*.

7.1.5 Coloration d'objets
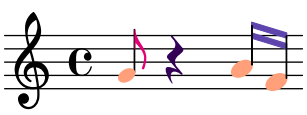
Des objets peuvent être colorisés individuellement. Une liste des noms des couleurs disponibles se trouve à l'annexe Section B.7 [Liste des couleurs], page 930.

```
\override NoteHead.color = #red
c''4 c''
\override NoteHead.color = #(x11-color 'SlateGrey)
d''
\override Stem.color = "deepskyblue"
e''
```



En plus du jeu limité de couleurs de base prédéfini sous forme de variable – voir les « couleurs normales » dans Section B.7 [Liste des couleurs], page 930, – n'importe quelle couleur peut être spécifiée sous forme de chaîne, qu'il s'agisse d'un nom de couleur prédéfini dans le style CSS (`https://www.w3.org/Style/CSS/`) ou d'un code hexadécimal préfixé d'un '`#`' (le tout entre guillemets informatiques) :

```
\override NoteHead.color = "lightsalmon"
\override Flag.color = "#E30074"
\override Beam.color = "#5e45ad"
\override Rest.color = "#3058"
g'8 \huge r4 a'16 f'
```



Lorsque ce code couleur comporte un canal alpha pour la transparence, spécifier un code sur huit caractères "*#RRGGBBAA*", ou sa forme abrégée "*#RGBA*", produira ses effets pour une
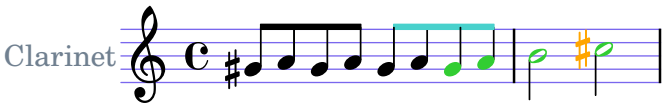
sortie SVG, mais pas pour du PostScript ni du PDF. Dans l'exemple précédent, on peut voir à travers le silence dès lors qu'il est compilé avec le moteur SVG, comme expliqué dans Section 23.3 [Formats de sortie alternatifs], page 668.

De manière différente, l'intégralité de la palette des couleurs définies pour X11 (`https://en.wikipedia.org/wiki/X11_color_names`) est accessible par la fonction Scheme `x11-color`. Cette fonction prend en argument une expression symbolique de la forme `'DarkSeaGreen4` ou bien une chaîne de caractères comme `"DarkSeaGreen4"`. La première formulation est à la fois plus rapide à écrire et aussi plus efficace. Néanmoins, la deuxième forme permet d'accéder aux noms composés des couleurs de X11 comme ici `"dark sea green 4"`.

Lorsque la fonction `x11-color` ne trouve pas le paramètre fourni, elle revient à la couleur par défaut, le noir. Le problème ressort de façon évidente au vu de la partition finale.

```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'SlateGrey) "Clarinet"
    }
  }

  \relative c'' {
  \override Staff.StaffSymbol.color = #(x11-color 'SlateBlue2)
  gis8 a
  \override Beam.color = #(x11-color "medium turquoise")
  gis a
  \override Accidental.color = #(x11-color 'orange)
  gis a
  \override NoteHead.color = #(x11-color "LimeGreen")
  gis a
  % this is deliberate nonsense; note that the stems remain black
  \override Stem.color = #(x11-color 'Boggle)
  b2 cis
}
```
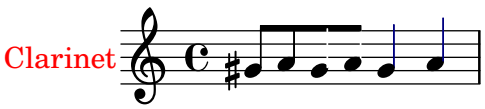


LilyPond prend aussi en charge un jeu de huit noms de couleur (`https://jfly.uni-koeln.de/color`) sans ambiguïté pour les gens souffrant de daltonisme ou non. Il est accessible sous le nom de `universal-color`.

```
\markup \with-color #(universal-color 'vermillion) vermillion
```

vermillion

Un autre moyen consiste à fournir à la fonction Scheme `rgb-color` les composantes de la couleur exacte au format rouge-vert-bleu (*RGB*) – chacune étant exprimée en en fraction de 256 (le 0.5 de l'exemple suivant correspond à 128) – ainsi qu'éventuellement un nombre définissant le niveau de transparence. Une fois de plus, cette transparence n'est effective que dans le cadre du format SVG, où la clef de l'extrait ci-dessous sera affectée.

```
\new Staff \with {
  instrumentName = \markup {
    \with-color #(x11-color 'red) "Clarinet"
    }
  \override Clef.color = #(rgb-color 0 0 0 0.5)
  }
\relative c'' {
  \override Stem.color = #(rgb-color 0 0 0)
  gis8 a
  \override Stem.color = #(rgb-color 1 1 1)
  gis8 a
  \override Stem.color = #(rgb-color 0 0 0.5)
  gis4 a
}
```



Voir aussi

Manuel de notation : Section 35.6 [Les commandes \tweak et \single], page 791, Section B.7 [Liste des couleurs], page 930.

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Problèmes connus et avertissements

Une couleur x11 n’aura pas forcément le même rendu qu’une couleur normale ayant un nom similaire.
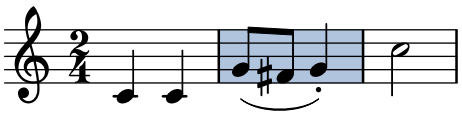
Les couleurs de X11 ne sont pas toutes perceptibles dans un navigateur internet. Aussi nous vous recommandons, dans le cadre d’une présentation multimédia, d’utiliser des couleurs CSS.

Vous ne pouvez pas coloriser individuellement des notes à l’intérieur d’un accord avec \override. Si besoin est, utilisez \tweak ou \single\override devant la note en question. Pour plus de détails, consultez Section 35.6 [Les commandes \tweak et \single], page 791.

7.1.6 Surlignage de portées

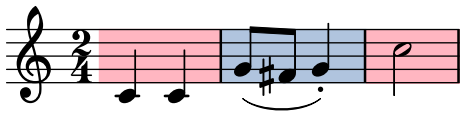
Pour des besoins pédagogiques ou d’analyse, il peut être utile de mettre en évidence un fragment musical comme, par exemple, pour indiquer une modulation. Ceci peut se réaliser à l’aide de la commande \staffHighlight complétée d’une couleur – pour les manières de saisir des couleurs, voir Section 7.1.5 [Coloration d’objets], page 298. Le surlignage s’interrompt par la commande \stopStaffHighlight.

```
\relative {
  \time 2/4
  c'4 4
  \staffHighlight "lightsteelblue"
  g'8( fis g4)-.
  \stopStaffHighlight
  c2
}
```



Lorsque plusieurs surlignages se succèdent, nul n'est besoin de stipuler un `\stopStaffHighlight`, dans la mesure où `\staffHighlight` interrompt implicitement l'éventuel surlignage en cours. Il n'est pas non plus nécessaire de positionner un `\stopStaffHighlight`. Ces manières implicites de terminer un surlignage sont particulièrement adaptées lorsque chaque mesure est surlignée.

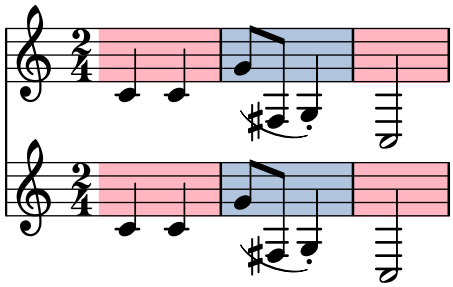
```
\relative {
  \time 2/4
  \staffHighlight "lightpink"
  c'4 4
  \staffHighlight "lightsteelblue"
  g'8( fis g4)-.
  \staffHighlight "lightpink"
  c2
}
```



Les portées sont par défaut surlignées de manière indépendante.

```
music = {
  \time 2/4
  \staffHighlight "lightpink"
  c'4 4
  \staffHighlight "lightsteelblue"
  g'8( fis g4)-.
  \staffHighlight "lightpink"
  c2
}

<<
  \new Staff \music
  \new Staff \music
>>
```

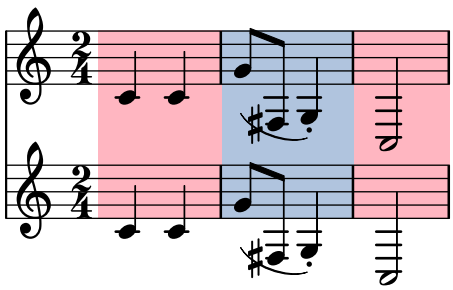


Plusieurs portées peuvent toutefois être surlignées de concert dès lors que le `Staff_highlight_engraver` aura été déplacé dans un contexte supérieur à `Staff` (ou bien `RhythmicStaff` ou similaire). Il suffit en pareil cas d'utiliser les commandes `\consists` et `\remove` – voir Section 33.4 [Modification des greffons de contexte], page 769, pour de plus amples informations. Le surlignage de toutes les portées s'obtient donc en déplaçant le graveur dans le contexte `Score`.

```
\layout {
  \context {
    \Staff
    \remove Staff_highlight_engraver
  }
  \context {
    \Score
    \consists Staff_highlight_engraver
  }
}

music = {
  \time 2/4
  \staffHighlight "lightpink"
  c'4 4
  \staffHighlight "lightsteelblue"
  g'8( fis g4)-.
  \staffHighlight "lightpink"
  c2
}

<<
  \new Staff \music
  \new Staff \music
>>
```

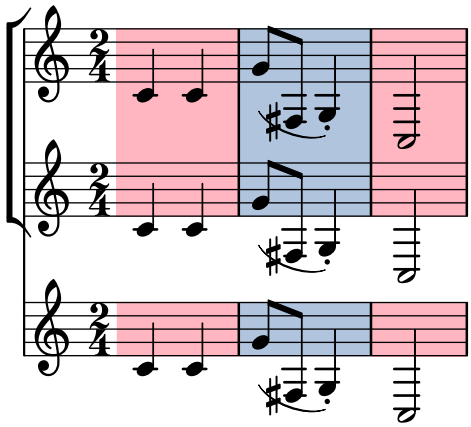


Le Staff_highlight_engraver peut aussi être déplacé dans des contextes intermédiaires comme un StaffGroup.

```
music = {
  \time 2/4
  \staffHighlight "lightpink"
  c'4 4
  \staffHighlight "lightsteelblue"
  g'8( fis g4)-.
  \staffHighlight "lightpink"
  c2
}

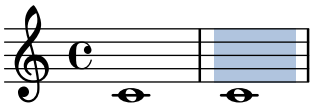
<<
  \new StaffGroup \with { \consists Staff_highlight_engraver } <<
    \new Staff \with { \remove Staff_highlight_engraver } \music
    \new Staff \with { \remove Staff_highlight_engraver } \music
  >>
```

```
  \new Staff \music
>>
```



La propriété `StaffHighlight.shorten-pair` permet d'ajuster les points de début et de fin de l'étendue du surlignage.

```
{
  c'1
  \once \override Staff.StaffHighlight.shorten-pair = #'(1.0 . 1.0)
  \staffHighlight lightsteelblue
  c'1
}
```



Commandes prédéfinies

`\staffHighlight`, `\stopStaffHighlight`.

Voir aussi

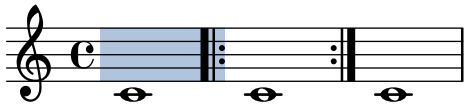
Manuel de notation : Section 7.1.5 [Coloration d'objets], page 298, Section 33.4 [Modification des greffons de contexte], page 769.

Référence des propriétés internes : Section "StaffHighlight" dans *Référence des propriétés internes*, Section "staff-highlight-interface" dans *Référence des propriétés internes*, Section "Staff_highlight_engraver" dans *Référence des propriétés internes*, Section "StaffHighlight-Event" dans *Référence des propriétés internes*, Section "staff-highlight-event" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Le surlignage peut se révéler quelque peu hasardeux en présence d'une barre de début de reprise. La propriété `shorten-pair` vue ci-dessus permet alors de contourner le problème.
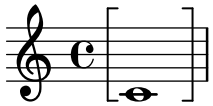
```
{
  \staffHighlight "lightsteelblue"
  c'1
  \stopStaffHighlight
  \repeat volta 2 { c'1 }
  c'1
}
```



7.1.7 Crochets de matériau optionel

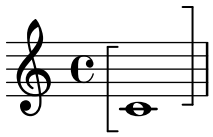
L'addition de matériau optionnel peut s'indiquer à l'aide de crochets englobant la portée.

```
{
  \startOptionalMaterial
  c'1
  \stopOptionalMaterial
}
```



Les positions de l'annotation que le crochet encadre peuvent se modifier. L'intervalle entre les extrémités est mesuré en espace de portée à partir de la ligne médiane.

```
{
  \tweak OptionalMaterialBracket.positions #'(-4 . 1)
  \startOptionalMaterial
  c'1
  \once \override Staff.OptionalMaterialBracket.positions =
  #'(-2 . 4)
  \stopOptionalMaterial
}
```



Commandes prédéfinies

`\startOptionalMaterial`, `\stopOptionalMaterial`.

Voir aussi
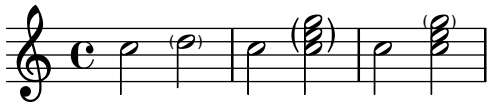
Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Optional_material_bracket_engraver" dans *Référence des propriétés internes*, Section "OptionalMaterialBracket" dans *Référence des propriétés internes*, Section "optional-material-bracket-interface" dans *Référence des propriétés internes*.

7.1.8 Parenthèses
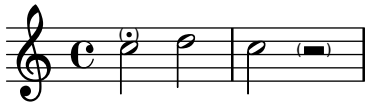
Des objets peuvent être mis entre parenthèses en saisissant `\parenthesize` juste avant l'événement musical.

```
\relative {
  c''2 \parenthesize d
  c2  \tweak Parentheses.font-size 2 \parenthesize <c e g>
  c2 <c e \parenthesize g>
}
```
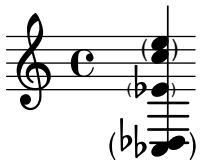


Les objets autres que des notes peuvent aussi être entre parenthèses. En ce qui concerne les articulations, l'instruction \parenthesize doit cependant être précédée d'un tiret.

```
\relative {
  c''2-\parenthesize -. d
  c2 \parenthesize r
}
```



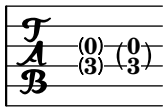
Lorsque seulement certaines notes d'un accord doivent être mises entre parenthèses, est requise l'utilisation d'une construction en parallèle << ... >>.

```
\new Voice \relative c {
  <<
    { \tweak Parentheses.font-size 0 \parenthesize <ces des> }
    { \parenthesize ees' }
    { \tweak Parentheses.font-size -2 \parenthesize <c' e> }
  >>
}
```
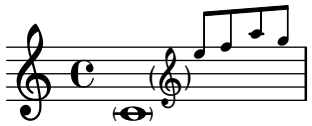


Dans le cas d'une tablature devra être spécifié NoteColumn pour mettre un accord entre parenthèses.

```
\new TabVoice {
  \override Parentheses.font-size = 0
  \parenthesize <f g>
  \parenthesize NoteColumn <f g>
}
```



Cette deuxième forme de la commande \parenthesize implique un chemin vers un objet graphique : \parenthesize *NomContexte.NomGrob*, voire même \parenthesize *NomGrob* dès lors que l'objet à parenthéser est créé dans le contexte le plus bas (typiquement Voice). La commande doit alors se libeller avant le moment musical, à l'instar d'un \once \override. Cette forme rend possible la mise entre parenthèses des objets dont l'origine n'est pas directement liée à un événement.

```
\new Staff \relative <<
  {
    \parenthesize NoteHead
    c'1
  }
  \new CueVoice {
    s2
    \voiceOne
    \once \override Staff.Parentheses.font-size = 3
    \parenthesize Staff.CueClef
    \cueClef treble
    e'8 f a g
  }
>>
```



Voir aussi

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Parenthesis_engraver” dans *Référence des propriétés internes*, Section “Parentheses” dans *Référence des propriétés internes*, Section “parentheses-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La propriété `font-size` de l’objet graphique `Parentheses` doit, à ce jour, être ajustée manuellement afin d’obtenir une taille de parenthèses correcte dans le cas des accords entre autres.

7.1.9 Hampes

Dès qu’une note est rencontrée, un objet `Stem` se crée automatiquement. Pour les rondes et les silences, ils sont aussi créés, mais en mode invisible.

L’orientation des hampes peut être définie manuellement – voir Section 36.1 [Direction et positionnement], page 801, à ce sujet.

Commandes prédéfinies
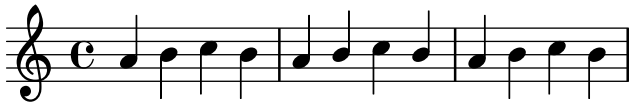
`\stemUp`, `\stemDown`, `\stemNeutral`.

Morceaux choisis

Direction par défaut des hampes de la ligne médiane

La direction des hampes des notes placées sur la ligne médiane de la portée est gérée par la propriété `neutral-direction` de l’objet `Stem`.

```
\relative c'' {
  a4 b c b
  \override Stem.neutral-direction = #up
  a4 b c b
  \override Stem.neutral-direction = #down
  a4 b c b
}
```
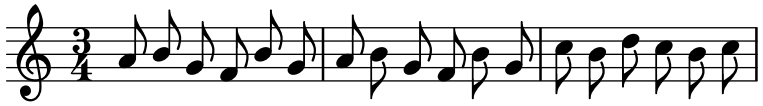


Changement automatique de l'orientation de hampe de la note médiane selon la mélodie

Afin de suivre la ligne mélodique, LilyPond peut inverser l'orientation de hampe de la note médiane, dès lors qu'aura été ajouté au contexte de voix le graveur `Melody_engraver`.

La propriété de contexte `suspendMelodyDecisions` permet, si besoin, de désactiver temporairement ce comportement.

```
\relative c'' {
  \time 3/4
  a8 b g f b g |
  \set suspendMelodyDecisions = ##t
  a  b g f b g |
  \unset suspendMelodyDecisions
  c  b d c b c |
}

\layout {
  \context {
    \Voice
    \consists "Melody_engraver"
    \autoBeamOff
  }
}
```



Voir aussi

Manuel de notation : Section 36.1 [Direction et positionnement], page 801.

Morceaux choisis : Section "Annotations éditoriales" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Stem_engraver" dans *Référence des propriétés internes*, Section "Stem" dans *Référence des propriétés internes*, Section "stem-interface" dans *Référence des propriétés internes*.

7.2 Hors de la portée
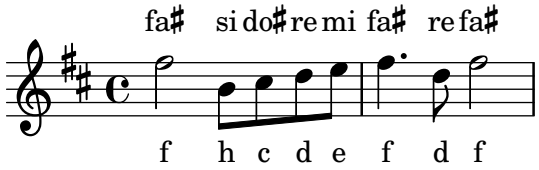
Nous allons nous intéresser ici à souligner des éléments inscrits dans la portée par des éléments qui lui seront externes.

7.2.1 Nom des notes

Le nom des notes peut s'imprimer sous forme de texte à l'aide du contexte `NoteNames`. Utilisé conjointement à une portée standard, chaque hauteur sera synchronisée avec son nom, qu'il soit imprimé au-dessus ou en dessous de la portée.

```
\language "italiano"
melody = \relative do'' {
  fad2 si,8 dod re mi fad4. re8 fad2
}

<<
  \new NoteNames { \melody }
  \new Staff { \key si \minor \melody }
  \new NoteNames {
    \set printNotesLanguage = "deutsch"
    \set printAccidentalNames = ##f
    \melody
  }
>>
```

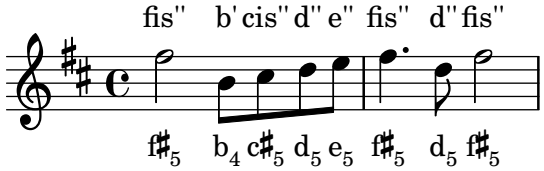


Le nom des notes est imprimé par défaut dans la langue utilisée pour la saisie ; la propriété `printNotesLanguage` permet cependant de sélectionner l'une des autres langues disponibles – voir Section 1.1.5 [Nom des notes dans d'autres langues], page 11. La présentation des altérations se gère à l'aide de la propriété `printAccidentalNames`.

L'utilisation conjointe de ces deux propriétés et l'activation de `printOctaveNames` donnera une représentation comparable à la syntaxe de saisie de LilyPond. Pour une représentation plus généraliste, on peut utiliser des numéros d'octave.

```
melody = \relative c'' {
  fis2 b,8 cis d e fis4. d8 fis2
}

<<
  \new NoteNames {
    \set printOctaveNames = ##t
    \set printAccidentalNames = #'lily
    \melody
  }
  \new Staff { \key b \minor \melody }
  \new NoteNames {
    \set printOctaveNames = #'scientific
    \melody
  }
>>
```

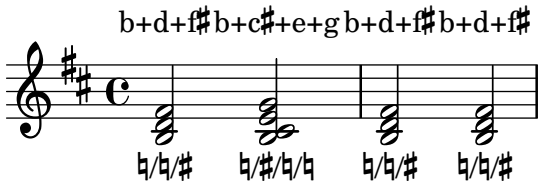


La propriété `noteNameSeparator` définit comment les accords seront représentés. D'autres fonctions de formatage peuvent se définir en tant que `noteNameFunction`. Une telle fonction

doit attendre une hauteur (pitch) et un contexte en tant qu'arguments même si l'un peut être ignoré.

```
somechords = \relative c' {
  <b d fis>2 <b cis e g> <b d fis> q
}

<<
  \new NoteNames {
    \set noteNameSeparator = "+"
    \somechords
  }
  \new Staff { \key b \minor \somechords }
  \new NoteNames {
    \set noteNameFunction =
      #(lambda (pitch ctx)
         (accidental->text-markup
          (ly:pitch-alteration pitch)))
    \somechords
  }
>>
```



Voir aussi

Manuel de notation : Section 1.1.5 [Nom des notes dans d'autres langues], page 11.

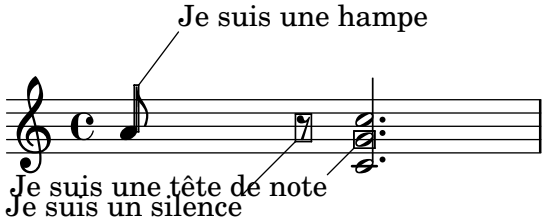
Référence des propriété internes : Section "NoteName" dans *Référence des propriétés internes*, Section "NoteNames" dans *Référence des propriétés internes*, Section "Note_name_engraver" dans *Référence des propriétés internes*.

7.2.2 Info-bulle

Vous pouvez marquer et nommer des éléments de notation à l'aide de bulles. L'objectif premier de cette fonctionnalité est d'expliquer la notation.

En voici un exemple :

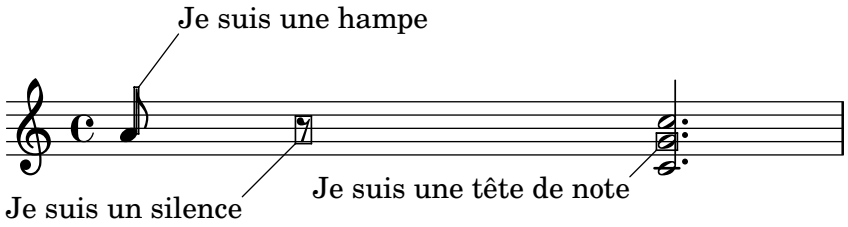
```
\new Voice \with { \consists Balloon_engraver }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Je suis un silence" }
  r
  <c, g'-\balloonText #'(-2 . -2) \markup { "Je suis une tête de note" } c>2.
}
```
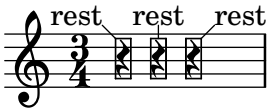


Vous disposez de deux fonctions musicales, balloonGrobText et balloonText. balloonGrobText prend en argument l'objet graphique à agrémenter et s'utilise comme \once \override. balloonText, quant à lui, s'utilise comme une simple articulation et fonctionne comme \tweak pour attacher du texte à l'une des notes d'un accord. Les autres arguments sont le décalage et le texte de la bulle.

Les info-bulles n'ont aucune influence sur l'espacement des notes ; on peut toutefois le gérer ainsi :

```
\new Voice \with { \consists Balloon_engraver }
\relative c'' {
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  a8
  \balloonGrobText #'Rest #'(-4 . -4) \markup { "Je suis un silence" }
  r
  \balloonLengthOn
  <c, g'-\balloonText #'(-2 . -2) \markup { "Je suis une tête de note" } c>2.
}
```
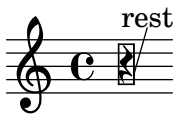


Le comportement par défaut en matière de point d'attachement du trait entre l'encadrement et le texte est indiqué ci-dessous.



Le calcul par défaut du point d'attachement peut se corriger à l'aide des propriétés X-attachment et Y-attachment qui peuvent prendre une valeur entre −1 et 1, valeurs qui correspondent à l'extrémité gauche (ou basse) et droite (ou haute) du cadre. L'alignement du texte se gère à l'aide de text-alignment-X et text-alignment-Y, de manière similaire.

```
\new Voice \with {
  \consists Balloon_engraver
}
{
  \once \override BalloonText.Y-attachment = -0.5
  \once \override BalloonText.text-alignment-X = 0.0
  \balloonGrobText Rest #'(1 . 3.5) "rest"
  r4
}
```



Commandes prédéfinies

\balloonLengthOn, \balloonLengthOff.

Voir aussi

Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “AnnotateOutputEvent” dans *Référence des propriétés internes*, Section “Balloon_engraver” dans *Référence des propriétés internes*, Section “BalloonText” dans *Référence des propriétés internes*, Section “balloon-interface” dans *Référence des propriétés internes*.

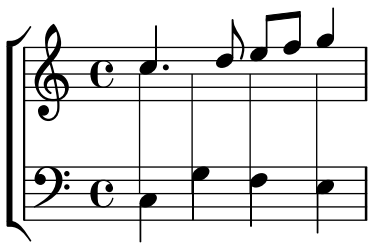
7.2.3 Quadrillage temporel

Vous pouvez tracer des lignes entre les portées, synchronisées avec les notes.

LilyPond a recours à deux graveurs distincts afin d’afficher le quadrillage : le `Grid_point_engraver` se charge de déterminer l’envergure du crochet, alors que le `Grid_line_span_engraver` se consacrera au tracé des lignes. Les lignes sont par défaut centrées horizontalement sous les notes et alignées sur la gauche des têtes. La propriété `gridInterval` spécifie quant à elle l’espace de temps entre chaque ligne.

```
\layout {
  \context {
    \Staff
    \consists Grid_point_engraver %% active les guides
    gridInterval = #1/4
  }
  \context {
    \Score
    \consists Grid_line_span_engraver
     %% centre les lignes guides horizontalement sous les notes
  }
}

\score {
  \new ChoirStaff <<
    \new Staff \relative {
      \stemUp
      c''4. d8 e8 f g4
    }
    \new Staff \relative {
    %% centre les lignes guides verticalement
      \clef bass
      \stemDown
      c4  g'  f  e
    }
  >>
}
```



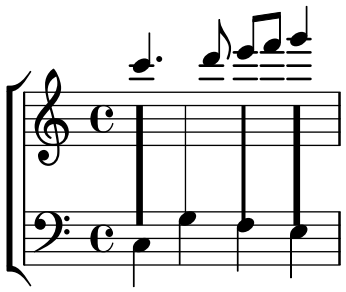
Morceaux choisis

Apparence du quadrillage temporel

Modifier certaines des propriétés du quadrillage temporel aura pour effet d'en changer l'apparence.

```
\new ChoirStaff <<
  \new Staff {
    \relative c'' {
      \stemUp
      c'4. d8 e8 f g4
    }
  }
  \new Staff {
    \relative c {
      % this moves them up one staff space from the default position
      \override Score.GridLine.extra-offset = #'(0.0 . 1.0)
      \stemDown
      \clef bass
      \once \override Score.GridLine.thickness = 5.0
      c4
      \once \override Score.GridLine.thickness = 1.0
      g'4
      \once \override Score.GridLine.thickness = 3.0
      f4
      \once \override Score.GridLine.thickness = 5.0
      e4
    }
  }
>>

\layout {
  \context {
    \Staff
    % set up grids
    \consists "Grid_point_engraver"
    % set the grid interval to one quarter note
    gridInterval = #1/4
    }
  \context {
    \Score
    \consists "Grid_line_span_engraver"
    % this moves them to the right half a staff space
    \override NoteColumn.X-offset = -0.5
  }
}
```



Voir aussi
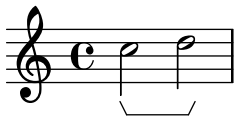
Morceaux choisis : Section “Annotations éditoriales” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Grid_line_span_engraver” dans *Référence des propriétés internes*, Section “Grid_point_engraver” dans *Référence des propriétés internes*, Section “GridLine” dans *Référence des propriétés internes*, Section “GridPoint” dans *Référence des propriétés internes*, Section “grid-line-interface” dans *Référence des propriétés internes*, Section “grid-point-interface” dans *Référence des propriétés internes*.

7.2.4 Crochets d’analyse

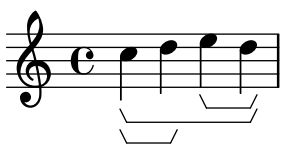
On utilise des crochets en analyse musicale, pour indiquer la structure d’une pièce.

```
\layout {
  \context {
    \Voice
    \consists Horizontal_bracket_engraver
  }
}
\relative {
  c''2\startGroup
  d\stopGroup
}
```



Les crochets d’analyse sont susceptibles d’être imbriqués :

```
\layout {
  \context {
    \Voice
    \consists Horizontal_bracket_engraver
  }
}
\relative {
  c''4\startGroup\startGroup
  d4\stopGroup
  e4\startGroup
  d4\stopGroup\stopGroup
}
```



Morceaux choisis

Crochets d'analyse au-dessus de la portée

Les crochets d'analyse viennent par défaut se positionner au-dessous de la portée. L'exemple suivant vous indique comment les faire apparaître en surplomb de la portée.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}

\relative c'' {
  \once \override HorizontalBracket.direction = #UP
  c2\startGroup
  d2\stopGroup
}
```



Crochet d'analyse avec texte

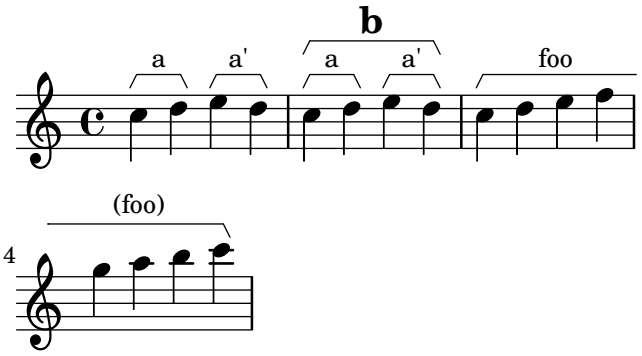
Un *markup* textuel peut venir s'ajouter aux crochets d'analyse grâce à la propriété `text` de l'objet graphique `HorizontalBracketText`. Plusieurs crochets présents en un même moment requièrent d'utiliser la commande `\tweak`.

Le texte ajouté sera répété, entre parenthèse, après un saut de ligne. L'ordre vertical des crochets imbriqués se gère à l'aide de la propriété `outside-staff-priority`.

```
\layout {
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
    \override HorizontalBracket.direction = #UP
  }
}

{
  \once\override HorizontalBracketText.text = "a"
    c''\startGroup d''\stopGroup
    \once\override HorizontalBracketText.text = "a'"
    e''\startGroup d''\stopGroup |
  c''-\tweak outside-staff-priority #801
      \tweak HorizontalBracketText.text
        \markup \bold \huge "b" \startGroup
     -\tweak HorizontalBracketText.text "a" \startGroup
    d''\stopGroup
    e''-\tweak HorizontalBracketText.text "a'" \startGroup
    d''\stopGroup\stopGroup |
  c''-\tweak HorizontalBracketText.text foo \startGroup
    d'' e'' f'' | \break
  g'' a'' b'' c'''\stopGroup
```

```
}
```



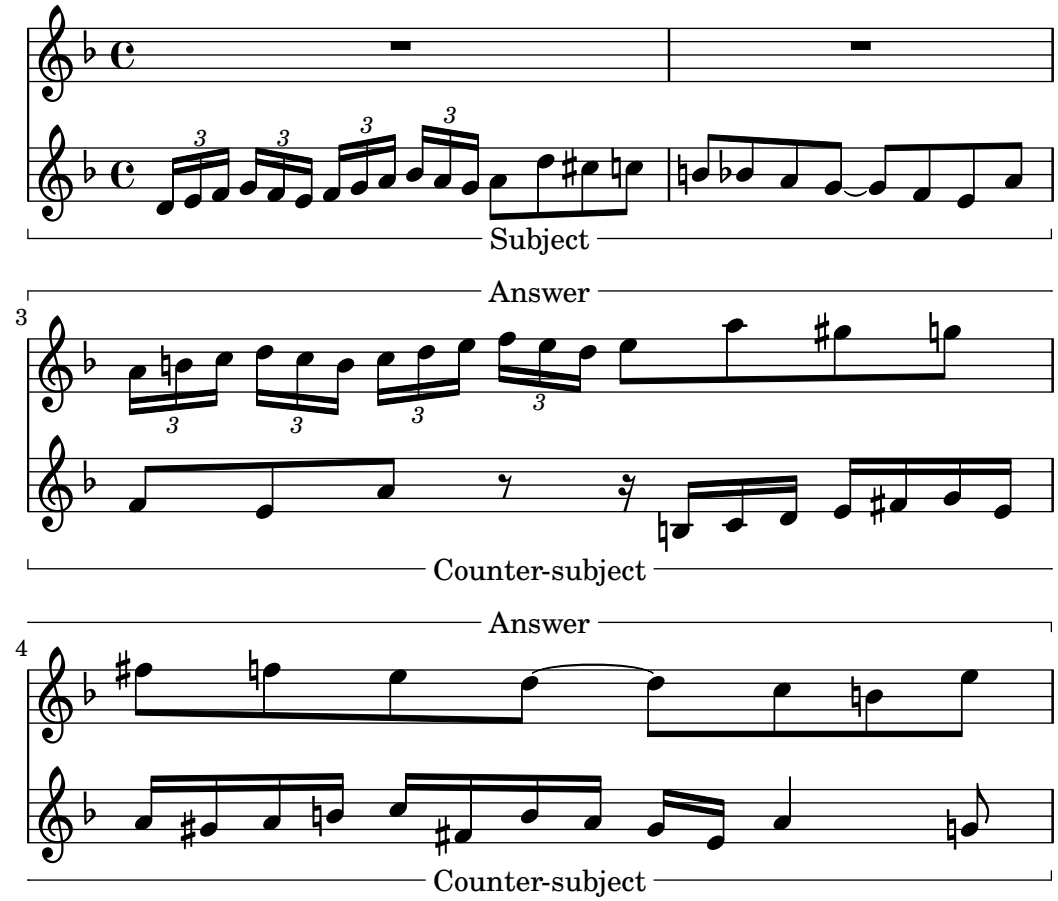
Crochet de mesure

Les bandeaux sur mesure sont un moyen alternatif d'imprimer un crochet annoté. Contrairement aux crochets horizontaux, ces bandeaux s'étendent entre deux barres de mesure plutôt qu'entre deux notes. Le texte est centré sur l'empan du crochet.

```
\layout {
  \context {
    \Staff
    \consists Measure_spanner_engraver
  }
}

<<
  \new Staff \relative c'' {
    \key d \minor
    R1*2
    \tweak text "Answer"
    \startMeasureSpanner
    \tuplet 3/2 8 {
      a16[ b c] d[ c b]  c[ d e] f[ e d]
    }
    e8 a gis g
    fis f e d~ d c b e
    \stopMeasureSpanner
  }
  \new Staff \relative c' {
    \key d \minor
    \tweak text "Subject"
      \tweak direction #DOWN
      \startMeasureSpanner
    \tuplet 3/2 8 {
      d16[ e f] g[ f e] f[ g a] bes[ a g]
    }
    a8 d cis c
    b bes a g~ g f e a
    \stopMeasureSpanner
    \tweak text "Counter-subject"
      \tweak direction #DOWN
      \startMeasureSpanner
    f8 e a r r16 b, c d e fis g e
```
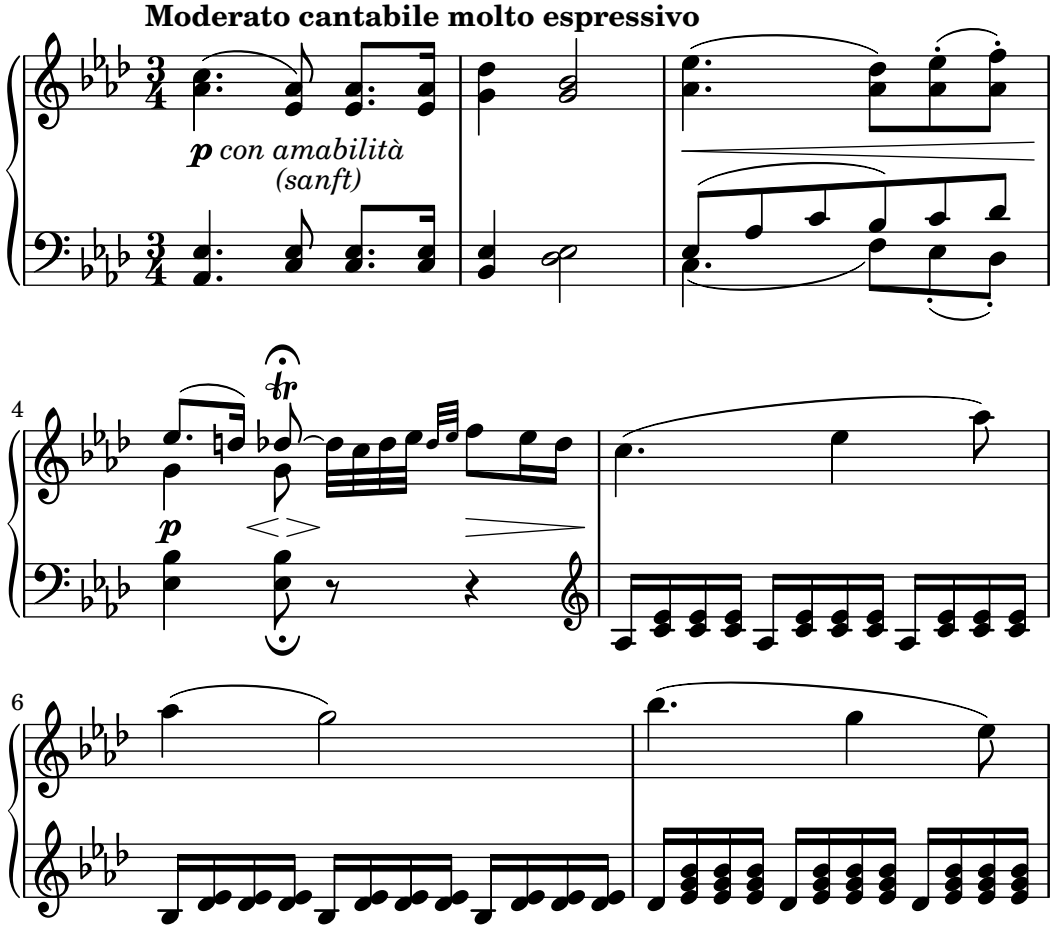
```
    a gis a b c fis, b a gis e a4 g8
    \stopMeasureSpanner
  }
>>
```



Voir aussi

Référence des propriétés internes : Section “Horizontal_bracket_engraver” dans *Référence des propriétés internes*, Section “HorizontalBracket” dans *Référence des propriétés internes*, Section “horizontal-bracket-interface” dans *Référence des propriétés internes*, Section “HorizontalBracketText” dans *Référence des propriétés internes*, Section “horizontal-bracket-text-interface” dans *Référence des propriétés internes*, Section “Measure_spanner_engraver” dans *Référence des propriétés internes*, Section “MeasureSpanner” dans *Référence des propriétés internes*, Section “measure-spanner-interface” dans *Référence des propriétés internes*, Section “Staff” dans *Référence des propriétés internes*.

8 Texte



Nous allons voir ici comment insérer du texte dans une partition, ainsi que différentes manières de le mettre en forme.

8.1 Ajout de texte

Cette partie constitue une introduction aux différentes manières d'ajouter du texte à une partition.

> **Note :** Pour écrire des accents et autres caractères spéciaux, il suffit de les insérer directement dans votre fichier LilyPond. Ce fichier devra être sauvegardé avec l'encodage UTF-8. Pour plus d'informations, voir Section 22.4.1 [Codage du texte], page 664.

8.1.1 Vue d'ensemble des objets textuels

Les objets textuels se saisissent soit comme de simples chaînes de caractères entre guillemets informatiques (optionnels pour un mot unique). Le mode *markup*, ou étiquette, accepte quant à lui une mise en forme fine du texte et des enrichissements graphiques comme indiqué dans Section 8.2 [Mise en forme du texte], page 329.

En tant que tels, les blocs *markup* peuvent s'utiliser :

- dans tout objet `TextScript` attaché à des notes par `-`, `^` ou `_` ; voir Section 8.1.2 [Commentaires textuels], page 320 ;

- en tant que bandeau (*spanner*) lorsqu'une indication se prolonge sur plusieurs temps ou mesures ; voir Section 8.1.3 [Indication textuelle avec extension], page 321 ;
- dans toute marque affichée au-dessus de la partition, tels les objets `RehearsalMark` ou `MetronomeMark` introduits respectivement par les mots-clés `\mark` ou `\tempo` ; voir Section 8.1.5 [Indications textuelles], page 323 ;
- en tant que bloc de texte indépendant, saisi en dehors de tout bloc `\score` – dans ce cas précis les commandes `\markup` ou `\markuplist` sont obligatoires et ne sauraient être remplacées par du simple texte borné par des guillemets informatiques ; voir Section 8.1.6 [Texte indépendant], page 328 ;
- dans toute définition au sein d'un bloc `\header` (par ex. `title`, `subtitle`, `composer`), ou dans des éléments spécifiques au bloc `\paper` tel que `evenHeaderMarkup` pour les numéros de page. Ceci est abordé dans Chapitre 21 [Titres et entêtes], page 615.

De nombreux autres objets basés sur du texte peuvent se saisir sous forme de bloc *markup* même si ce n'est pas leur usage premier.

- Les doigtés peuvent se remplacer par des blocs de *markup* dès lors qu'ils sont introduits par la commande `\finger` ; voir Section 7.1.2 [Doigtés], page 290.
- Les syllabes des paroles peuvent être formatées avec une commande `\markup` ; voir Section 9.1 [Vue d'ensemble de la musique vocale], page 355.
- Les noms des accords sont en fait définis sous forme de blocs de *markup* et peuvent donc être redéfinis de la même manière pour adapter les modificateurs d'accord ou les exceptions ; voir Section 15.2 [Gravure des accords], page 527.
- Les nuances se saisissent de la manière la plus simple. Il est néanmoins possible de définir sa Section 3.1.3 [Personnalisation des indications de nuance], page 171, sous forme d'objet *markup*. Certaines nuances, comme un crescendo, sont affichées sous forme de bandeau et peuvent être redéfinies au travers de certaines propriétés telle que `crescendoText` ; voir Section 3.1.2 [Nuances], page 162.
- Des objets moins courants sont constitués de blocs *markup*, tels les indications en Section 7.2.2 [Info-bulle], page 309.

Il est en fait possible d'utiliser `\markup` pour personnaliser l'apparence de pratiquement n'importe quel objet graphique (*grob*) en appliquant une dérogation soit à sa propriété `text` s'il en dispose ou de sa propriété `stencil`. Une partie de la logique qui rend ceci possible est expliquée dans Section "Flexibilité architecturale" dans *Essai*.

L'exemple qui suit illustre l'ubiquité des blocs *markup* non seulement comme les objets ci-dessus présentés, mais aussi en remplaçant des objets musicaux par des objets textuels de différentes manières.

```
\header { title = \markup "Header" }

dyn =
#(make-dynamic-script #{ \markup \serif "DynamicText" #})

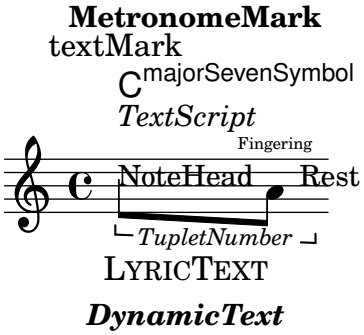
\markup \box "Top-level markup"

\score {
  <<
    \new ChordNames
    \with {
      majorSevenSymbol = \markup "majorSevenSymbol"
    }
    \chordmode { c1:maj7 }
```

```
    \new Staff {
      \tempo \markup "MetronomeMark"
      \textMark "textMark"
      \once \override TupletNumber.text =
        \markup "TupletNumber"
      \tuplet 3/2 {
        \once \override NoteHead.stencil =
          #ly:text-interface::print
        \once \override NoteHead.text =
          \markup \lower #0.5 "NoteHead"
        c''8^\markup \italic "TextScript"
        a'\finger \markup \serif "Fingering"
        \once \override Rest.stencil =
          #(lambda (grob)
             (grob-interpret-markup grob #{
               \markup  "Rest"
               #}))
        r
      }
    }
    \new Lyrics \lyricmode {
      \markup \smallCaps "LyricText" 1
    }
    \new Dynamics { s1\dyn }
  >>
}
```

Header

Top-level markup



Voir aussi

Manuel de notation : Section 8.1.2 [Commentaires textuels], page 320, Section 7.1.2 [Doigtés], page 290, Section 15.2 [Gravure des accords], page 527, Section 8.1.3 [Indication textuelle avec extension], page 321, Section 8.1.5 [Indications textuelles], page 323, Section 7.2.2 [Info-bulle], page 309, Section 8.2 [Mise en forme du texte], page 329, Section 3.1.2 [Nuances], page 162, Section 3.1.3 [Personnalisation des indications de nuance], page 171, Section 8.1.6 [Texte indépendant], page 328, Section 9.1 [Vue d'ensemble de la musique vocale], page 355.

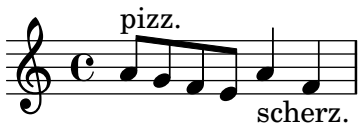
Essai sur la gravure musicale automatisée : Section "Flexibilité architecturale" dans *Essai*.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

8.1.2 Commentaires textuels
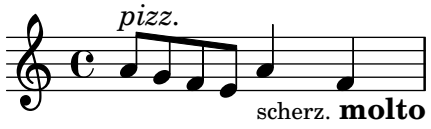
Vous pouvez ajouter à une partition des indications sous forme textuelle, comme dans l'exemple suivant. Ces indications se placeront manuellement au-dessus ou au-dessous de la portée selon la syntaxe utilisée – cf. Section 36.1 [Direction et positionnement], page 801.

```
\relative { a'8^"pizz." g f e a4-"scherz." f }
```
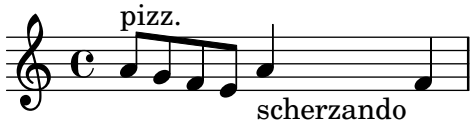


Cette syntaxe est en fait un raccourci. Des constructions plus élaborées d'annotation peuvent être obtenues en ayant recours à un bloc `\markup` et selon les préceptes énoncés dans Section 8.2 [Mise en forme du texte], page 329.

```
\relative {
  a'8^\markup { \italic pizz. } g f e
  a4_\markup { \tiny scherz. \bold molto } f }
```



Par défaut, ces indications n'affectent en rien l'espacement des notes. Leur longueur peut néanmoins être prise en considération : dans l'exemple qui suit, le premier commentaire n'influe pas sur l'espacement, à l'inverse du second.

```
\relative {
  a'8^"pizz." g f e
  \textLengthOn
  a4_"scherzando" f
}
```



En plus d'indications textuelles, les notes peuvent se voir attacher des articulations, comme indiqué au chapitre Section 3.1.1 [Articulations et ornements], page 158.

Pour de plus amples détails sur la manière de combiner indications textuelles et articulations, reportez-vous au chapitre Section "Positionnement des objets" dans *Manuel d'initiation*.

Commandes prédéfinies

`\textLengthOn`, `\textLengthOff`.

Voir aussi

Manuel d'initiation : Section "Positionnement des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Section 36.1 [Direction et positionnement], page 801, Section 8.2 [Mise en forme du texte], page 329.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

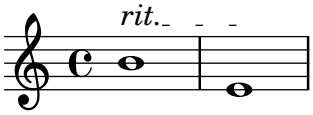
S'assurer que tous les éléments textuels et les paroles respectent les marges du document requiert des calculs relativement lourds ; vous pouvez toutefois vous en affranchir en ajoutant

```
\override Score.PaperColumn.keep-inside-line = ##f
```

8.1.3 Indication textuelle avec extension

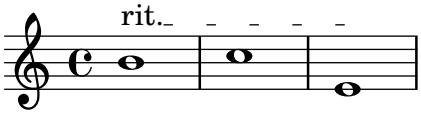
Certaines indications d'interprétation comme *rallentando*, *accelerando* ou *trilles*, s'inscrivent textuellement et se prolongent sur plusieurs notes à l'aide d'une ligne pleine, pointillée ou ondulée. Ces objets, que l'on appelle « extenseurs » (*spanners* en anglais), se dessinent entre deux notes à l'aide de la syntaxe suivante :

```
\relative {
  \override TextSpanner.bound-details.left.text = "rit."
  b'1\startTextSpan
  e,\stopTextSpan
}
```



Le texte à imprimer est spécifié en tant que propriété de l'objet `TextSpanner`. Il apparaîtra par défaut en italique ; cependant, rien ne s'oppose à un autre graphisme dès lors que vous faites appel à un bloc `\markup` – voir Section 8.2 [Mise en forme du texte], page 329.

```
\relative {
  \override TextSpanner.bound-details.left.text =
    \markup { \upright "rit." }
  b'1\startTextSpan c
  e,\stopTextSpan
}
```



Le style de ligne se définit lui aussi comme une propriété de l'objet. Les détails concernant la syntaxe à utiliser sont expliqués au chapitre Section 36.5 [Styles de ligne], page 809.

Commandes prédéfinies

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`, `\startTextSpan`, `\stopTextSpan`.
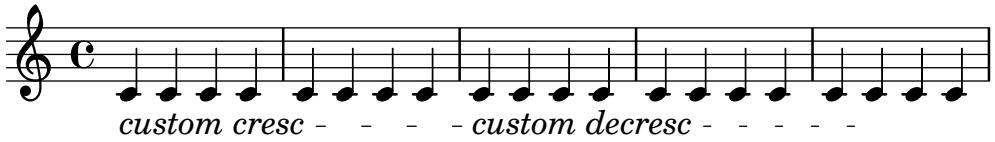
Morceaux choisis

Personnalisation des extenseurs de nuance textuelle

Il s'agit de fonctions postfix pour personnaliser l'extension des crescendos textuels. L'extension devrait débuter sur la première note de la mesure. Il faut utiliser `-\mycresc` – comme une articulation – sous peine que le départ de l'extension n'apparaisse qu'à la note suivante.

```
% Two functions for (de)crescendo spanners where you can explicitly
% give the spanner text.
mycresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'CrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))
mydecresc =
#(define-music-function (mymarkup) (markup?)
   (make-music 'DecrescendoEvent
               'span-direction START
               'span-type 'text
               'span-text mymarkup))

\relative c' {
  c4-\mycresc "custom cresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4-\mydecresc "custom decresc" c4 c4 c4 |
  c4 c4 c4 c4 |
  c4 c4\! c4 c4
}
```



Voir aussi

Manuel de notation : Section 8.2 [Mise en forme du texte], page 329, Section 3.1.2 [Nuances], page 162, Section 36.5 [Styles de ligne], page 809.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

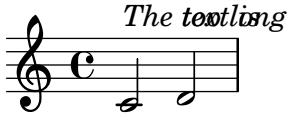
Référence des propriétés internes : Section “TextSpanner” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

LilyPond ne peut traiter qu’un seul extenseur à la fois par voix.

Les textes aux extrémités d’un bandeau peuvent être amenés à se chevaucher.

```
{
  \once \override TextSpanner.bound-details.left.text = "The text is"
  \once \override TextSpanner.bound-details.right.text = "too long"
  c'2\startTextSpan d'2\stopTextSpan
}
```

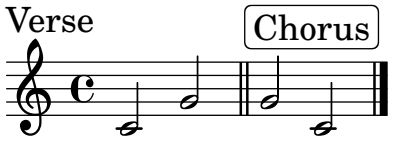


Un moyen de contournement est disponible à la rubrique Chapitre 30 [Espacement horizontal], page 736.

8.1.4 Marque de section

La commande `\sectionLabel` indique le début d'un passage nommé. Elle est tout à fait appropriée dans le cadre d'une division en sections créées par `\section` et peut s'utiliser de manière indépendante.

```
\fixed c' {
  \sectionLabel "Verse"
  c2 g
  \section
  \sectionLabel \markup { \rounded-box { Chorus } }
  g2 c
  \bar "|."
}
```



Voir aussi

Manuel de notation : Section 2.5.7 [Division en sections], page 150, Section 2.5.5 [Indications de repère], page 144, Section 8.2 [Mise en forme du texte], page 329.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SectionLabel" dans *Référence des propriétés internes*, Section "SectionLabelEvent" dans *Référence des propriétés internes*.

8.1.5 Indications textuelles
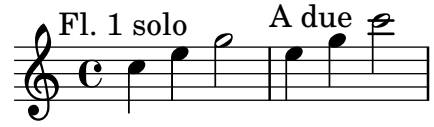
Les indications textuelles sont des objets qui, contrairement aux scripts (voir Section 8.1.2 [Commentaires textuels], page 320), ne s'impriment pas en regard des notes, mais entre elles, et souvent alignées sur une barre de mesure.

> **Note :** Les versions précédentes de LilyPond utilisaient l'instruction `\mark` pour les indications textuelles, bien quelle ait été initialement conçue pour les repères de répétition (voir Section 2.5.5 [Indications de repère], page 144). Les commandes `\textMark` et `\textEndMark` sont plus appropriées en matière d'indications textuelles de par leurs réglages par défaut et la capacité qu'elles offrent de présenter plusieurs indications en un même endroit. Il est donc préférable d'utiliser `\textMark` et `\textEndMark` en lieu et place de `\mark "Texte"` ou `\mark \markup ...`. Pour la conversion de code utilisant `\mark`, les dérogations portées à `RehearsalMark` devraient être modifiées en `TextMark`.

Avant d'insérer une indication textuelle, il peut être judicieux de chercher une commande qui serait plus appropriée. Lorsque le texte identifie une section, utilisez `\sectionLabel` (voir Section 8.1.4 [Marque de section], page 323). Pour un saut ou un renvoi arbitraire, il vaut mieux utiliser `\jump` (voir Section 4.1.8 [Indications de reprise manuelles], page 210). Ces instructions disposent de mises en forme par défaut différentes et créent d'autres objets qui peuvent être personnalisés indépendamment des indications textuelles génériques à l'aide de feuilles de style.
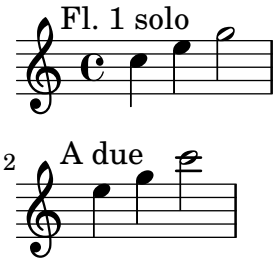
Une indication textuelle se saisit à l'aide de `\textMark` ou `\textEndMark`. La commande `\textMark` affiche une marque alignée sur sa gauche.

```
\fixed c'' {
  \textMark "Fl. 1 solo"
  c4 e g2
  \textMark "A due"
  e4 g c'2
}
```
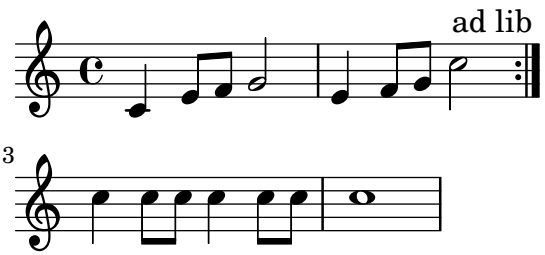


Lorsqu'intervient un saut de ligne à l'endroit où est placé `\textMark`, l'indication apparaîtra sur le système suivant.

```
\fixed c'' {
  \textMark "Fl. 1 solo"
  c4 e g2
  \break
  \textMark "A due"
  e4 g c'2
}
```
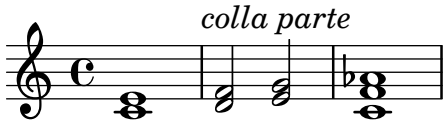


Contrairement à `\textMark`, `\textEndMark` crée une marque alignée sur sa droite. Si elle se présente lors d'un saut de ligne, l'indication apparaîtra en fin du système précédent.

```
\fixed c' {
  \repeat volta 2 {
    c4 e8 f g2
    e4 f8 g c'2
    \textEndMark "ad lib"
  }
  \break
  c'4 8 8 4 8 8
  c'1
}
```



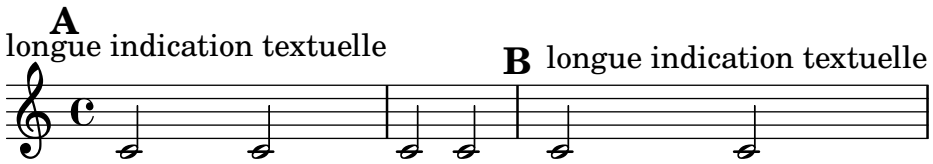
Le texte peut être mis en forme de différentes manières dès lors qu'est utilisé un bloc `\markup`, comme indiqué à la rubrique Section 8.2 [Mise en forme du texte], page 329.

```
\relative {
  <c' e>1
  \textMark \markup { \italic { colla parte } }
  <d f>2 <e g>
  <c f aes>1
}
```
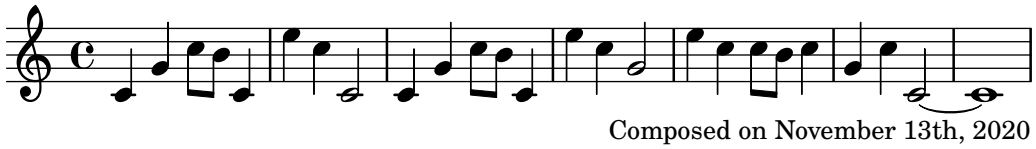


Les commandes \markLengthOn et \markLengthOff – voir Section 2.3.3 [Indication métronomique], page 88, – sont opérationnelles dans le cadre des indications textuelles.

```
{
  \mark \default
  c'2 2
  \textEndMark "longue indication textuelle"
  \markLengthOn
  2 2
  \mark \default
  2 2
  \textEndMark "longue indication textuelle"
}
```
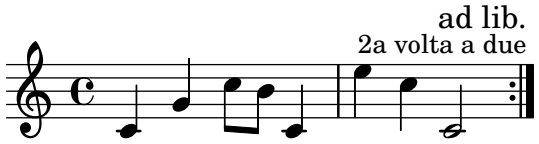


Les indications textuelles peuvent se présenter en dessous de la portée.

```
\fixed c' {
  c4 g c'8 b c4 e' c' c2
  c4 g c'8 b c4 e' c' g2
  e'4 c' c'8 b c'4
  g4 c' c2~ c1
  \tweak direction #DOWN
    \tweak font-size -1
      \textEndMark "Composed on November 13th, 2020"
}
```



Plusieurs indications peuvent intervenir en un même moment. L'ordre dans lequel elles s'empileront peut se gérer à l'aide de la propriété outside-staff-priority (voir Section 29.3 [Résolution des collisions verticales], page 733). Une autre façon de procéder pourrait consister à utiliser une seule indication sous la forme \markup \column { ... } .

```
\fixed c' {
  \repeat volta 2 {
    c4 g c'8 b c4 e' c' c2
    \textEndMark "ad lib."
    \tweak outside-staff-priority 1200
      \tweak font-size -1
        \textEndMark "2a volta a due"
  }
}
```
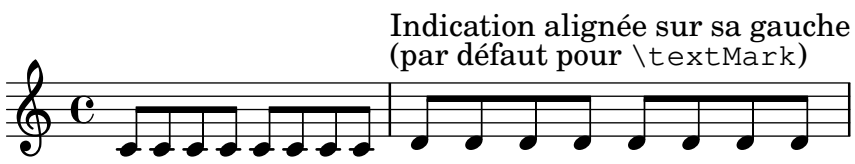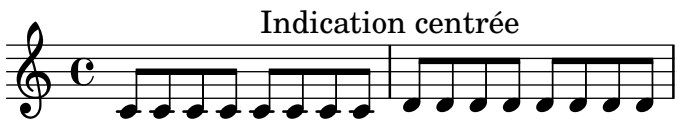


L'alignement d'une indication textuelle peut se modifier.
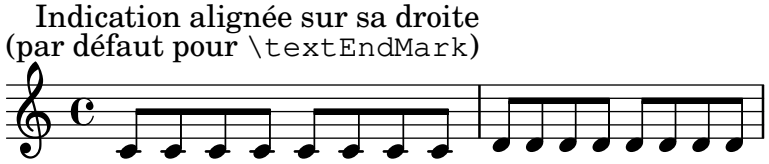
```
{
  c'8 8 8 8 8 8 8 8
  \textMark \markup \column {
    \line { Indication alignée sur sa gauche }
    \line { (par défaut pour \concat { \typewriter "\\textMark" ) } }
  }
  d'8 8 8 8 8 8 8 8
}

{
  c'8 8 8 8 8 8 8 8
  \tweak self-alignment-X #CENTER
    \textMark "Indication centrée"
  d'8 8 8 8 8 8 8 8
}

{
  c'8 8 8 8 8 8 8 8
  \tweak self-alignment-X #RIGHT
    \textMark \markup \right-column {
      \line { Indication alignée sur sa droite }
      \line { (par défaut pour \concat { \typewriter "\\textEndMark" ) } }
    }
  d'8 8 8 8 8 8 8 8
}
```







Commandes prédéfinies

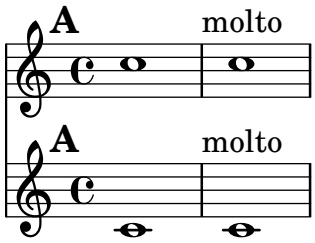
`\textMark`, `\textEndMark`, `\markLengthOn`, `\markLengthOff`.

Morceaux choisis

Impression des indications sur toutes les portées d'un système

Bien que ces indications textuelles et repères ne soient habituellement imprimées qu'au niveau de la portée supérieure d'un système, leur affectation peut être répercutée à chacune des portées.

```
\score {
  <<
    \new Staff { \mark \default c''1 \textMark "molto" c'' }
    \new Staff { \mark \default c'1 \textMark "molto" c' }
  >>
  \layout {
    \context {
      \Score
      \remove Mark_engraver
      \remove Text_mark_engraver
      \remove Staff_collecting_engraver
    }
    \context {
      \Staff
      \consists Mark_engraver
      \consists Text_mark_engraver
      \consists Staff_collecting_engraver
    }
  }
}
```



Voir aussi

Manuel de notation : Section 2.5.5 [Indications de repère], page 144, Section 2.3.3 [Indication métronomique], page 88, Section B.8 [La fonte Emmentaler], page 935, Section 8.1.4 [Marque de section], page 323, Section 8.2 [Mise en forme du texte], page 329, Section 8.2.5 [Notation musicale dans du texte formaté], page 345.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Text_mark_engraver" dans *Référence des propriétés internes*, Section "TextMark" dans *Référence des propriétés internes*, Section "TextMarkEvent" dans *Référence des propriétés internes*.

8.1.6 Texte indépendant

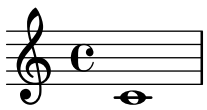
Un bloc `\markup` ou `\markuplist` peut exister en lui-même, indépendamment de tout bloc `\score`, et venir en préambule par exemple – voir le chapitre Section 20.5 [Structure de fichier], page 612, à ce propos.

```
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
```

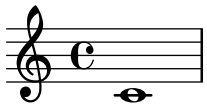
Tomorrow, and tomorrow, and tomorrow...

De cette manière, vous pouvez insérer du texte en dehors de la musique. Ceci devient particulièrement utile lorsque le fichier source contient plusieurs morceaux. Pour plus d'informations à ce propos, reportez-vous au chapitre Section 20.2 [Plusieurs partitions dans un même ouvrage], page 609.

```
\score {
  c'1
}
\markup {
  Tomorrow, and tomorrow, and tomorrow...
}
\score {
  c'1
}
```



Tomorrow, and tomorrow, and tomorrow...



Les blocs de textes peuvent s'étendre sur plusieurs pages à l'aide de la commande `\markuplist`. Ceci permet de générer des ouvrages complets uniquement grâce à LilyPond. Cette fonctionnalité, ainsi que la syntaxe appropriée, est abordée plus en détail au chapitre Section 8.2.1 [Introduction au formatage de texte], page 329.

Commandes prédéfinies

`\markup`, `\markuplist`.

Morceaux choisis

Bloc de texte indépendant sur deux colonnes

L'utilisation de la commande `\markup` permet de distribuer un bloc de texte indépendant sur plusieurs colonnes.

```
\markup {
  \fill-line {
    \hspace #1
    \column {
      \line { O sacrum convivium }
      \line { in quo Christus sumitur, }
      \line { recolitur memoria passionis ejus, }
      \line { mens impletur gratia, }
      \line { futurae gloriae nobis pignus datur. }
      \line { Amen. }
    }
    \hspace #2
    \column \italic {
      \line { O sacred feast }
      \line { in which Christ is received, }
      \line { the memory of His Passion is renewed, }
      \line { the mind is filled with grace, }
      \line { and a pledge of future glory is given to us. }
      \line { Amen. }
    }
    \hspace #1
  }
}
```

O sacrum convivium
in quo Christus sumitur,
recolitur memoria passionis ejus,
mens impletur gratia,
futurae gloriae nobis pignus datur.
Amen.

O sacred feast
in which Christ is received,
the memory of His Passion is renewed,
the mind is filled with grace,
and a pledge of future glory is given to us.
Amen.

Voir aussi

Manuel de notation : Section 8.2 [Mise en forme du texte], page 329, Section 20.2 [Plusieurs partitions dans un même ouvrage], page 609, Section 20.5 [Structure de fichier], page 612.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

8.2 Mise en forme du texte

Nous allons voir dans les lignes qui suivent la manière de mettre en forme du texte à l’aide de la syntaxe propre au mode *markup*.

8.2.1 Introduction au formatage de texte

Un bloc `\markup` ou `\markuplist` permet d’ajouter du texte et dispose de sa propre syntaxe que nous appellerons le « mode *markup* ». De tels blocs peuvent s’utiliser partout, comme indiqué dans Section 8.1.1 [Vue d’ensemble des objets textuels], page 317.

En mode *markup*, les mots s’écrivent tels quels. Un mot isolé ne requiert pas de guillemets.

```
\markup intenso
```

intenso

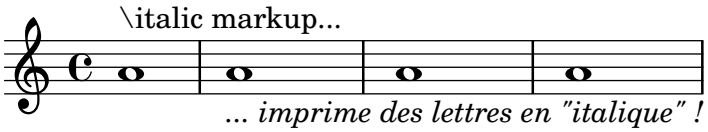
Une suite de mots peut se regrouper en les bornant par des guillemets informatiques ".

```
\markup "molto intenso"
```

molto intenso

Au delà du simple fait de grouper des mots, les guillemets informatiques permettent d'imprimer littéralement des caractères spéciaux – tels qu'un '\' ou un '#' – sans influer sur le formatage du texte. Il est de ce fait possible d'imprimer des guillemets informatiques " dès lors qu'ils sont précédés d'une oblique inverse.

```
\relative {
  a'1^"\\italic markup..."
  a_\markup { \italic "... imprime des lettres en \"italique\" !" }
  a a
}
```



La mise en forme s'obtient à l'aide de commandes dédiées dont le nom est précédé d'une oblique inverse. Elles prennent un certain nombre d'arguments spécifiques à la commande. Une liste exhaustive des commandes spécifiques au mode *markup* est fournie dans l'annexe Section A.1 [Commandes pour *markup*], page 837, (en anglais).

```
\markup \italic "string. assai"
\markup \with-color "red" intenso
```

string. assai

intenso

Les commandes de *markup* peuvent être imbriquées. Un bloc *markup* sera clôturé dès lors que toutes les commandes qu'il contient auront reçu leurs arguments.

```
\markup \with-color "red" \italic intenso
```

intenso

Plusieurs expressions *markup* peuvent se regrouper par des accolades pour former une *liste de markups*. En l'absence de formatage particulier, les éléments d'une liste de *markups* sont affichés en enfilade.

```
\markup { molto \italic intenso }
```

molto *intenso*

Certaines commandes n'attendent pas un simple *markup*, mais une liste de *markups*, ce qui permet des présentations plus élaborées qu'une simple enfilade.

```
\markup \center-column {
  \bold "Des Simplicius Simplicissimus Jugend"
  "Karl Amadeus Hartmann"
}
```

Des Simplicius Simplicissimus Jugend
Karl Amadeus Hartmann

De plus, certaines commandes ne renvoient pas un *markup*, mais une liste de *markups*. Leur résultat peut alors s'utiliser là où on attendait une liste de *markup*. Une liste de ces commandes est fournie à l'annexe Section A.2 [Commandes pour liste de *markups*], page 914, (en anglais).

```
\markup \string-lines
    "Twinkle, twinkle, little star,
     How I wonder what you are!"
```

Twinkle, twinkle, little star, How I wonder what you are!

```
\markup \center-column \string-lines
    "Twinkle, twinkle, little star,
     How I wonder what you are!"
```

Twinkle, twinkle, little star,
How I wonder what you are!

Les éléments d'une liste composite de *markups* sont tout bonnement traités comme des éléments de la liste principale.

```
\markup \center-column {
  \bold "Des Simplicius Simplicissimus Jugend"
  { Karl Amadeus \smallCaps Hartmann }
}
```

Des Simplicius Simplicissimus Jugend
Karl
Amadeus
HARTMANN

Plusieurs éléments d'une liste de *markups* seront mis en rang sur une même ligne s'ils sont regroupés à l'aide de la commande `\line`.

```
\markup \center-column {
  \bold "Des Simplicius Simplicissimus Jugend"
  \line { Karl Amadeus \smallCaps Hartmann }
}
```

Des Simplicius Simplicissimus Jugend
Karl Amadeus HARTMANN

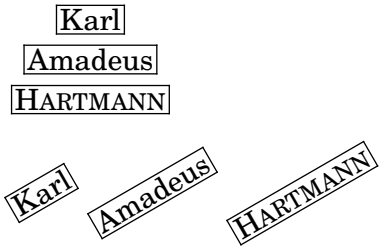
Le comportement des commandes prenant en argument des *markups* mais s'appliquent à des listes de *markups* constitue une fonctionnalité spéciale. Lorsque leur dernier argument est une liste et non un *markup*, chacun des éléments de cette liste se voit appliquer le traitement de la commande.

```
\markup \box { Karl Amadeus \smallCaps Hartmann }
```

Karl Amadeus HARTMANN

Dans ce cas, le résultat est par contre une liste de *markup* qui peut donc être passée à une commande attendant une liste de *markups* ou une commande attendant un *markup*, avec à nouveau le comportement que nous venons de voir.

```
\markup \center-column \box { Karl Amadeus \smallCaps Hartmann }
\markup \rotate #30 \box { Karl Amadeus \smallCaps Hartmann }
```



L'application de la commande `\line` à une liste de *markups* permet qu'elle soit traitée comme un argument *markup* unique.

```
\markup \box { Karl Amadeus \smallCaps Hartmann }
\markup \box \line { Karl Amadeus \smallCaps Hartmann }
```

Karl Amadeus HARTMANN

Karl Amadeus HARTMANN

Lorsque l'intégralité du contenu d'une expression `\markup` est une liste de *markups*, elle est implicitement reproduite avec la commande `\line`. Ses éléments sont donc empilés horizontalement et groupés dans un seul bloc de texte insécable. La commande `\markuplist` agit différemment : elle attend une liste de *markups* et affiche individuellement les éléments *markups* sur la page, les empilant verticalement, autorisant par ailleurs les sauts de page. L'exemple ci-dessous illustre cette différence.

```
\markup \box \wordwrap {
  Lorem ipsum dolor sit amet, consectetur
  adipisicing elit, sed do eiusmod tempor incididunt
  ut labore et dolore magna aliqua.  Ut enim ad minim
  veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat.
}

\markuplist \box \wordwrap-lines {
  Lorem ipsum dolor sit amet, consectetur
  adipisicing elit, sed do eiusmod tempor incididunt
  ut labore et dolore magna aliqua.  Ut enim ad minim
  veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat.
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
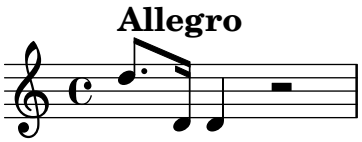
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

commodo consequat.

Vous pouvez stocker les étiquettes textuelles en tant que variables, et attacher ces identifiants à des notes, comme ici :

```
allegro = \markup \bold \large Allegro

{
  d''8.^\allegro
  d'16 d'4 r2
}
```



La syntaxe de la commande \etc permet de définir des raccourcis personnalisés utilisables dans le cadre des commandes \markup.

```
\markup reddish = \markup \with-color "tomato" \etc

\markup { molto \reddish intenso }
```

molto intenso

Le fonctionnement interne de ces commandes et la façon d'en implémenter de nouveaux est abordée dans Section “Fonctions pour markups” dans *Extension de LilyPond*.

Voir aussi

Manuel de notation : Section A.2 [Commandes pour liste de *markups*], page 914, Section A.1 [Commandes pour *markup*], page 837, Section 8.1.1 [Vue d'ensemble des objets textuels], page 317.

Manuel d'extension des fonctionnalités : Section “Fonctions pour markups” dans *Extension de LilyPond*

Fichiers d'initialisation : `scm/define-markup-commands.scm`, `scm/markup.scm`.

Morceaux choisis : Section “Texte” dans *Morceaux choisis*.

Référence des propriétés internes : Section “TextScript” dans *Référence des propriétés internes*.

Problèmes connus et avertissements
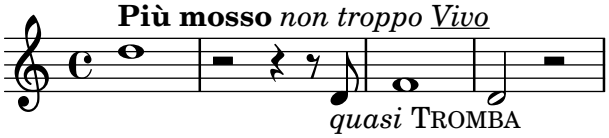
Les messages d'erreur de syntaxe en mode *markup* sont peu explicites.

8.2.2 Sélection de la fonte et de la taille

Le mode *markup* autorise des changements élémentaires de la fonte :

```
\relative {
  d''1^\markup {
    \bold { Più mosso }
    \italic { non troppo \underline Vivo }
  }
  r2 r4 r8
  d,_\markup { \italic quasi \smallCaps Tromba }
  f1 d2 r
```

```
}
```

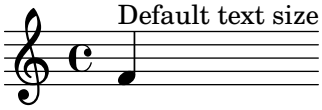


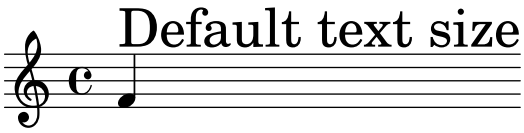
La taille globale du texte apparaissant dans les *markups* peut se régler à l'aide de la variable de papier `text-font-size`. Ceci permet d'ajuster une fonte principale qui pourrait paraître plus grande ou plus petite bien qu'ayant la même taille nominale. La valeur est exprimée en points (sans mention de l'unité) : la valeur par défaut dépend de la taille de portée et résulte du calcul `(staff-height / 20 * 11)`.

Voir Section 36.2 [Distances et unités de mesure], page 803, pour de plus amples informations quant aux dimensions utilisées par LilyPond.

```
\score {
  { f'^"Default text size" }
  \layout { text-font-size = 10 }
}

\score {
  { f'^"Default text size" }
  \layout { text-font-size = 20 }
}
```

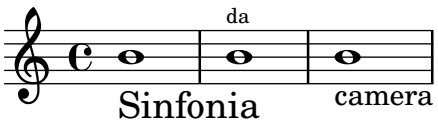




La taille des caractères se modifie, relativement à la taille globale des portées, de différentes manières.

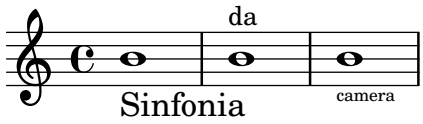
Vous pouvez adopter l'une des tailles prédéfinies, comme ici :

```
\relative b' {
  b1_\markup { \huge Sinfonia }
  b1^\markup { \teeny da }
  b1-\markup { \normalsize camera }
}
```



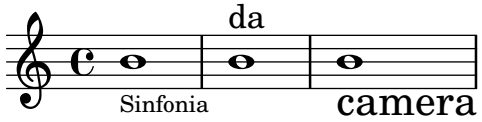
Vous pouvez la modifier relativement à sa valeur précédente :

```
\relative b' {
  b1_\markup { \larger Sinfonia }
  b1^\markup { \smaller da }
  b1-\markup { \magnify #0.6 camera }
}
```



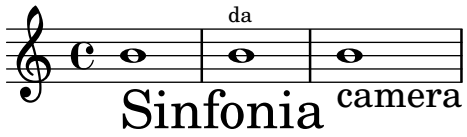
Vous pouvez l'augmenter ou la diminuer par rapport à la taille globale de portée :

```
\relative b' {
  b1_\markup { \fontsize #-2 Sinfonia }
  b1^\markup { \fontsize #1 da }
  b1-\markup { \fontsize #3 camera }
}
```



Vous pouvez lui attribuer une valeur arbitraire quelle que soit la taille de portée globale :

```
\relative b' {
  b1_\markup { \abs-fontsize #20 Sinfonia }
  b1^\markup { \abs-fontsize #8 da }
  b1-\markup { \abs-fontsize #14 camera }
}
```



Lorsque le texte comporte des espaces, mieux vaut le borner par des guillemets informatiques ; s'en suivra une meilleure adéquation entre la taille des espaces et celle des autres caractères :

```
\markup \fontsize #6 \bold { Sinfonia da camera }
\markup \fontsize #6 \bold { "Sinfonia da camera" }
```

Sinfonia da camera
Sinfonia da camera

Vous pouvez imprimer du texte en indice ou en exposant. Celui-ci sera dans une taille plus petite, mais rien ne s'oppose à ce que vous lui affectiez une taille normale :

```
\markup {
  \column {
    \line { 1 \super st movement }
    \line { 1 \normal-size-super st movement
      \sub { (part two) }  }
  }
}
```

1 st movement
1 st movement (part two)

Le mode *markup* vous permet de sélectionner d'autres familles de fontes. Par défaut, LilyPond utilise une police avec empattement, et tout changement doit être explicite. Dans la dernière ligne de l'exemple qui suit, vous noterez qu'il n'y a aucune différence entre les premier et deuxième mots.

```
\markup {
  \column {
    \line { Act \number 1 }
    \line { \sans { Scene I. } }
    \line { \typewriter { Verona. An open place. } }
    \line { Enter \serif Valentine and Proteus. }
  }
}
```

Act 1
Scene I.
Verona. An open place.
Enter Valentine and Proteus.

Certaines familles de police spécifiques aux nombres ou aux nuances par exemple, ne disposent pas de tous les caractères, comme nous l'avons vu dans les chapitres Section 3.1.3 [Personnalisation des indications de nuance], page 171, et Section 4.1.8 [Indications de reprise manuelles], page 210.

Lorsqu'un changement survient au milieu d'un mot, il se peut qu'un espacement supplémentaire apparaisse. Il suffit en pareil cas de concaténer les différents éléments :

```
\markup {
  \column {
    \line {
      \concat { 1 \super st }
      movement
    }
    \line {
      \concat { \dynamic p , }
      \italic { con dolce espressione }
    }
  }
}
```

1st movement
p, *con dolce espressione*

Une liste des différentes commandes permettant de changer de fonte ou d'utiliser des fontes personnalisées est disponible à l'annexe Section A.1.1 [Font markup], page 837.

Pour savoir comment personnaliser des fontes, reportez-vous au chapitre Section 8.3 [Fontes], page 347.

Commandes prédéfinies

`\teeny`, `\tiny`, `\small`, `\normalsize`, `\large`, `\huge`, `\smaller`, `\larger`.

Voir aussi

Manuel de notation : Section A.1.1 [Font markup], page 837, Section 8.3 [Fontes], page 347, Section 4.1.8 [Indications de reprise manuelles], page 210, Section 3.1.3 [Personnalisation des indications de nuance], page 171.

Fichiers d'initialisation : `scm/define-markup-commands.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Problèmes connus et avertissements
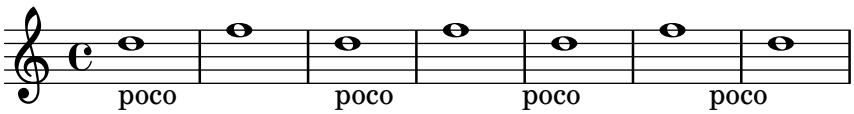
Le recours aux commandes `\teeny`, `\tiny`, `\small`, `\normalsize`, `\large` et `\huge` produiront des espacements nettement moins réguliers que si vous utilisez `\fontsize`.

8.2.3 Alignement du texte

Cette partie traite de la manière de positionner du texte en mode *markup*. On déplace l'intégralité d'un objet *markup* en utilisant la syntaxe décrite au chapitre Section "Déplacement d'objets" dans *Manuel d'initiation*.

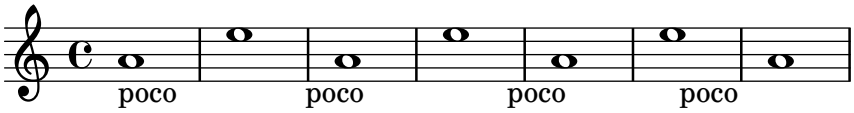
Les objets de type *markup* peuvent s'aligner de différentes manières. Une indication textuelle est par défaut alignée sur son extrémité gauche. Dans l'exemple qui suit, il n'y a aucune différence entre les deux premiers *markups*. Cet exemple indique aussi différentes syntaxes pour les commandes d'alignement.

```
\relative {
  d''1-\markup { poco }
  f
  d-\markup { \left-align poco }
  f
  d-\markup { \center-align { poco } }
  f
  d-\markup \right-align { poco }
}
```



L'alignement horizontal peut être ajusté à l'aide d'une valeur numérique :

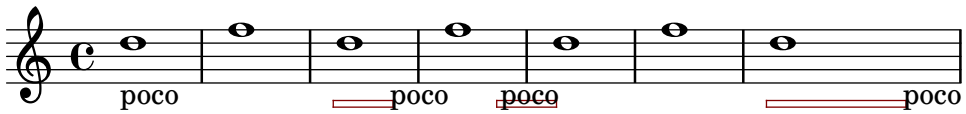
```
\relative {
  a'1-\markup { \halign #-1 poco }
  e'
  a,-\markup { \halign #0 poco }
  e'
  a,-\markup { \halign #0.5 poco }
  e'
  a,-\markup { \halign #2 poco }
}
```



Enfin, les mots et tout autre objet peuvent se déplacer horizontalement en les faisant précéder d'un décalage. Un décalage négatif est possible, ce qui aura pour effet de déplacer les objets

qui le suivent dans la direction opposée. Bien que le décalage soit normalement invisible, des commandes spécifiques permettent de le mettre en évidence comme dans l'exemple ci-dessous :

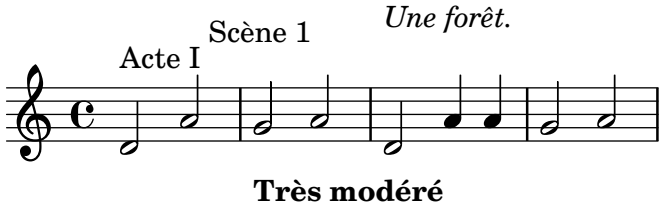
```
\relative {
  d''1-\markup { poco }
  f
  d-\markup \concat { \with-color #darkred \box \hspace #4 poco }
  f
  d-\markup \concat { \with-color #darkred \box \hspace #-4 poco }
  f
  d-\markup \concat { \with-color #darkred \box \hspace #10 poco }
}
```



Certains objets possèdent leurs propres procédures d'alignement, qui annuleront toute spécification d'alignement que vous pourriez leur affecter. La solution consiste alors à déplacer l'intégralité de ces objets *markup*, comme indiqué par exemple au chapitre Section 8.1.5 [Indications textuelles], page 323.

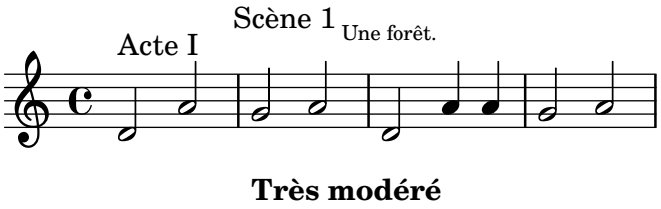
L'alignement vertical est quant à lui un peu plus compliqué. Comme nous l'avons vu ci-avant, les objets *markup* peuvent être déplacés dans leur intégralité. Il est néanmoins possible de déplacer certains éléments spécifiques au sein d'un bloc *markup*. En pareil cas, l'élément à déplacer doit être précédé d'un « point d'ancrage » – un autre élément du *markup* ou un objet invisible. L'exemple qui suit illustre ces deux possibilités. Vous noterez par ailleurs que le dernier *markup*, dépourvu de point d'ancrage, n'est de ce fait pas déplacé.

```
\relative {
  d'2^\markup {
    Acte I
    \raise #2 { Scène 1 }
  }
  a'
  g_\markup {
    \lower #4 \bold { Très modéré }
  }
  a
  d,^\markup \raise #4 \italic {
    Une forêt.
    }
  a'4 a g2 a
}
```
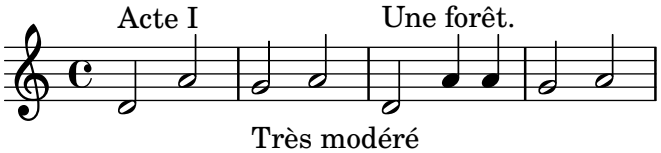


Certaines commandes permettent de régler l'alignement des objets textuels en mode *markup*, tant au niveau horizontal que vertical.

```
\relative {
  d'2^\markup {
    Acte I
    \translate #'(2 . 2) "Scène 1"
  }
  a'
  g_\markup {
    \general-align #Y #5 \bold "Très modéré"
  }
  a
  d,^\markup \translate-scaled #'(-3 . 2) \teeny {
   "Une forêt."
  }
  a'4 a g2 a
}
```



En raison de la manière dont est implémenté le colonage, `\vspace` **ne peut pas** s'utiliser pour bouger arbitrairement vers le haut ou le bas des objets d'un *markup* `\column`. La naïveté de l'approche suivante est donc vouée à l'échec.
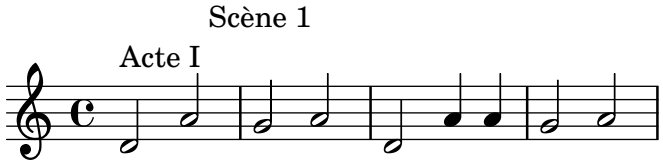
```
\relative {
  d'2^\markup {
    Acte I
    \column {
      \vspace #-2
      "Scène 1"
    }
  }
  a'
  g_\markup \column {
    \vspace #1
    "Très modéré"
  }
  a
  d,^\markup \column {
    "Une forêt."
    \vspace #2
  }
  a'4 a g2 a
}
```



Ce qui est fonctionnel consiste à insérer `\vspace` entre deux objets à l'étendue non nulle.

```
\relative {
  d'2^\markup {
    Acte I
    \column {
      " "
      \vspace #-2
      "Scène 1"
    }
  }
  a'
  g_\markup \column {
    " "
    \vspace #1
    "Très modéré"
  }
  a
  d,^\markup \column {
    "Une forêt."
    \vspace #2
    " "
  }
  a'4 a g2 a
}
```

Une forêt.



Très modéré

Un objet de type *markup* peut contenir plusieurs lignes de texte. Dans l'exemple suivant, chaque élément ou expression ira se placer sur sa propre ligne, tantôt alignée à gauche, tantôt centrée.

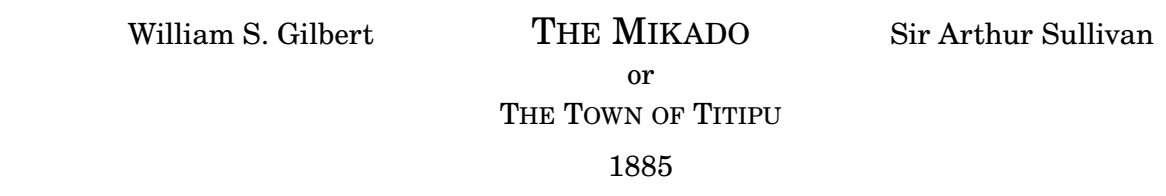
```
\markup {
  \column {
    a
    "b c"
    \line { d e f }
  }
  \hspace #10
  \center-column {
    a
    "b c"
    \line { d e f }
  }
}
```

a
b c
d e f

a
b c
d e f

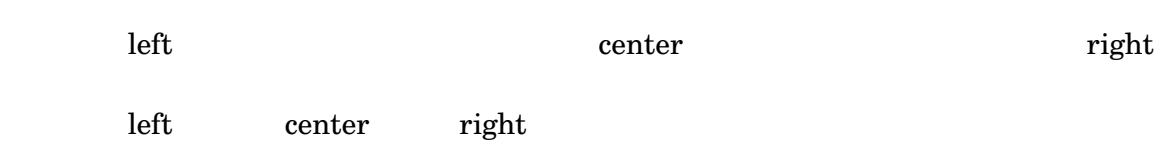
Pareillement, une liste d'éléments ou d'expressions sera répartie sur une ligne entière, voire même centrée sur toute la page s'il n'y a qu'un seul élément. De telles expressions peuvent à leur tour contenir du texte multiligne ou une autre expression *markup*.

```
\markup {
  \fill-line {
    \line { William S. Gilbert }
    \center-column {
      \huge \smallCaps "The Mikado"
      or
      \smallCaps "The Town of Titipu"
    }
    \line { Sir Arthur Sullivan }
  }
}
\markup {
  \fill-line { 1885 }
}
```



Les éléments peuvent se répartir sur une largeur spécifiée par une dérogation à la propriété `line-width`. Celle-ci est réglé par défaut à `#f`, autrement dit l'entièreté de la ligne :

```
\markup {
  \column {
    \fill-line { left center right }
    \null
    \override #'(line-width . 30)
    \fill-line { left center right }
  }
}
```



Les indications textuelles, lorsqu'elles sont relativement longues, peuvent se répartir sur plusieurs lignes en fonction de la largeur de ligne. Le texte sera alors soit aligné à gauche, soit justifié, comme le montre l'exemple suivant :

```
\markup {
  \column {
    \line  \smallCaps { La vida breve }
    \line \bold { Acto I }
    \wordwrap \italic {
      (La escena representa el corral de una casa de
      gitanos en el Albaicín de Granada. Al fondo una
      puerta por la que se ve el negro interior de
      una Fragua, iluminado por los rojos resplandores
      del fuego.)
```

```
    }
    \hspace #0

    \line \bold { Acto II }
    \override #'(line-width . 50)
    \justify \italic {
      (Calle de Granada. Fachada de la casa de Carmela
      y su hermano Manuel con grandes ventanas abiertas
      a través de las que se ve el patio
      donde se celebra una alegre fiesta)
    }
  }
}
```

La vida breve

Acto I

(La escena representa el corral de una casa de gitanos en el Albaicín de Granada. Al fondo una puerta por la que se ve el negro interior de una Fragua, iluminado por los rojos resplandores del fuego.)

Acto II

(Calle de Granada. Fachada de la casa de Carmela y su hermano Manuel con grandes ventanas abiertas a través de las que se ve el patio donde se celebra una alegre fiesta)

Une liste des différentes commandes permettant d'aligner du texte en mode *markup* est disponible à l'annexe Section A.1.2 [Markup for text alignment], page 850.

Voir aussi

Manuel d'initiation : Section "Déplacement d'objets" dans *Manuel d'initiation*.

Manuel de notation : Section A.1.2 [Markup for text alignment], page 850, Section 8.1.5 [Indications textuelles], page 323.

Fichiers d'initialisation : `scm/define-markup-commands.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

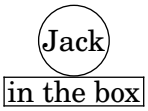
8.2.4 Éléments graphiques dans du texte formaté

Vous pouvez, grâce au mode *markup*, ajouter divers objets graphiques à votre partition.

Certaines commandes de *markup* permettent d'ornementer des éléments textuels avec des graphismes, à l'instar de l'exemple suivant :

```
\markup \fill-line {
  \center-column {
    \circle Jack
    \box "in the box"
    \null
    \line {
      Erik Satie
      \hspace #3
      \bracket "1866 - 1925"
```

```
    }
    \null
    \rounded-box \bold Prelude
  }
}
```

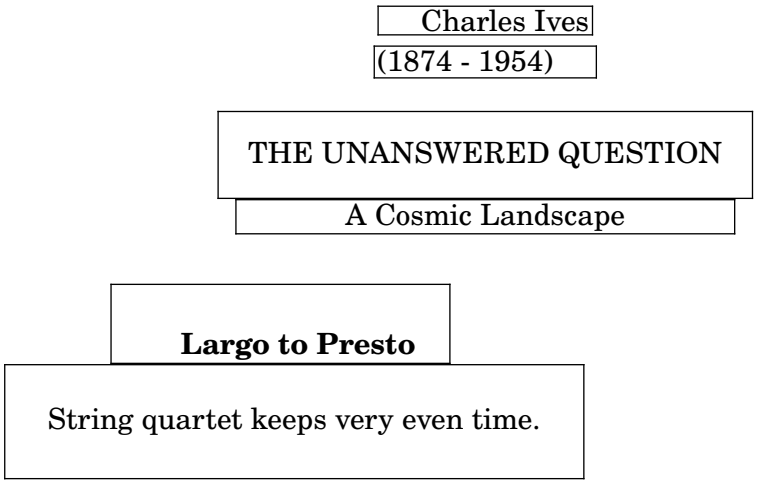


Erik Satie [1866 - 1925]

Prelude

Certaines directives peuvent nécessiter d'accroître l'espacement autour du texte – voir l'annexe Section A.1.2 [Markup for text alignment], page 850, pour une liste des différentes commandes particulières au mode *markup* ainsi que leur description.

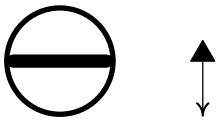
```
\markup \fill-line {
  \center-column {
    \box \pad-x-left #3 "Charles Ives"
    \box \pad-x-right #3 "(1874 - 1954)"
    \null
    \box \pad-markup #2 "THE UNANSWERED QUESTION"
    \box \pad-x #8 "A Cosmic Landscape"
    \null
  }
}
\markup \column {
  \line {
    \hspace #10
    \box \pad-to-box #'(-5 . 20) #'(0 . 5)
      \bold "Largo to Presto"
  }
  \box \pad-around #3 "String quartet keeps very even time."
}
```



Vous pouvez imprimer certains graphismes ou symboles sans qu'il n'y ait de texte. Ces objets peuvent même se combiner, à l'instar de n'importe quelle expression *markup*.

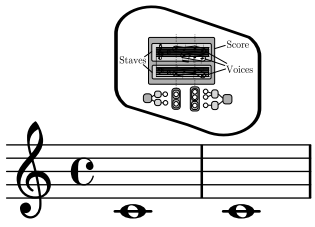
```
\markup {
  \combine
    \draw-circle #4 #0.4 ##f
    \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1
  \hspace #5

  \center-column {
    \triangle ##t
    \combine
      \draw-line #'(0 . 4)
      \arrow-head #Y #DOWN ##f
  }
}
```



Des fonctionnalités graphiques avancées vous permettent même d'inclure dans une partition un fichier image converti au format PostScript encapsulé (extension `eps`), ou bien de tracer un graphique directement dans le fichier source à partir d'instructions PostScript pures. Nous vous conseillons, en pareil cas, de toujours spécifier les dimensions du dessin, comme dans ce qui suit :

```
c'1^\markup {
  \combine
    \epsfile #X #10 "./context-example.eps"
    \with-dimensions #'(-0.5 . 10.5) #'(-2 . 8)
    \postscript "
      -0.3 1 translate
      2.7 2 scale
      newpath
      2 -1 moveto
      4 -2 4 1 1 arct
      4 2 3 3 1 arct
      0 4 0 3 1 arct
      0 0 1 -1 1 arct
      closepath
      stroke"
  }
c'
```



L'annexe Section A.1.3 [Graphical markup], page 868, répertorie les différentes commandes en matière de graphisme.

Voir aussi

Manuel de notation : Section A.1.2 [Markup for text alignment], page 850, Chapitre 7 [Annotations éditoriales], page 286, Section 36.3 [Dimensions], page 803, Section A.1.3 [Graphical markup], page 868.

Fichiers d'initialisation : `scm/define-markup-commands.scm`, `scm/stencil.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

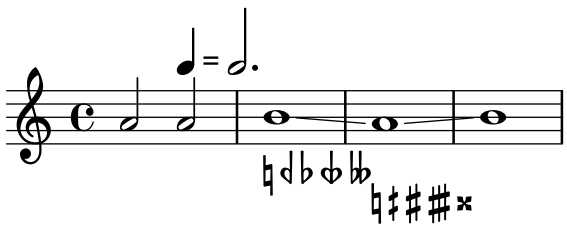
Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

8.2.5 Notation musicale dans du texte formaté

Divers éléments de notation peuvent orner une partition, au moyen d'un objet *markup*.

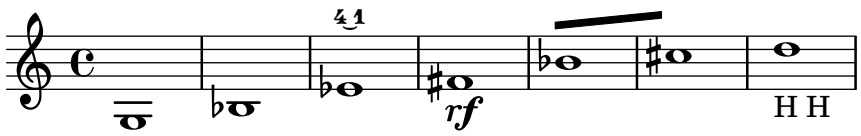
Notes et altérations sont données à l'aide d'instructions *markup* :

```
a'2 a'^\markup {
  \note {4} #1
  =
  \note-by-number #1 #1 #1.5
}
b'1_\markup {
  \natural \semiflat \flat
  \sesquiflat \doubleflat
}
\glissando
a'1_\markup {
  \natural \semisharp \sharp
  \sesquisharp \doublesharp
}
\glissando b'
```



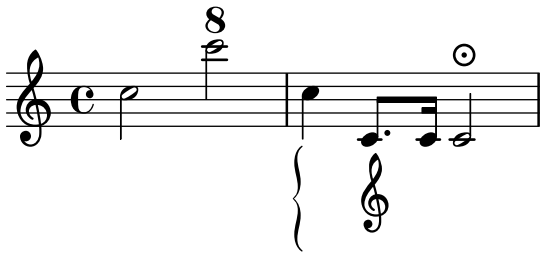
Le mode *markup* permet d'accéder à d'autres objets de notation :

```
\relative {
  g1 bes
  ees\finger \markup \tied-lyric "4~1"
  fis_\markup { \dynamic rf }
  bes^\markup {
    \beam #8 #0.1 #0.5
  }
  cis
  d-\markup {
    \markalphabet #8
    \markletter #8
  }
}
```



En règle générale, tout symbole musical peut être inclus dans un *markup*, comme le montre l'exemple qui suit. Ces différents symboles sont répertoriés dans l'annexe Section B.8 [La fonte Emmentaler], page 935.

```
\relative {
  c''2
  c'^\markup { \musicglyph "eight" }
  c,4_\markup { \left-brace #40 }
  c,8._\markup { \musicglyph "clefs.G_change" }
  c16
  c2^\markup { \musicglyph "timesig.neomensural94" }
}
```

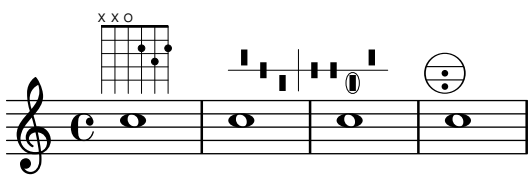


Le mode *markup* supporte aussi les diagrammes spécifiques à certains instruments :

```
\relative {
  c''1^\markup {
    \fret-diagram-terse "x;x;o;2;3;2;"
  }
  c^\markup {
    \harp-pedal "^-v|--ov^"
  }
  c
  c^\markup {
    \combine
      \musicglyph "accordion.discant"
      \combine

\raise #0.5 \musicglyph "accordion.dot"
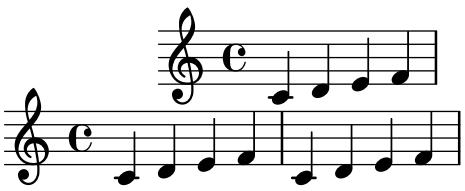
\raise #1.5 \musicglyph "accordion.dot"
  }
}
```



La documentation sur ces diagrammes se trouve à l'annexe Section A.1.6 [Instrument-specific markup], page 892.

Rien ne s'oppose à ce qu'une étiquette comporte une partition.

```
\relative {
  c'4 d^\markup {
    \score {
      \relative { c'4 d e f }
      \layout { }
    }
  }
  e f |
  c d e f
}
```



Les différentes commandes *markup* relatives à la notation musicale sont répertoriées à l'annexe Section A.1.4 [Markup for music and musical symbols], page 879.

Voir aussi

Manuel de notation : Section B.8 [La fonte Emmentaler], page 935, Section A.1.4 [Markup for music and musical symbols], page 879.

Fichiers d'initialisation : `scm/define-markup-commands.scm`, `scm/fret-diagrams.scm`, `scm/harp-pedals.scm`.

Morceaux choisis : Section "Texte" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

L'espacement vertical d'un `\score` contenu dans un objet *markup* se contrôle par la propriété `baseline-skip`. Tout réglage au sein d'un bloc `\paper` est ignoré.

8.3 Fontes

La gestion des fontes dans LilyPond est assurée par plusieurs bibliothèques : *FontConfig* (`https://fontconfig.or`) se charge de répertorier les différentes fontes disponibles ; quant à *Pango* (`https://pango.org`), elle se charge plus particulièrement de leur rendu.

Ce chapitre indique comment accéder aux fontes et polices de caractère dans LilyPond. Vous y apprendrez aussi comment changer de fonte en cours de partition.

8.3.1 Formats de fontes non pris en charge

Quelques précisions pour commencer : LilyPond crée le PDF final à partir d'un fichier PostScript intermédiaire. Par voie de conséquence, certains formats de fontes ne peuvent être utilisés parce que PostScript, en tant que standard ancien ne bénéficiant plus d'aucune mise à jour, ne les prend pas en charge. Voici une liste de formats de fonte qui sont déficients.

CFF2 Ce format de fonte est devenu ces dernières années le format par défaut des fontes OpenType (`.otf`). Bien que son prédécesseur (CFF, qui utilise aussi une extension `.otf`) soit pris en charge, le CFF2 – plus compact – ne l'est pas. LilyPond émet un avertissement en console s'il rencontre une telle fonte, et la conversion en PDF échouera ou bien certains glyphes de la fonte seront absents du PDF.

Font Variations
: De telles fontes comportent habituellement un « VF » dans leur nom ; elles ne sont malheureusement pas prises en charge, qu'elles soient en TrueType ou en OpenType, et sont donc exclues d'entrée de jeu.

fontes colorisées
: Ces fontes ne sont pas prises en charge, ce qui concerne tout particulièrement les fontes d'Emojis échelonnables. Toutefois, certaines fontes contiennent aussi des versions en noir et blanc des Emojis qui pourraient être acceptés, dès lors qu'elles sont dans un format de fonte pris en charge.

OpenType collections
: Il s'agit d'un format de type conteneur (habituellement avec une extension `.otc`) où de multiples fontes sont dans un même fichier. Il est à ce jour souvent utilisé en dernier ressort par le système d'exploitation lorsqu'aucune fonte ne convient, quelle qu'en soit la raison.

fontes bitmap
: Ce format n'est absolument pas pris en charge et est automatiquement exclu avant même que LilyPond consulte la liste des fontes disponibles. Ceci inclut certaines anciennes fontes Emoji colorisées.

De nombreuses fontes sont disponibles aussi bien au format TrueType que OpenType. Si la version `.otf` échoue, il faut essayer la version `.ttf`. Lorsqu'aucune alternative à une fonte CFF2 n'est disponible, on peut tenter l'option de LilyPond en ligne de commande `-dbackend=cairo` ; actuellement expérimental, ce moteur contourne l'étape PostScript et crée directement la sortie PDF, ce qui permet de prendre en charge CFF2 (et aussi les OpenType collections).

8.3.2 Localisation des fontes

En sus de celles déjà installées sur le système, d'autres fontes peuvent s'ajouter au répertoire de FontConfig, ce qui les rendra donc disponibles pour les partitions LilyPond, à l'aide des commandes suivantes :

```
#(ly:font-config-add-font "chemin/au/fichier-fonte")
#(ly:font-config-add-directory "chemin/au/dossier/")
```

Le chemin fourni à ces deux commandes peut être absolu ou relatif, ce qui permet de compiler une partition sur n'importe quel système dans la mesure où les fichiers de fontes concernés sont transmis avec les fichiers sources de la partition.

L'instruction `#(ly:font-config-display-fonts)` permettra de vérifier que la fonte requise est bien détectée par FontConfig. Elle affiche en console la liste exhaustive des fontes du système ainsi que le nom exact qu'il faudra transmettre à LilyPond ; ce nom peut être différent du nom du fichier. Par ailleurs, lancer `lilypond -dshow-available-fonts` dans un terminal produira le même effet.

8.3.3 Familles de fontes

Vous disposez de trois familles de fontes[1] textuelles : une police avec empattement (*sérif*), une police sans empattement (*sans* sérif) et une police à chasse fixe (monospace ou *typewriter*). En fonction du moteur de rendu utilisé, ces familles seront mappées selon des alias différents.

Pour le moteur `svg` :

Famille générique	**Famille de fonte SVG**
serif	serif

[1] Dans sa forme la plus simple, une *famille de fonte* contient habituellement les styles de police romain, italique, gras et gras italique.

sans	sans-serif
typewriter	monospace

« serif », « sans-serif » et « monospace » sont des « generic-family » au titre des spécifications SVG et CSS.

Pour les autres moteurs :

Famille générique	**Fonte par défaut (alias)**	**Listes de définition des alias**
serif	LilyPond Serif	C059, Century SchoolBook URW, Century Schoolbook L, TeX Gyre Schola, DejaVu Serif, . . . , serif
sans	LilyPond Sans Serif	Nimbus Sans, Nimbus Sans L, TeX Gyre Heros, DejaVu Sans, . . . , sans-serif
typewriter	LilyPond Monospace	Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, TeX Gyre Cursor, DejaVu Sans Mono, . . . , monospace

Lorsqu'un caractère est absent de la première fonte listée, il sera remplacé par celui de la fonte suivante.

Il est à noter que les fontes URW distribuées avec LilyPond (« C059 », « Nimbus Mono PS » et « Nimbus Sans ») disposent d'une particularité : par défaut, et en complément des ligatures standard telles que « fl » ou « ffi », elles substituent la chaîne « Nr. » par le caractère « Numero Sign » (U+2116) dès lors que le script « latn » est sélectionné. On peut toutefois s'en préserver temporairement par l'insertion d'un caractère de largeur nulle et non jointant (*zero-width non-joiner* ZWNJ, U+200C) entre les caractères « N » et « r ». Les lignes ci-dessous auront pour conséquence que LilyPond insérera toujours un caractère *ZWNJ*.
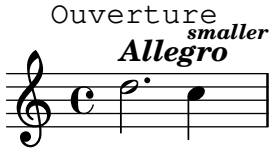
```
\paper {
  #(add-text-replacements!
      `(("Nr." . ,(format #f "N~ar." (ly:wide-char->utf-8 #x200C)))))
}
```

« LilyPond Serif », « LilyPond Sans Serif » et « LilyPond Monospace » sont des alias de fonte définis dans le fichier de configuration de FontConfig spécifique à LilyPond `00-lilypond-fonts.conf` qui se trouve normalement dans le répertoire `/usr/local/share/lilypond/2.27.4/fonts`.

La correspondance exacte entre ces trois familles de fontes et les fontes réelles peut être contrôlée à l'aide de la propriété `fonts` – voir Section 8.3.5 [Choix des fontes], page 351.

Chaque famille dispose en principe de différents styles et niveaux de graisse. L'exemple qui suit illustre la manière de changer la famille, le style, la graisse ou la taille. Notez bien que l'argument fourni à `font-size` correspond à la correction à apporter à la taille par défaut.

```
\override Score.TextMark.font-family = #'typewriter
\textMark "Ouverture"
\override Voice.TextScript.font-shape = #'italic
\override Voice.TextScript.font-series = #'bold
d''2.^\markup "Allegro"
\override Voice.TextScript.font-size = -3
c''4^smaller
```



Cette syntaxe s'applique aussi en mode *markup*, bien que celui-ci dispose d'une syntaxe allégée comme nous l'avons vu dans Section 8.2.2 [Sélection de la fonte et de la taille], page 333 :

```
\markup {
  \column {
    \line {
      \override #'((font-shape . italic) (font-size . 4))
      Idomeneo,
    }
    \line {
      \override #'(font-family . typewriter) {
        \override #'(font-series . bold) re
        di
      }
      \override #'(font-family . sans) Creta
    }
  }
}
```



Problèmes connus et avertissements

Dans de rares situations, la bibliothèque FontConfig est incapable de sélectionner une font sans ambiguïté : lorsque le nom de la famille de fonte et la graisse (autrement dit la valeur de la sous-propriété `font-series` de LilyPond) de deux différentes fontes sont identiques, elles sont considérées comme équivalentes. C'est par exemple le cas avec « Minion Pro » et « Minion Pro Display » qui, toutes deux, ont comme nom de famille « Minion Pro » et une graisse « Regular ». La seule solution en pareille circonstance est de masquer à FontConfig l'existence de la fonte non désirée.

- Si vos fontes sont stockées dans un répertoire qui n'est pas examiné par FonConfig par défaut, il faut utiliser `ly:font-config-add-font` au lieu de `ly:font-config-add-directory` pour charger explicitement la fonte désirée uniquement.
- Si les fontes sont stockées dans un répertoire examiné par défaut (par exemple `~/.fonts`), il faudra en supprimer la fonte non désirée de telle sorte que FontConfig ne découvre que la fonte requise.

8.3.4 Fonctionnalités des fontes

Le recours aux fontes OpenType permet d'utiliser certaines fonctionnalités de ces fontes.[2] Toutefois, les fontes OpenType ne disposent pas toutes de l'intégralité de ces fonctionnalités. Dans le cas où la fonctionnalité demandée n'est pas disponible dans la fonte choisie, cette fonctionnalité est tout bonnement ignorée. Les exemples ci-dessous utilisent « TeX Gyre Schola », autrement dit le style romain de la famille.

```
\paper {
  property-defaults.fonts.serif = "TeX Gyre Schola"
}

\markup "Style normal : Hello HELLO"

\markup \caps "Petites capitales : Hello"
```

[2] La sélection de scripts ou de langages des fontes OpenType n'est à ce jour par prise en charge.

```
\markup \override #'(font-features . ("smcp"))
  "Vraies petites capitales : Hello"

\markup "Style numérique normal : 0123456789"

\markup \override #'(font-features . ("onum"))
  "Style numérique ancien : 0123456789"

\markup \override #'(font-features . ("salt 0"))
  "Alternative stylistique 0 : εφπρθ"

\markup \override #'(font-features . ("salt 1"))
  "Alternative stylistique 1 : εφπρθ"

\markup \override #'(font-features . ("onum" "smcp" "salt 1"))
  "Fonctionnalités multiples : Hello 0123456789 εφπρθ"
```

Style normal : Hello HELLO

Petites capitales : Hello

Vraies petites capitales : Hello

Style numérique normal : 0123456789

Style numérique ancien : 0123456789

Alternative stylistique 0 : εφπρθ

Alternative stylistique 1 : ϵϕϖϱϑ

Fonctionnalités multiples : Hello 0123456789 ϵϕϖϱϑ

Une liste exhaustive des fonctionnalités des fontes OpenType est disponible à l'adresse `https://www.microsoft.com/typography/otspec/featurelist.htm`, et les différents types de fonctionnalités des fontes OpenType sont recensés dans le message `https://lists.gnu.org/archive/html/lilypond-devel/2017-08/msg00004.html`.

Voir aussi

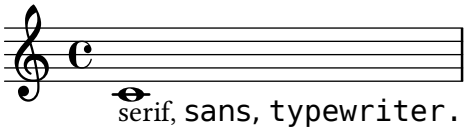
Manuel de notation : Section B.8 [La fonte Emmentaler], page 935, Section 8.2.5 [Notation musicale dans du texte formaté], page 345, Section 36.8 [Rotation des objets], page 819, Section 8.2.2 [Sélection de la fonte et de la taille], page 333, Section A.1.1 [Font markup], page 837.

8.3.5 Choix des fontes

On peut tout à fait modifier le jeu de polices par défaut de LilyPond.

```
\paper  {
  property-defaults.fonts.serif = "Linux Libertine O"
  property-defaults.fonts.sans = "DejaVu Sans"
  property-defaults.fonts.typewriter = "DejaVu Sans Mono"
}

\relative c'{
  c1-\markup {
    serif,
    \sans sans,
    \typewriter typewriter. }
}
```

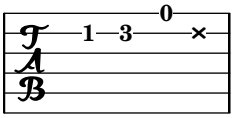


Il est aussi possible de substituer les fontes musicales à l'aide de la même syntaxe – voir Section 23.5 [Changement des fontes musicales], page 669.

Il est possible de changer de fonte pour un objet particulier ou une partie d'un *markup* à l'aide d'une dérogation à la propriété `fonts`. Dans l'exemple suivant, la police des « têtes de note » normales de la tablature est changée – c'est normalement une famille `serif`– alors que celles dessinées en forme de croix conservent la police par défaut (de la famille `music`).

```
\layout {
  \override TabVoice.TabNoteHead.property-defaults.fonts.serif =
    "Linux Libertine O"
}

\new TabStaff { c' d' e' \deadNote c' }
```



Et voici comment opérer sur cette propriété `fonts` dans le cadre d'un *markup* :

```
\markup \override #'(fonts . ((serif . "Linux Libertine O")
                              (typewriter . "DejaVu Sans Mono"))) {
  Copyright © John Doe \typewriter john@doe.org
}
```

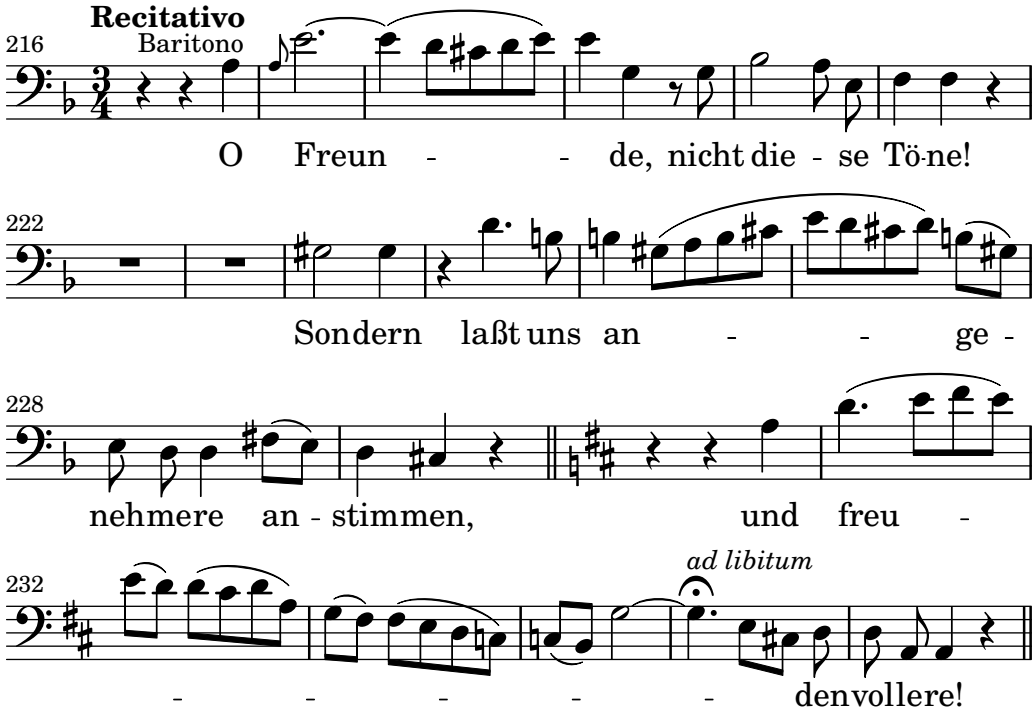
Copyright ' John Doe john@doe.org

Voir aussi

Manuel de notation : Section A.1.1 [Font markup], page 837, Section 23.5 [Changement des fontes musicales], page 669, Section 8.3.3 [Familles de fontes], page 348, Section 8.3.2 [Localisation des fontes], page 348, Section 8.2.2 [Sélection de la fonte et de la taille], page 333.

Notation spécialisée

9 Musique vocale



Ce chapitre traite de la musique vocale : comment la saisir et comment s'assurer que les paroles s'alignent avec les notes de la mélodie correspondante.

9.1 Vue d'ensemble de la musique vocale

En complément de généralités, ce sous-chapitre aborde quelques styles particuliers en terme de musique vocale.

9.1.1 Références en matière de musique vocale

Graver de la musique vocale soulève plusieurs problèmes ; ils sont abordés soit dans ce chapitre, soit dans d'autres parties de la documentation de LilyPond.

- La plupart du temps, les paroles ne sont constituées que de texte simple. Cette forme de notation est abordée dans Section "Écriture de chants simples" dans *Manuel d'initiation*.
- La musique vocale nécessite souvent de recourir au mode `markup`, aussi bien pour des paroles que pour d'autres éléments textuels comme le nom des personnages. Cette syntaxe est expliquée dans Section 8.2.1 [Introduction au formatage de texte], page 329.
- L'impression d'un *ambitus* – ou tessiture – que l'on trouve en tête de certaines partitions, est abordée dans Section 1.3.7 [Ambitus], page 41.
- Les indications de nuance viennent, par défaut, se placer sous la portée. Il en va différemment pour la musique vocale, de telle sorte qu'elles ne soient pas mélangées avec les paroles. Ceci fait l'objet de la rubrique Section 9.5.2 [Mise en forme d'une partition chorale], page 402.

Voir aussi

Glossaire musicologique : Section "ambitus" dans *Glossaire*.

Manuel d'initiation : Section "Écriture de chants simples" dans *Manuel d'initiation*.

Manuel de notation : Section 1.3.7 [Ambitus], page 41, Section 8.2.1 [Introduction au formatage de texte], page 329, Section 9.5.2 [Mise en forme d'une partition chorale], page 402.

Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

9.1.2 Saisie des paroles

Il existe un mode de saisie spécialement adapté aux paroles. On l'introduit avec le mot-clé `\lyricmode`, ou en utilisant `\addlyrics` ou `\lyricsto`. Ce mode permet de saisir des paroles ainsi que leur ponctuation, de telle sorte que le caractère a ne sera plus interprété comme une note, un *la* pour les latinistes, mais comme une syllabe. Les syllabes sont saisies comme des notes, mais les hauteurs sont alors remplacées par du texte. Exemple avec une comptine anglaise :

```
\lyricmode { Three4 blind mice,2 three4 blind mice2 }
```

Il y a deux manières principales de préciser la place exacte des syllabes : soit en spécifiant explicitement la durée de chaque syllabe – comme dans l'exemple ci-dessus – soit en alignant automatiquement les paroles sur les notes d'une mélodie ou d'une voix en utilisant `\addlyrics` ou `\lyricsto`. La première méthode est abordée plus en détail à la rubrique Section 9.1.5 [Durée explicite des syllabes], page 362, la deuxième à la rubrique Section 9.1.4 [Durée automatique des syllabes], page 359.

Dans les paroles, un mot ou une syllabe commence par une lettre de l'alphabet, et se termine par une espace. Toute syllabe doit donc être séparée d'une autre par une espace, tout autre caractère – chiffre ou ponctuation – étant considéré comme partie intégrante de cette même syllabe. L'exemple suivant comporte une faute de frappe évidente :
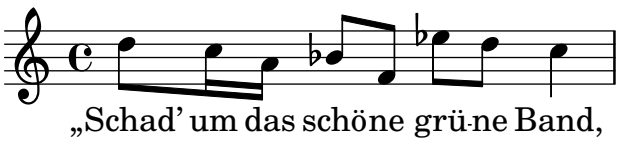
```
\lyricmode { lah lah lah}
```

la dernière syllabe contient une `}` ; il y a de fait un défaut de parité avec l'accolade ouvrante, et la compilation échouera fort probablement. Prenez dès à présent l'habitude de toujours encadrer d'espaces une accolade :

```
\lyricmode { lah lah lah }
```
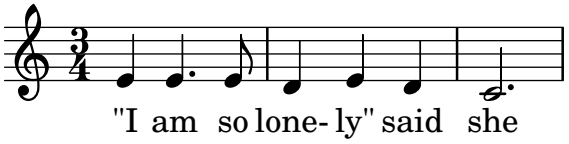
Pour utiliser des lettres accentuées ou des caractères spéciaux – cœurs ou guillemets inversés par exemple – il suffit de les insérer dans le fichier et de veiller à le sauvegarder avec le codage UTF-8. Voir à ce sujet Section 22.4 [Caractères spéciaux], page 664, pour plus de détails.

```
\relative { d''8 c16 a bes8 f ees' d c4 }
\addlyrics { „Schad' um das schö -- ne grü -- ne Band, }
```



Pour utiliser des guillemets informatiques standard, faites-les précéder d'une barre oblique inverse et encadrez d'une paire de guillemets la syllabe ainsi composée :

```
\relative { \time 3/4 e'4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone- "ly\"" said she }
```



Expliquer exactement comment LilyPond repère le début d'un mot en mode paroles (*Lyrics*) est quelque peu compliqué. En mode *Lyrics*, un mot peut commencer par : tout caractère alphabétique, `_`, `?`, `!`, `:`, `'`, un des codes de contrôle `^A` à `^F` et `^Q` à `^W`, `^Y`, `^^`, tout caractère ASCII de code strictement supérieur à 127, ou enfin l'un des symboles `` ` ``, `'`, `"` ou `^`, s'il est précédé d'une barre oblique inverse.

LilyPond permet de contrôler très finement le rendu des paroles grâce au mode `\markup`, utilisable y compris au sein du mode `\lyricmode`. Des explications complètes sont disponibles au chapitre Section 8.2 [Mise en forme du texte], page 329.
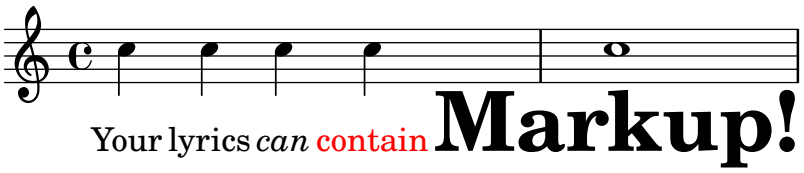
Morceaux choisis

Mise en forme individuelle de syllabes

La commande `\markup` permet d'individualiser la mise en forme de certaines syllabes d'un bloc `\lyricmode`.

```
mel = \relative c'' { c4 c c c c1 }
lyr = \lyricmode {
  Your lyrics \markup { \italic can }
  \markup { \with-color #red contain }
  \markup { \fontsize #8 \bold Markup! }
}

<<
  \new Voice = "melody" \mel
  \new Lyrics \lyricsto "melody" \lyr
>>
```



Voir aussi

Manuel d'initiation : Section "Chansons" dans *Manuel d'initiation*

Manuel de notation : Section 22.4 [Caractères spéciaux], page 664, Section 9.1.4 [Durée automatique des syllabes], page 359, Section 9.1.5 [Durée explicite des syllabes], page 362, Section 8.3 [Fontes], page 347, Section 8.2 [Mise en forme du texte], page 329, Chapitre 19 [Modes de saisie], page 605.

Référence des propriétés internes : Section "LyricText" dans *Référence des propriétés internes*.

9.1.3 Alignement des paroles sur la mélodie

Les paroles sont interprétées à partir à partir du mode `\lyricmode` et imprimées dans un contexte `Lyrics` – voir Section 33.1 [Tout savoir sur les contextes], page 760.

```
\new Lyrics \lyricmode { ... }
```

Deux variantes à `\lyricmode` permettent de plus d'associer un contexte pour synchroniser les syllabes à la musique. La plus commode consiste à ajouter un `\addlyrics` directement après le contenu musical du contexte `Voice` qui devrait se synchroniser avec le contexte `Lyrics` alors implicitement créé. L'instruction `\lyricsto` est plus versatile en ceci qu'elle requiert de spécifier à la fois le contexte `Voice` associé et de créer explicitement un contexte `Lyrics` pour contenir les paroles. Pour de plus amples détails, voir Section 9.1.4 [Durée automatique des syllabes], page 359.

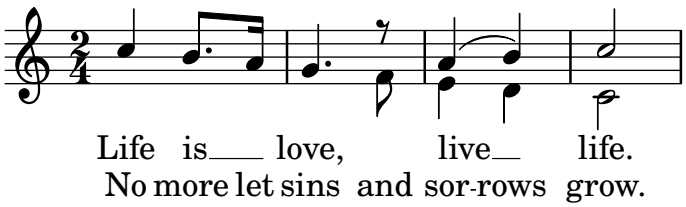
Vous disposez de deux méthodes pour aligner des paroles sur une mélodie :

- Les paroles peuvent s'aligner automatiquement, la durée des syllabes étant déterminée à partir d'un contexte de voix ou, dans certaines circonstances, une mélodie associée, grâce aux commandes `\addlyrics` et `\lyricsto` ou en définissant la propriété `associatedVoice`. Ceci est détaillé à la rubrique Section 9.1.4 [Durée automatique des syllabes], page 359.

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e4 d c2
    }
  >>

% takes durations and alignment from notes in "one"
  \new Lyrics \lyricsto "one" {
    Life is __ _ love, live __ life.
  }

% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



La première ligne de paroles est saisie selon la méthode la plus simple.

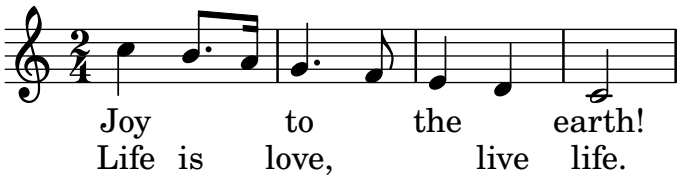
Vous pouvez constater, dans la deuxième ligne, que les paroles s'alignent selon les durées d'une voix différente. Ceci est particulièrement utile lorsque le texte s'agence différemment selon les couplets et que les durées sont accessibles grâce à des contextes `Voice` particuliers. Pour de plus amples détails, rendez-vous à la rubrique Section 9.3 [Couplets], page 388.

- Les paroles s'aligneront indépendamment de la valeur des notes dès lors que vous utiliserez le mode `\lyricmode` et affecterez explicitement leur durée à chaque syllabe.

```
<<
  \new Voice = "one" \relative {
    \time 2/4
    c''4 b8. a16 g4. f8 e4 d c2
  }

% uses previous explicit duration of 2;
  \new Lyrics \lyricmode {
    Joy to the earth!
  }
```

```
% explicit durations, set to a different rhythm
  \new Lyrics \lyricmode {
    Life4 is love,2. live4 life.2
  }
>>
```



La première ligne de paroles ne s'aligne pas vraiment sur les notes parce qu'aucune durée n'a été spécifiée. En fait, LilyPond adopte la dernière durée mentionnée, un 2, et l'applique à chaque mot.

La deuxième ligne illustre la manière d'aligner des paroles sans tenir compte de la durée des notes. Cette façon de procéder permet de traiter un alignement différent selon les couplets lorsqu'il n'y a pas moyen de déduire les durées à partir d'un contexte musical ; la rubrique Section 9.1.5 [Durée explicite des syllabes], page 362, aborde ceci plus en détails. Cette technique permet aussi d'ajouter des dialogues, comme indiqué à la rubrique Section 9.6.5 [Dialogue et musique], page 411.

Voir aussi

Manuel d'initiation : Section "Alignement des paroles sur une mélodie" dans *Manuel d'initiation*.

Référence des propriétés internes : Section "Lyrics" dans *Référence des propriétés internes*.

9.1.4 Durée automatique des syllabes

Les paroles peuvent être automatiquement alignées sous une mélodie, de trois manières différentes :

- en utilisant la commande `\lyricsto` pour spécifier le contexte de voix qui contient la mélodie,
- en introduisant les paroles par la commande `\addlyrics`, placée juste après le contexte `Voice` qui contient la mélodie,
- en définissant la propriété `associatedVoice` pour synchroniser les paroles avec un autre contexte de voix, ce à n'importe quel moment.

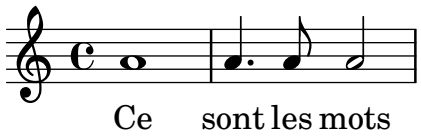
Ces trois méthodes permettent d'ajouter les traits d'union séparant les syllabes d'un même mot ainsi que d'indiquer la tenue de la dernière syllabe. Ceci fait l'objet de la rubrique Section 9.1.8 [Traits d'union et de prolongation], page 367.

Le contexte `Voice` contenant la mélodie sur laquelle les paroles vont s'aligner doit rester actif, au risque de voir la suite du texte disparaître. Ceci peut se produire lorsqu'il y a des moments où l'on ne chante pas. La rubrique Section 33.3 [Conservation d'un contexte], page 767, vous indiquera comment maintenir un contexte actif.

Utilisation de `\lyricsto`

Vous pouvez aligner automatiquement des paroles sous une mélodie en spécifiant à l'aide de la commande `\lyricsto` le contexte de voix qui contient cette mélodie :

```
<<
  \new Voice = "melodie" \relative {
    a'1 a4. a8 a2
  }
  \new Lyrics \lyricsto "melodie" {
    Ce sont les mots
  }
>>
```



Cette commande adapte les paroles aux notes de la voix (contexte `Voice` dans le jargon LilyPond) *melodie*. Ce contexte `Voice` doit exister avant l'affectation des paroles par `\lyricsto`. La commande `\lyricsto` introduit automatiquement le mode `\lyricmode`. Les paroles viendront par défaut se placer en dessous des notes. Pour un autre positionnement, voir Section 9.2.2 [Positionnement vertical des paroles], page 370.

Utilisation de `\addlyrics`

La commande `\addlyrics` n'est en fait qu'une manière plus aisée d'écrire de la musique vocale dans une structure LilyPond plus complexe.
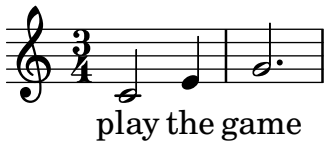
```
{ MUSIQUE }
\addlyrics { PAROLES }
```

revient au même que

```
\new Voice = "blah" { MUSIQUE }
\new Lyrics \lyricsto "blah" { PAROLES }
```
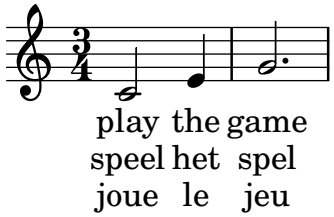
En voici un exemple :

```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
}
```



On peut ajouter davantage de couplets en multipliant le nombre de sections `\addlyrics`.

```
{
  \time 3/4
  \relative { c'2 e4 g2. }
  \addlyrics { play the game }
  \addlyrics { speel het spel }
  \addlyrics { joue le jeu }
}
```



Cependant, la commande `\addlyrics` ne peut pas gérer les constructions polyphoniques et ne permet pas d'associer des paroles à un contexte `TabVoice`. Dans ces cas là, mieux vaut employer `\lyricsto`.

Utilisation de `associatedVoice`
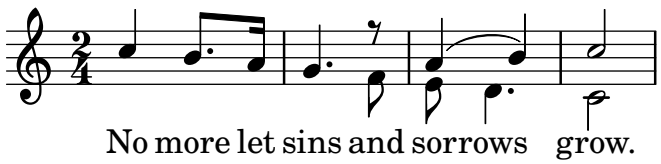
La propriété `associatedVoice` permet de basculer de mélodie pour la synchronisation des paroles. Elle s'emploie de la manière suivante :

```
\set associatedVoice = "lala"
```

La valeur que vous attribuez à cette propriété (ici `"lala"`) doit désigner un contexte `Voice` nommé, sans quoi les mélismes ne seront pas imprimés correctement.

Voici un exemple de cette manière de procéder :

```
<<
  \new Staff <<
    \time 2/4
    \new Voice = "one" \relative {
      \voiceOne
      c''4 b8. a16 g4. r8 a4 ( b ) c2
    }
    \new Voice = "two" \relative {
      \voiceTwo
       s2 s4. f'8 e8 d4. c2
    }
  >>
% takes durations and alignment from notes in "one" initially
% then switches to "two"
  \new Lyrics \lyricsto "one" {
    No more let
    \set associatedVoice = "two"  % must be set one syllable early
    sins and sor -- rows grow.
  }
>>
```



Voir aussi

Manuel de notation : Section 33.3 [Conservation d'un contexte], page 767, Section 9.1.8 [Traits d'union et de prolongation], page 367.
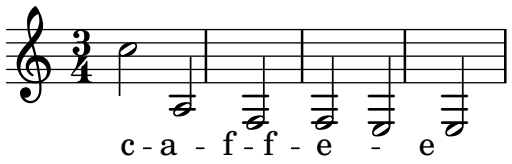
9.1.5 Durée explicite des syllabes

On peut aussi se passer de `\addlyrics`, `\lyricsto` et `associatedVoice` pour saisir des paroles. Dans ce cas, les syllabes sont entrées comme des notes – du texte remplaçant les hauteurs – ce qui veut dire que vous devez définir leur durée explicitement.
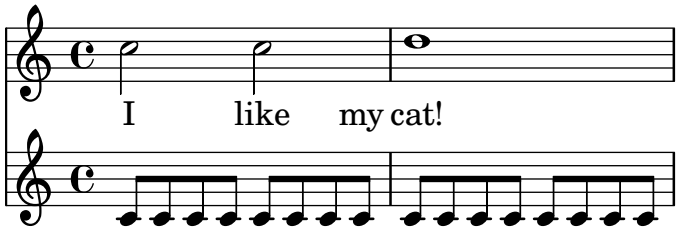
Les traits d'union seront imprimés entre les syllabes, à l'inverse des mélismes puisqu'il n'y a pas de voix associée.

Voici deux illustrations de cette technique :

```
<<
  \new Voice = "melody" {
    \time 3/4
    c''2 a f f e e
  }
  \new Lyrics \lyricmode {
    c4. -- a -- f -- f -- e2. -- e
  }
>>
```
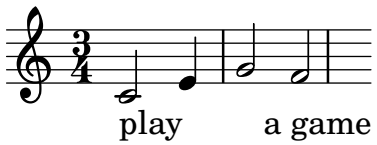



```
<<
  \new Staff {
    \relative {
      c''2 c2
      d1
    }
  }
  \new Lyrics {
    \lyricmode {
      I2 like4. my8 cat!1
    }
  }
  \new Staff {
    \relative {
      c'8 c c c c c c c
      c8 c c c c c c c
    }
  }
>>
```



Cette manière de procéder est tout à fait adaptée lorsqu'un fond musical accompagne des dialogues – voir Section 9.6.5 [Dialogue et musique], page 411.

Les syllabes seront alignées selon la dérogation apportée à la propriété `self-alignment-X` :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    c'2 e4 g2 f
  }
  \new Lyrics \lyricmode {
    \override LyricText.self-alignment-X = #LEFT
    play1 a4 game4
  }
>>
```



Voir aussi

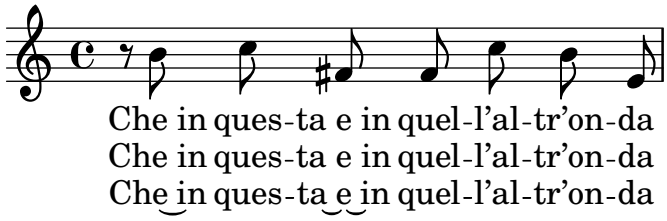
Manuel de notation : Section 9.6.5 [Dialogue et musique], page 411.

Référence des propriétés internes : Section "Lyrics" dans *Référence des propriétés internes*, Section "Voice" dans *Référence des propriétés internes*.

9.1.6 Plusieurs syllabes sur une note

Pour attribuer plus d'une syllabe à une même note, vous pouvez soit les mettre entre guillemets, soit utiliser le caractère souligné (_) pour obtenir une espace, ou bien encore utiliser un tilde (~) pour obtenir une liaison entre les syllabes.

```
{
  \relative {
    \autoBeamOff
    r8 b' c fis, fis c' b e,
  }
  \addlyrics
  {
    % Ensure hyphens are visible
    \override LyricHyphen.minimum-distance = 1.0
    Che_in ques -- ta_e_in quel -- l'al -- tr'on -- da
  }
  \addlyrics { "Che in" ques -- "ta e in" quel -- l'al -- tr'on -- da }
  \addlyrics { Che~in ques -- ta~e~in quel -- l'al -- tr'on -- da }
}
```



Voir aussi

Référence des propriétés internes : Section "LyricCombineMusic" dans *Référence des propriétés internes*.

9.1.7 Plusieurs notes pour une même syllabe

Parfois, tout particulièrement en musique médiévale ou baroque, plusieurs notes correspondent à une même syllabe. Ces vocalises sont appelées Section "mélismes" dans *Glossaire*. La syllabe à vocaliser est traditionnellement alignée par la gauche sur la première note du mélisme.
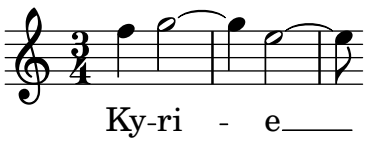
Lorsqu'un mélisme tombe sur une syllabe autre que la dernière d'un mot, un trait d'union étiré, indiqué par un double tiret `--` dans le fichier source, reliera cette syllabe à la suivante.

Lorsqu'un mélisme tombe sur la dernière syllabe d'un mot ou que ce mot n'en comporte qu'une, l'usage est d'indiquer la « tenue » jusqu'à la dernière note de la vocalise. Ceci s'obtient en ajoutant un double caractère souligné `__` après cette syllabe.

Vous disposez de cinq méthodes pour indiquer la présence d'un mélisme :

- Une liaison de prolongation constitue de fait un mélisme :
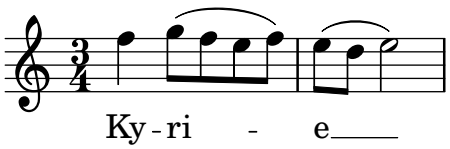
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g2 ~ |
    4 e2 ~ |
    8
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
>>
```



- LilyPond considère une liaison d'articulation comme un mélisme – il s'étendra de la première à la dernière note couverte par cette liaison. Il s'agit là de la façon traditionnelle de saisir des paroles :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 ( f e f )
    e8 ( d e2 )
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e __
  }
```
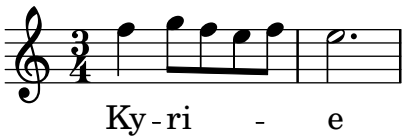
```
>>
```



Notez bien qu'une liaison de phrasé – indiquée par `\(...\)` – n'a aucune incidence sur la gestion des mélismes.

- LilyPond considère des notes regroupées par une ligature manuelle comme un mélisme, si tant est que la procédure de ligature automatique a été désactivée – voir Section 2.4.2 [Définition des règles de ligature automatique], page 107.
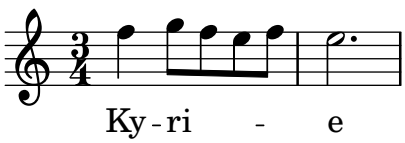
```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \autoBeamOff
    f''4 g8[ f e f]
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```



Ceci ne peut, vous en conviendrez, prendre en compte des durées plus longues que la croche.
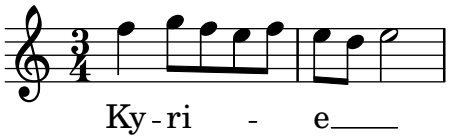
- LilyPond considère un groupe de notes non liées, mais encadrées par `\melisma` et `\melismaEnd`, comme constituant un mélisme :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8
    \melisma
    f e f
    \melismaEnd
    e2.
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e
  }
>>
```
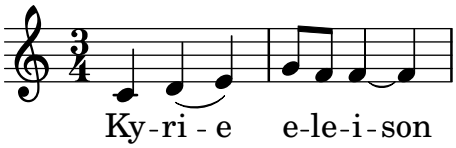


- Vous pouvez indiquer un mélisme directement dans les paroles, à l'aide d'un caractère souligné simple _ pour chaque note faisant partie de la vocalise :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    f''4 g8 f e f
    e8 d e2
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ _ _ e __ _ _
  }
>>
```



Vous pouvez totalement désactiver l'interprétation des liaisons de prolongation ou d'articulation et des ligatures apparaissant dans une mélodie comme fait générateur d'un mélisme. Il suffit en ce cas de définir `melismaBusyProperties` :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] f4 ~ 4
  }
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- e e -- le -- i -- son
  }
>>
```



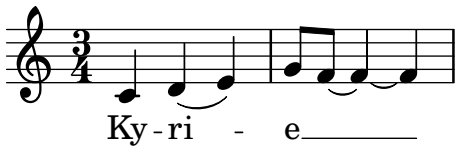
Certains réglages de `melismaBusyProperties` permettent de prendre en compte ou non les liaisons de tenue, les liaisons d'articulation et les ligatures dans la détection automatique des mélismes – voir `melismaBusyProperties` à la rubrique Section "Tunable context properties" dans *Référence des propriétés internes*.

Dans le cas où les indications de mélisme doivent être totalement ignorées, il vous faudra alors activer `ignoreMelismata` – voir Section 9.3.5 [Rythme différent selon le couplet], page 393.

Lorsque, dans un passage où la propriété `melismaBusyProperties` est active, survient un mélisme, vous pouvez l'indiquer dans les paroles par un simple caractère souligné pour chaque note à inclure :

```
<<
  \new Voice = "melody" \relative {
    \time 3/4
    \set melismaBusyProperties = #'()
    c'4 d ( e )
    g8 [ f ] ~ 4 ~ f
  }
```

```
  \new Lyrics \lyricsto "melody" {
    Ky -- ri -- _ e __ _ _ _
  }
>>
```



Commandes prédéfinies

`\autoBeamOff`, `\autoBeamOn`, `\melisma`, `\melismaEnd`.

Voir aussi

Glossaire musicologique : Section “melisma” dans *Glossaire*.

Manuel d’initiation : Section “Alignement des paroles sur une mélodie” dans *Manuel d’initiation*.

Manuel de notation : Section 9.1.3 [Alignement des paroles sur la mélodie], page 357, Section 2.4.2 [Définition des règles de ligature automatique], page 107, Section 9.1.4 [Durée automatique des syllabes], page 359, Section 9.3.5 [Rythme différent selon le couplet], page 393.

Référence des propriétés internes : Section “Tunable context properties” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Certains mélismes ne sont pas détectés automatiquement ; vous devrez alors prolonger vous-même les syllabes concernées à l’aide d’un double caractère souligné.

9.1.8 Traits d’union et de prolongation

Un mélisme sur la dernière syllabe d’un mot est indiqué par une longue ligne horizontale basse s’étirant jusqu’à la syllabe suivante. Une telle ligne, que nous appellerons prolongateur ou extenseur, s’obtient en saisissant ‘ __ ’ (notez les espaces entourant le double caractère souligné). Il est aussi possible de créer automatiquement des lignes d’extension en activant la propriété `autoExtenders`.

> **Note :** Dans une partition, les mélismes, ou vocalises, sont matérialisés par une ligne de prolongation. On l’indique par un double caractère souligné. Lorsqu’ils sont assez courts, ces mélismes peuvent s’indiquer par un souligné unique, ce qui aura pour effet de sauter une note à chaque fois et de ne pas imprimer de ligne.

hyphnation hyphnation, trait d’union

Un trait d’union séparant deux syllabes d’un même mot s’obtient en saisissant ‘ -- ’ (notez les espaces entourant le tiret double). Ce trait d’union sera centré entre les deux syllabes et sa longueur sera proportionnelle à l’espace les séparant.

Dans les partitions très serrées, les traits d’union peuvent ne pas être imprimés. Cet inconvénient peut être contrôlé par les propriétés `minimum-distance` pour gérer l’espace minimum entre deux syllabes, et `minimum-length`, seuil en deçà duquel il n’y a pas de trait d’union, toutes deux attachées à l’objet `LyricHyphen`.

Un trait d’union ne sera pas, par défaut, imprimé après un saut de ligne lorsque le nouveau système débute sur une nouvelle syllabe. Basculer la propriété `after-line-breaking` à `#t` permet de répéter le trait d’union en pareille situation.

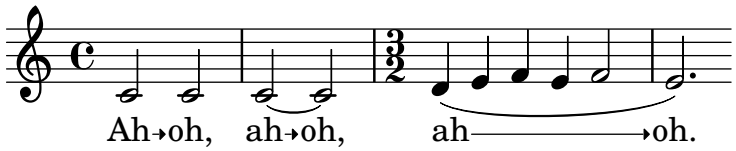
Voir aussi

Référence des propriétés internes : Section “LyricExtender” dans *Référence des propriétés internes*, Section “LyricHyphen” dans *Référence des propriétés internes*.

9.1.9 Changement graduel de voyelle

Un changement graduel de voyelle (ou une consonne maintenue) peut s'indiquer en ajoutant une flèche entre les syllabes, à l'aide de la commande `\vowelTransition` – voir Gould p. 452–453. La flèche indique la longueur de la transition et sera par défaut toujours affichée – de l'espace sera ajouté en cas d'espacement contraint. Une liaison de tenue ou de vocalise indique une absence d'articulation si ce n'est un changement de voyelle. La taille minimale de la flèche se règle à l'aide de la propriété `minimum-length` de l'objet `VowelTransition`.

```
{
  c'2 c'
  \set melismaBusyProperties = #'()
  c'2 ~ c'
  \time 3/2
  d'4( e' f' e' f'2
  e'2.)
}
\addlyrics
{
  Ah \vowelTransition oh,
  ah \vowelTransition oh,
  ah \vowelTransition _ _ _ _
  oh.
}
```



Voir aussi

Glossaire musicologique : Section “vowel transition” dans *Glossaire*.

Référence des propriétés internes : Section “VowelTransition” dans *Référence des propriétés internes*.
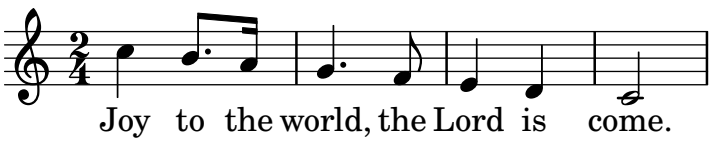
9.2 Situations particulières en matière de paroles

9.2.1 Travail avec des paroles et variables

Vous pouvez créer des variables pour contenir les paroles, dès lors que vous faites appel au mode approprié :

```
musicOne = \relative {
  c''4 b8. a16 g4. f8 e4 d c2
}
verseOne = \lyricmode {
  Joy to the world, the Lord is come.
}
\score {
  <<
```

```
    \new Voice = "one" {
      \time 2/4
      \musicOne
    }
    \new Lyrics \lyricsto "one" {
      \verseOne
    }
  >>
}
```
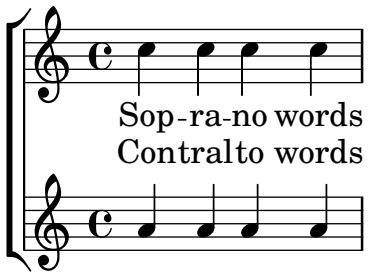


La fonction `\lyricmode` permet de définir une variable pour les paroles. Point n'est besoin de spécifier les durées si vous utilisez `\addlyrics` ou `\lyricsto` lorsque vous y faites référence.

Pour une organisation différente ou plus complexe, mieux vaut commencer par créer et alimenter les variables contenant mélodies et paroles, puis définir la hiérarchie des portées et des lignes de paroles, et enfin combiner correctement mélodies et paroles à l'aide de la commande `\context`. Vous serez ainsi assuré que la voix à laquelle il est fait référence par `\lyricsto` aura bien été préalablement définie, comme dans l'exemple suivant :

```
sopranoMusic = \relative { c''4 c c c }
contraltoMusic = \relative { a'4 a a a }
sopranoWords = \lyricmode { Sop -- ra -- no words }
contraltoWords = \lyricmode { Con -- tral -- to words }

\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \sopranoMusic
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos"
    \new Staff {
      \new Voice = "contraltos" {
        \contraltoMusic
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        \sopranoWords
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        \contraltoWords
      }
    }
  >>
```

```
}
```



Voir aussi
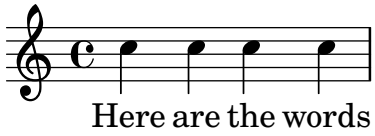
Manuel de notation : Section 9.2.2 [Positionnement vertical des paroles], page 370.

Référence des propriétés internes : Section “LyricCombineMusic” dans *Référence des propriétés internes*, Section “Lyrics” dans *Référence des propriétés internes*.

9.2.2 Positionnement vertical des paroles

Selon le type de musique, les paroles apparaîtront au-dessus ou au-dessous d’une portée ou bien entre deux portées. Positionner des paroles en dessous de la portée à laquelle elles se rattachent est de loin la chose la plus simple : il suffit de mentionner le contexte de paroles après le contexte de portée :

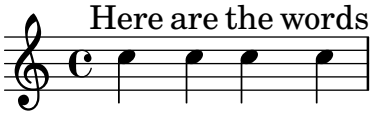
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Positionner les paroles au-dessus de la portée se fait de deux manières différentes, le plus simple étant d’utiliser la même syntaxe que ci-dessus, à ceci près que la ligne de paroles sera positionnée de manière explicite. L’argument à `alignAboveContext` est le nom d’un contexte `Staff` ou l’un de ses semblables – l’utilisation d’un `Voice` est inopérant.

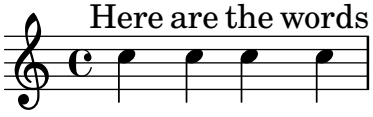
L’utilisation de `alignBelowContext` permet de positionner des paroles au-dessous d’une portée. Ce n’est en règle générale nécessaire que pour forcer le positionnement sous une portée qui n’est pas celle par défaut.

```
\score {
  <<
    \new Staff = "staff" {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics \with { alignAboveContext = "staff" } {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



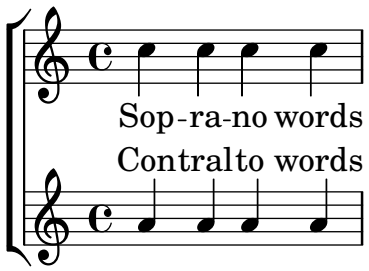
Autre façon de procéder, cette fois-ci en deux étapes. Nous commençons par déclarer un contexte `Lyrics` que nous laissons vide, puis les contextes `Staff` et `Voice`. Dans un deuxième temps, nous ajoutons l'instruction `\context` et la commande `\lyricsto` pour affecter les paroles au contexte de voix en question. Voici comment cela se présente :

```
\score {
  <<
    \new Lyrics = "lyrics" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "melody" {
        \relative { c''4 c c c }
      }
    }
    \context Lyrics = "lyrics" {
      \lyricsto "melody" {
        Here are the words
      }
    }
  >>
}
```



Lorsque deux voix sont isolées chacune sur une portée, vous pouvez placer les paroles entre les deux portées en utilisant l'une des méthodes que nous venons de voir. En voici un exemple, basé sur la deuxième méthode :

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice = "sopranos" {
        \relative { c''4 c c c }
      }
    }
    \new Lyrics = "sopranos"
    \new Lyrics = "contraltos" \with {
      % lyrics above a staff should have this override
      \override VerticalAxisGroup.staff-affinity = #DOWN
    }
    \new Staff {
      \new Voice = "contraltos" {
        \relative { a'4 a a a }
      }
    }
    \context Lyrics = "sopranos" {
      \lyricsto "sopranos" {
        Sop -- ra -- no words
      }
    }
    \context Lyrics = "contraltos" {
      \lyricsto "contraltos" {
        Con -- tral -- to words
      }
    }
  >>
}
```



Vous pouvez générer d'autres combinaisons de paroles et portées à partir de ces exemples, ou en examinant ce qui figure à la rubrique Section "Modèles pour ensemble vocal" dans *Manuel d'initiation* du manuel d'initiation.

Morceaux choisis

Agencement de paroles séparées sur une seule ligne

On peut vouloir positionner les paroles de différents intervenants sur une même ligne, notamment pour un dialogue par bribes. Cet extrait montre comment y parvenir en ajustant la propriété `nonstaff-nonstaff-spacing` de l'objet graphique `VerticalAxisGroup`.

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
              .nonstaff-nonstaff-spacing
              .minimum-distance = ##f
  }
}

aliceSings = \markup { \smallCaps "Alice" }
eveSings = \markup { \smallCaps "Eve" }

<<
  \new Staff <<
    \new Voice = "alice" {
      f'4^\aliceSings g' r2 |
      s1 |
      f'4^\aliceSings g' r2 |
      s1 | \break
      % ...
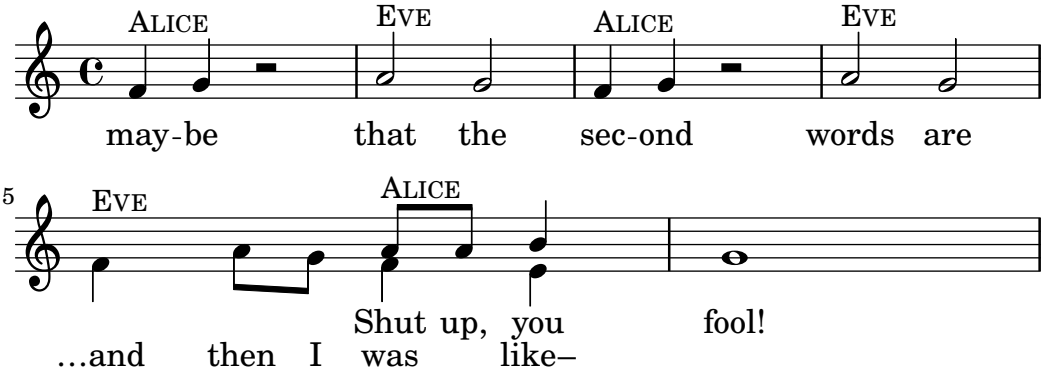
      \voiceOne
      s2 a'8^\aliceSings a' b'4 |
      \oneVoice
      g'1
    }
    \new Voice = "eve" {
      s1 |
      a'2^\eveSings g' |
      s1 |
      a'2^\eveSings g'
      % ...

      \voiceTwo
      f'4^\eveSings a'8 g' f'4 e' |
      \oneVoice
      s1
    }
  >>

  \new Lyrics \lyricsto "alice" {
    may -- be
    sec -- ond
    % ...
    Shut up, you fool!
  }

  \new Lyrics \lyricsto "eve" {
    that the
    words are
    % ...
    ...and then I was like--
```

```
  }
>>
```



Voir aussi

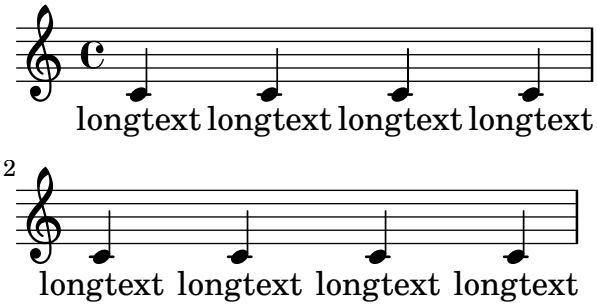
Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*.

Manuel de notation : Section 33.2 [Création et référencement d'un contexte], page 763, Section 33.7 [Ordonnancement des contextes], page 779.

9.2.3 Positionnement horizontal des syllabes

La propriété `minimum-distance` de l'objet `LyricSpace` permet d'accroître l'espacement des paroles.

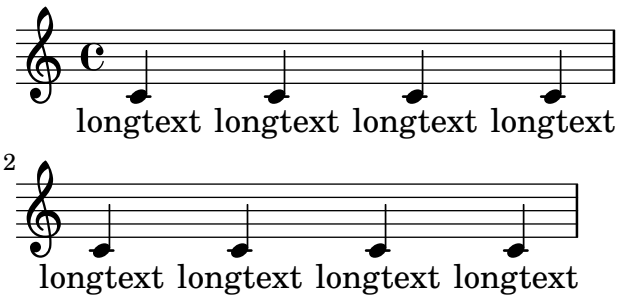
```
\relative c' {
  c c c c
  \override Lyrics.LyricSpace.minimum-distance = 1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```



Pour que ce réglage s'applique à toute la partition, définissez-le dans le bloc `\layout`.

```
\score {
  \relative {
  c' c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
```

```
  \layout {
    \context {
      \Lyrics
      \override LyricSpace.minimum-distance = 1.0
    }
  }
}
```



Morceaux choisis
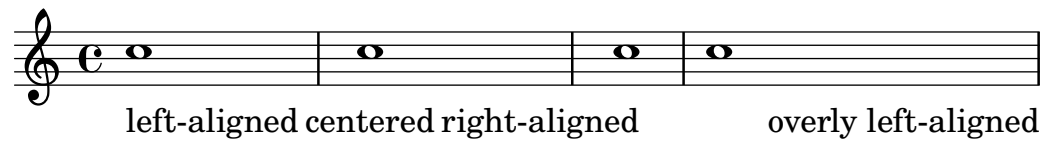
Alignement des syllabes

L'alignement horizontal des paroles peut se gérer à l'aide de la propriété `self-alignment-X` de l'objet `LyricText`. Les valeurs `-1` ou `#LEFT` produiront un alignement par la gauche, les valeurs `0` ou `#CENTER` un alignement centré, et les valeurs `1` ou `#RIGHT` un alignement par la droite. D'autres valeurs numériques sont permises – attention à ne pas oublier le préfixe Scheme '`#`' en cas de nombre négatif.

```
\layout {
  ragged-right = ##f
}

\relative c'' {
  c1 c c c
}

\addlyrics {
  \once \override LyricText.self-alignment-X = #LEFT
  "left-aligned"
  \once \override LyricText.self-alignment-X = #CENTER
  "centered"
  \once \override LyricText.self-alignment-X = 1
  "right-aligned"
  \once \override LyricText.self-alignment-X = #-1.5
  "overly left-aligned"
}
```

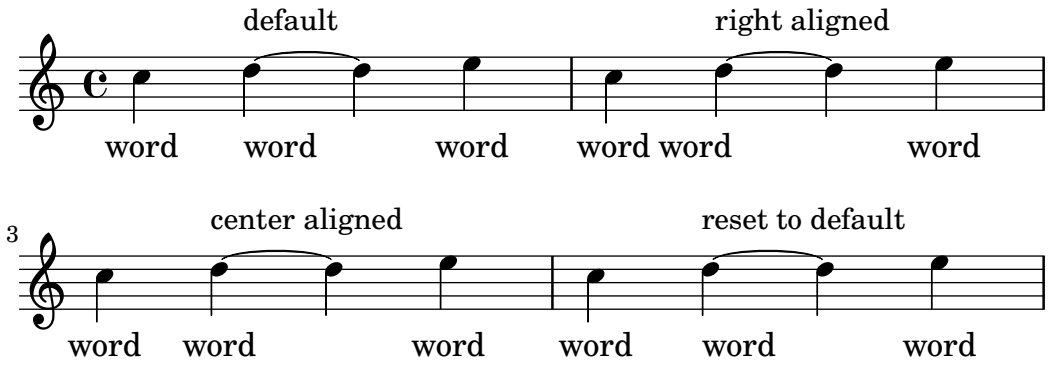


Alignement de la première syllabe d'un mélisme

Par défaut, une syllabe qui entame un mélisme est alignée sur la note par la gauche. Cet alignement peut se modifier à l'aide de la propriété `lyricMelismaAlignment`.

```
<<
  \new Staff {
    \new Voice = "vocal" \relative c'' {
      \override TextScript.staff-padding = #2
      c d-^\markup default  d e
      c d-^\markup "right aligned" d e
      c d-^\markup "center aligned" d e
      c d-^\markup "reset to default" d e
    }
  }
  \new Lyrics \lyricsto "vocal" {
    word word word
    \set lyricMelismaAlignment = #RIGHT
    word word word
    \set lyricMelismaAlignment = #CENTER
    word word word
    \unset lyricMelismaAlignment
    word word word
  }
>>

\layout {
  ragged-right = ##f
}
```



Problèmes connus et avertissements

L'assurance que tous les scripts textuels et les paroles resteront bien à l'intérieur des marges requiert des ressources non négligeables. Afin de réduire le temps de traitement, vous pouvez désactiver cette fonctionnalité en ajoutant

```
\override Score.PaperColumn.keep-inside-line = ##f
```

Pour s'assurer que les paroles ne seront pas traversées par des barres de mesure, il faut ajouter

```
\layout {
  \context {
    \Lyrics
      \consists Bar_engraver
      \consists Separating_line_group_engraver
      \hide BarLine
  }
}
```
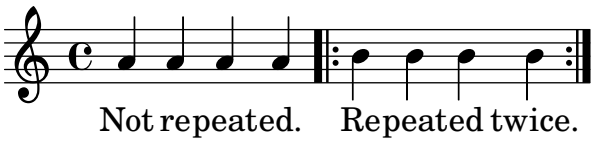
9.2.4 Paroles et reprises

La répétition de *fragments musicaux* est abordée de manière détaillée dans un Chapitre 4 [Répétitions et reprises], page 193. Nous nous intéresserons ici aux moyens d'ajouter des paroles à des reprises.

Reprises simples

Les paroles attachées à un fragment musical répété devraient adopter rigoureusement la même construction que la musique, si tant est qu'elles ne diffèrent pas d'une fois sur l'autre.
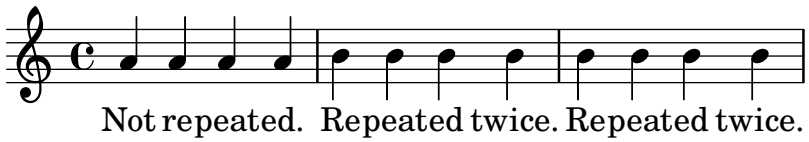
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- ed twice. }
      }
    }
  >>
}
```



Les mots seront alors correctement répétés si la reprise est développée.
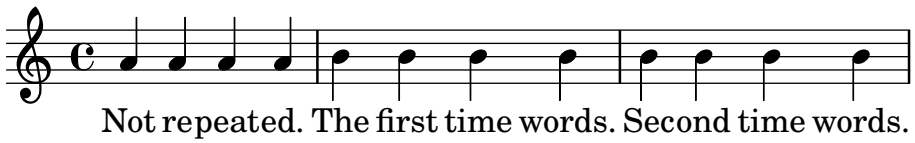
```
\score {
  \unfoldRepeats {
    <<
      \new Staff {
        \new Voice = "melody" {
          \relative {
            a'4 a a a
            \repeat volta 2 { b4 b b b }
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Not re -- peat -- ed.
          \repeat volta 2 { Re -- peat -- ed twice. }
        }
      }
    >>
  }
```

```
}
```



Lorsque la reprise est développée et que les paroles diffèrent, saisissez le texte normalement :

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \*2 { b4 b b b }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        The first time words.
        Sec -- ond time words.
      }
    }
  >>
}
```



Lorsque les paroles diffèrent pour une reprise non développée – utilisation de `volta` au lieu de `unfold` – les paroles en question doivent être saisies dans des contextes `Lyrics` séparés ; ils seront combinés dans une section parallèle :
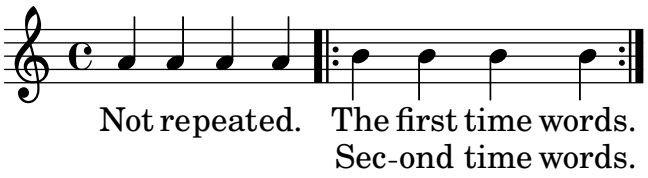
```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b b b }
        }
      }
    }
    \new Lyrics \lyricsto "melody" {
      Not re -- peat -- ed.
      <<

{ The first time words. }

\new Lyrics {
```

```
  \set associatedVoice = "melody"
  Sec -- ond time words.

}
      >>
    }
  >>
}
```



Et ce quel que soit le nombre de « couplets » :

```
\score {
  <<
    \new Staff {
      \new Voice = "singleVoice" {
        \relative {
  a'4 a a a
  \repeat volta 3 { b4 b b b }
          c4 c c c

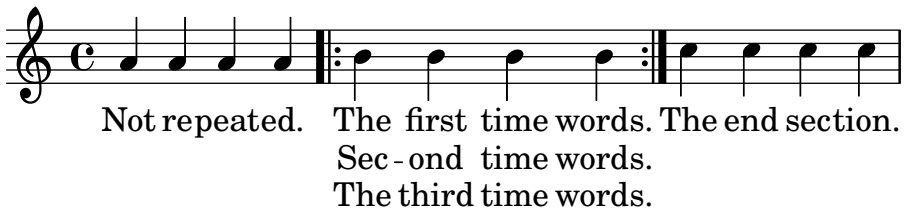
}
      }
    }
    \new Lyrics \lyricsto "singleVoice" {
      Not re -- peat -- ed.
      <<
        { The first time words. }

\new Lyrics {
  \set associatedVoice = "singleVoice"
  Sec -- ond time words.

}

\new Lyrics {
  \set associatedVoice = "singleVoice"
  The third time words.

}
      >>
      The end sec -- tion.
    }
  >>
}
```



Cependant, lorsque la partition comporte plusieurs portées, cas typique d'un `ChoirStaff`, les paroles des deuxième et troisième couplets seront repoussées sous la dernière portée. L'instruction `alignBelowContext` permet alors de les repositionner correctement :

```
\score {
  <<
    \new Staff {
      \new Voice = "melody" {
        \relative {
  a'4 a a a
  \repeat volta 3 { b4 b b b }
          c4 c c c

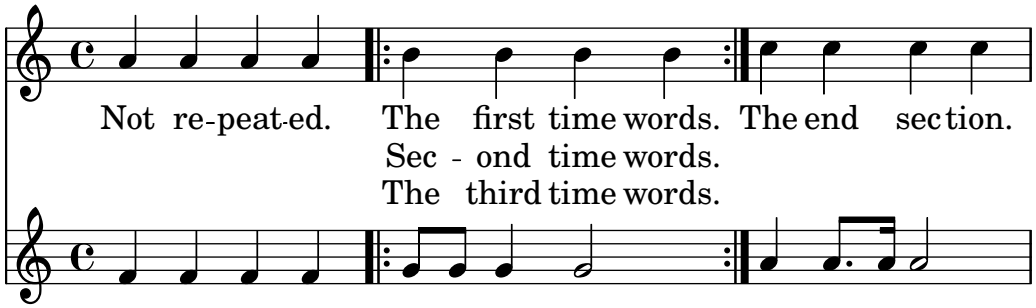
}
      }
    }
    \new Lyrics = "firstVerse" \lyricsto "melody" {
      Not re -- peat -- ed.
      <<
        { The first time words. }

\new Lyrics = "secondVerse"
        \with { alignBelowContext = "firstVerse" } {
  \set associatedVoice = "melody"
  Sec -- ond time words.

}

\new Lyrics = "thirdVerse"
        \with { alignBelowContext = "secondVerse" } {
  \set associatedVoice = "melody"
  The third time words.
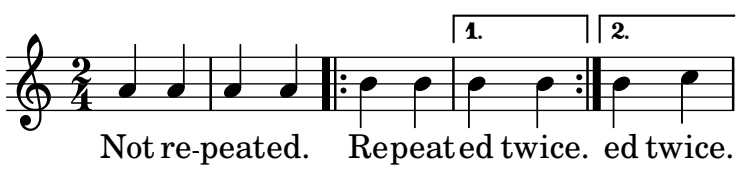
}
      >>
      The end sec -- tion.
    }
    \new Voice = "harmony" {
      \relative {
        f'4 f f f
        \repeat volta 2 { g8 g g4 g2 }
        a4 a8. a16 a2
      }
    }
  >>
}
```



Reprises avec alternative

Les paroles d'un fragment répété, lorsqu'elles sont identiques et qu'aucune alternative ne débute par un silence, peuvent adopter la même construction que la musique. Ceci permet par ailleurs une expansion correcte à la fois de la musique et des paroles lors de l'utilisation de `\unfoldRepeats`.

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          a'4 a a a
          \repeat volta 2 { b4 b }
          \alternative {
            \volta 1 { b b }
            \volta 2 { b c }
          }
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Not re -- peat -- ed.
        \repeat volta 2 { Re -- peat -- }
        \alternative {
          \volta 1 { ed twice. }
          \volta 2 { ed twice. }
        }
      }
    }
  >>
}
```

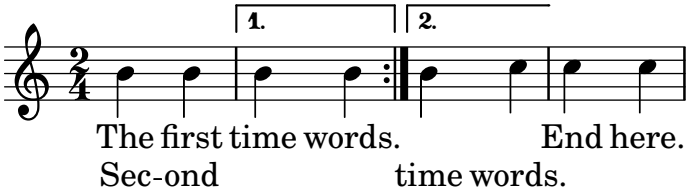


Cette identité de structure n'est toutefois pas possible lorsque les paroles sont différentes ou que l'un des blocs `\alternative` débute par un silence. Des instructions `\skip` devront venir s'insérer dans les paroles pour « sauter » les notes des alternatives qui ne les concernent pas.

N'utilisez pas de simple caractère souligné pour sauter une note. N'oubliez pas qu'un caractère souligné indique un mélisme ; la syllabe précédente sera donc alignée à gauche.

> **Note :** La commande `\skip` doit comporter une durée quelle qu'elle soit – elle sera toujours ignorée lorsque les paroles sont associées à une mélodie à l'aide de `\addlyrics` ou `\lyricsto`. Chaque `\skip` correspond à une seule note quelle qu'en soit la durée.

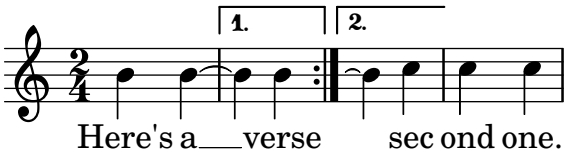
```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b }
          \alternative {
            \volta 1 { b b }
            \volta 2 { b c }
          }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        The first time words.
        \*2 { \skip 1 }
        End here.
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Sec -- ond
        \*2 { \skip 1 }
        time words.
      }
    }
  >>
}
```



Lorsqu'une note se prolonge sur les alternatives, la tenue est indiquée normalement pour la première alternative, et à l'aide de l'instruction `\repeatTie` pour les suivantes. Cette liaison « de répétition » pose problème en matière d'alignement des paroles puisque la longueur de l'alternative est accrue en raison de la liaison.

D'autre part, une liaison de prolongation crée un mélisme qui sera effectif pour la première alternative, mais pas pour les autres. La solution pour « recaler » les paroles consiste à désactiver temporairement la détection automatique de mélismes et insérer des « blancs ».

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \set melismaBusyProperties = #'()
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
          }
          \unset melismaBusyProperties
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        \repeat volta 2 { Here's a __ }
        \alternative {
          \volta 1 { \skip 1 verse }
          \volta 2 { \skip 1 sec }
        }
        ond one.
      }
    }
  >>
}
```

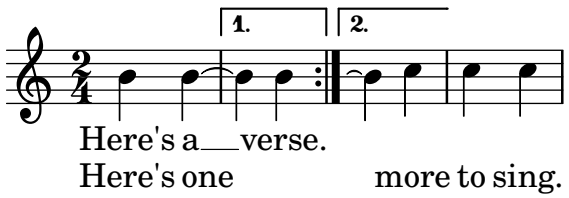


Notez bien que l'utilisation conjointe de `\unfoldRepeats` et de `\repeatTie` entraîne l'impression d'une double liaison, sauf à supprimer les `\repeatTie`.

Lorsque les paroles sont différentes sur le fragment répété, la construction avec `\repeat` est inefficace ; vous devrez alors insérer des blancs :

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
```
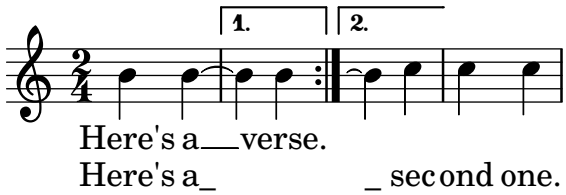
```
            }
            c4 c
          }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Here's a __ verse.
          \*2 { \skip 1 }
        }
      }
      \new Lyrics {
        \lyricsto "melody" {
          Here's one
          \*2 { \skip 1 }
          more to sing.
        }
      }
    >>
}
```



Les indications de mélisme et traits d'union en début d'alternative doivent être insérées manuellement :

```
\score {
  <<
    \new Staff {
      \time 2/4
      \new Voice = "melody" {
        \relative {
          \repeat volta 2 { b'4 b ~}
          \alternative {
            \volta 1 { b b }
            \volta 2 { b \repeatTie c }
          }
          c4 c
        }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
        Here's a __ verse.
        \*2 { \skip 1 }
      }
    }
    \new Lyrics {
      \lyricsto "melody" {
```

```
        Here's "a_"
        \skip 1
        "_" sec -- ond one.
      }
    }
  >>
}
```
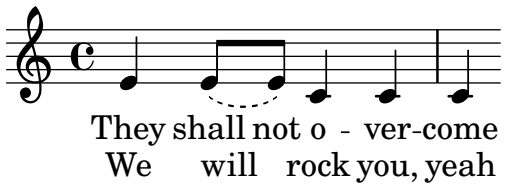


Voir aussi

Manuel de notation : Section 33.3 [Conservation d'un contexte], page 767, Chapitre 4 [Répétitions et reprises], page 193.

9.2.5 Paroles alternatives
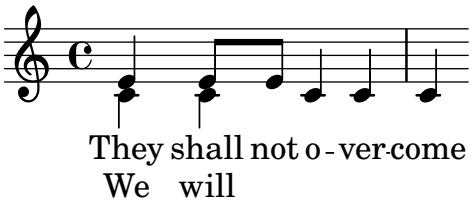
Il arrive parfois, dans un fragment répété, qu'une note soit divisée pour répondre au texte. Vous pouvez indiquer cette adaptation rythmique en désactivant temporairement la détection automatique des mélismes tout en spécifiant ces mélismes au niveau des paroles :

```
\score {
  <<
    \new Voice = "melody" {
      \relative c' {
        \set melismaBusyProperties = #'()
        \slurDashed
        e4 e8( e) c4 c |
        \unset melismaBusyProperties
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "melody" {
      We will _ rock you, yeah
    }
  >>
}
```



En donnant un nom à chaque voix et en leur attribuant spécifiquement des paroles, vous pourrez traiter le cas où notes et rythme diffèrent d'une fois sur l'autre :

```
\score {
  <<
    \new Voice = "melody" {
      \relative {
        <<
          {
            \voiceOne
            e'4 e8 e
          }
          \new Voice = "splitpart" {
            \voiceTwo
            c4 c
          }
        >>
        \oneVoice
        c4 c |
        c
      }
    }
    \new Lyrics \lyricsto "melody" {
      They shall not o -- ver -- come
    }
    \new Lyrics \lyricsto "splitpart" {
      We will
    }
  >>
}
```



Il n'est pas rare, en musique chorale, qu'une voix se divise pendant plusieurs mesures. Bien qu'une construction du type << {...} \\ {...} >>, où deux expressions musicales (ou plus) séparées par des doubles obliques inversées peuvent sembler être le moyen adéquat de définir cette division, **toutes** les expressions qu'elle contient seront assignées à de **nouveaux contextes de voix**, ce qui aura pour effet qu'aucune parole ne leur sera affectée – les paroles sont attachées au contexte de voix initial. Il vaut mieux construire ce passage comme une polyphonie temporaire – voir Section 5.2.1 [Polyphonie sur une portée], page 227.

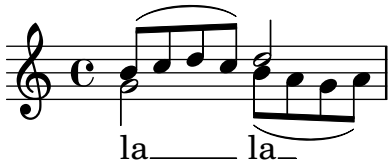
9.2.6 Polyphonie et paroles communes

Lorsque deux voix au rythme différent partagent les mêmes paroles, l'alignement des syllabes sur l'une des voix peut gêner la lecture de l'autre voix. Par exemple, la deuxième extension de syllabe ci-dessous est trop courte puisque les paroles ne sont alignées que sur la voix du haut :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
```

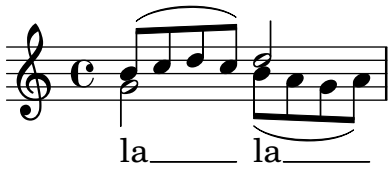
```
  \new Voice = "sopranoVoice" { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new Lyrics \lyricsto "sopranoVoice" \words
>>
```



Le résultat attendu sera obtenu grâce à l'alignement des paroles sur un contexte NullVoice supplémentaire, celui-ci contenant une combinaison judicieuse des deux voix. Les notes du contexte NullVoice, bien que n'apparaissant pas sur la version imprimable, peuvent servir à aligner correctement les syllabes :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

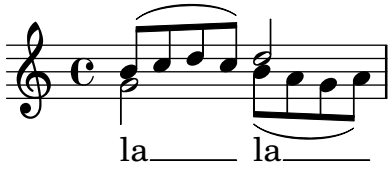
\new Staff <<
  \new Voice { \voiceOne \soprano }
  \new Voice { \voiceTwo \alto }
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```



Cette façon de procéder permet par ailleurs d'utiliser la fonction \partCombine qui, normalement, ne peut s'utiliser avec des paroles :

```
soprano = \relative { b'8( c d c) d2 }
alto = \relative { g'2 b8( a g a) }
aligner = \relative { b'8( c d c) b( a g a) }
words = \lyricmode { la __ la __ }

\new Staff <<
  \new Voice \partCombine \soprano \alto
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
>>
```



Problèmes connus et avertissements

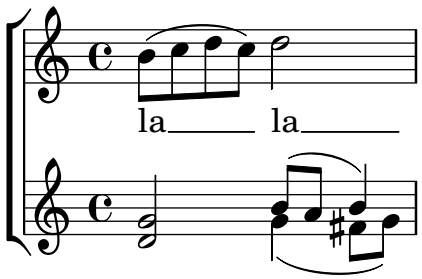
La commande `\addlyrics` ne peut traiter que des paroles attachées à un contexte `Voice` ; elle ne peut donc s'utiliser avec un `NullVoice`.

La fonction `\partCombine` est abordée en détails dans Section 5.2.5 [Regroupement automatique de parties], page 238.

Pour finir, cette méthode est aussi utilisable lorsque les voix sont sur des portées différentes, et ne se limite pas à deux voix :

```
soprano = \relative { b'8( c d c) d2 }
altoOne = \relative { g'2 b8( a b4) }
altoTwo = \relative { d'2 g4( fis8 g) }
aligner = \relative { b'8( c d c) d( d d d) }
words = \lyricmode { la __ la __ }

\new ChoirStaff \with { \accepts NullVoice } <<
  \new Staff \soprano
  \new NullVoice = "aligner" \aligner
  \new Lyrics \lyricsto "aligner" \words
  \new Staff \partCombine \altoOne \altoTwo
>>
```

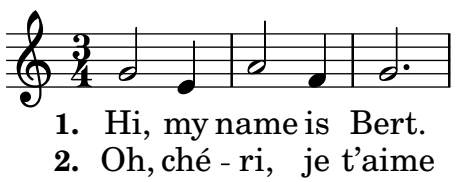


9.3 Couplets

9.3.1 Numérotation des couplets

On peut ajouter un numéro aux couplets en définissant la variable `stanza` :

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \stanza "1. "
  Hi, my name is Bert.
} \addlyrics {
  \stanza "2. "
  Oh, ché -- ri, je t'aime
}
```

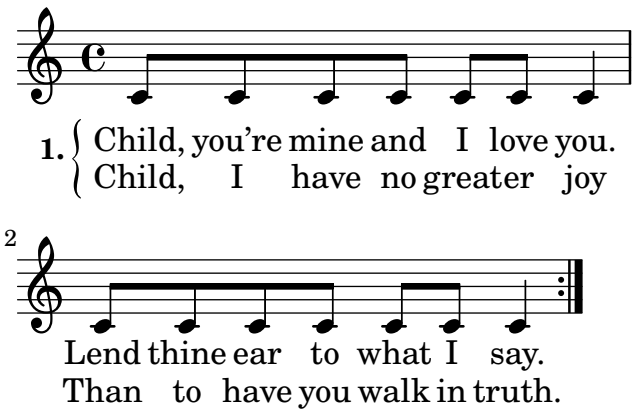


Ces numéros se placeront juste avant le début de la première syllabe. Deux lignes de couplet peuvent aussi être regroupées, par exemple dans le cas d'une reprise avec des paroles différentes.

```
stanzaOneOne = \lyricmode {
  \stanza \markup {
    \column {
      \vspace #.2
      \line { "1." \left-brace #30 }
    }
  }
  Child, you're mine and I love you.
  Lend thine ear to what I say.
}

stanzaOneThree = \lyricmode {
  Child, I have no great -- er joy
  Than to have you walk in truth.
}

\new Voice {
  \repeat volta 2 {
    c'8 c' c' c' c' c' c'4
    c'8 c' c' c' c' c' c'4
  }
}
\addlyrics { \stanzaOneOne }
\addlyrics { \stanzaOneThree }
```
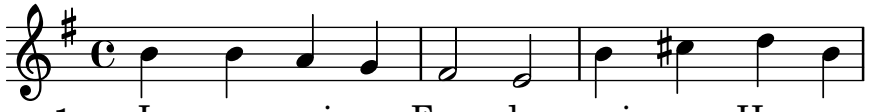


9.3.2 Rappel de couplet

Il est parfois désirable de répéter le numéro du couplet à chaque nouvelle ligne. Ceci s'obtient après avoir activé la propriété de contexte `stanzaReminders` :
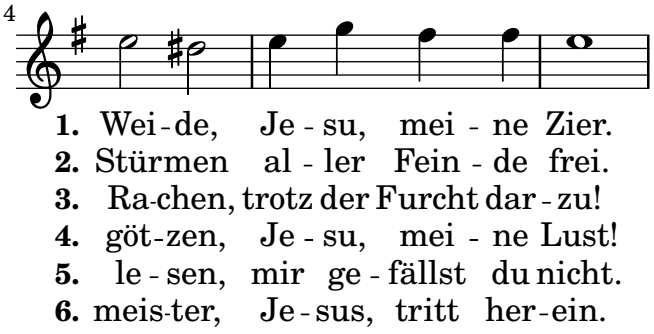
```
\layout {
  \set stanzaReminders = ##t
}

\new Voice \relative {
  \key e \minor
  b'4 b a g
  fis2 e
  b'4 cis d b \break
  e2 dis
  e4 g fis fis
```

```
  e1
}
\addlyrics {
  \stanza "1."
  Je -- su, mei -- ne Freu -- de,
  mei -- nes Her -- zens Wei -- de,
  Je -- su, mei -- ne Zier.
}
\addlyrics {
  \stanza "2."
  Un -- ter dei -- nem Schir -- men
  bin ich vor den Stür -- men
  al -- ler Fein -- de frei.
}
\addlyrics {
  \stanza "3."
  Trotz dem al -- ten Dra -- chen,
  trotz des To -- des Ra -- chen,
  trotz der Furcht dar -- zu!
}
\addlyrics {
  \stanza "4."
  Weg mit al -- len Schät -- zen!
  Du bist mein Er -- göt -- zen,
  Je -- su, mei -- ne Lust!
}
\addlyrics {
  \stanza "5."
  Gu -- te Nacht, o We -- sen,
  das die Welt er -- le -- sen,
  mir ge -- fällst du nicht.
}
\addlyrics {
  \stanza "6."
  Weicht, ihr Trau -- er -- geis -- ter,
  denn mein Freu -- den -- meis -- ter,
  Je -- sus, tritt her -- ein.
}
```
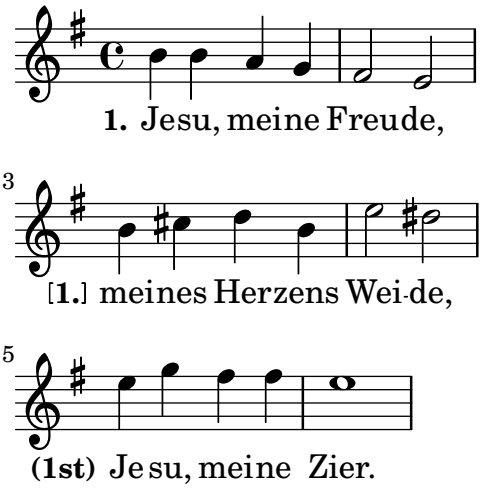


1. Je - su, mei-ne Freu-de, mei - nes Herzens
2. Un - ter dei-nem Schirmen bin ich vor den
3. Trotz dem al - ten Dra-chen, trotz des To - des
4. Weg mit al - len Schätzen! Du bist mein Er-
5. Gu - te Nacht, o We - sen, das die Welt er-
6. Weicht, ihr Trau-er - geis - ter, denn mein Freuden-



Les rappels de couplet sont, par défaut, identiques aux numéros de couplet correspondant. Ce comportement peut se modifier à l'aide de la propriété `stanzaReminderText`, soit en fournissant directement un nouveau *markup*, soit par une fonction qui accepte le *markup* du numéro de couplet originel en tant qu'argument.
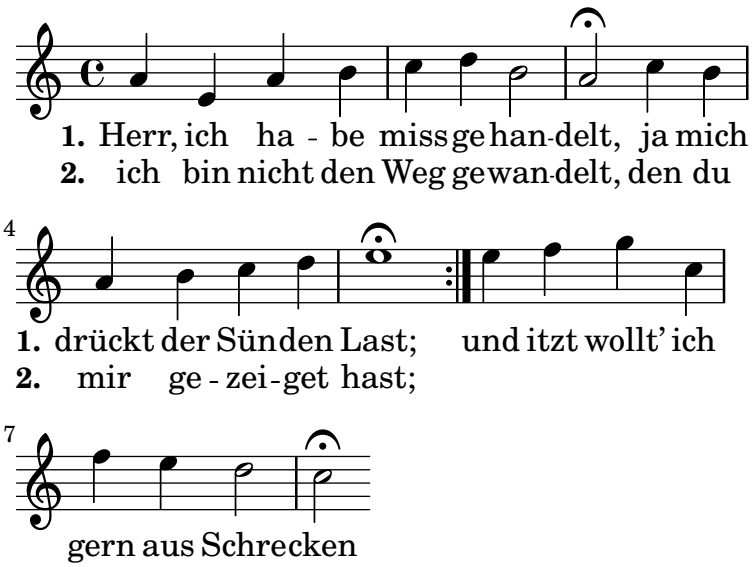
```
\new Voice \relative {
  \key e \minor
  b'4 b a g
  fis2 e \break
  b'4 cis d b
  e2 dis \break
  e4 g fis fis
  e1
}
\addlyrics {
  \stanza "1."
  \set stanzaReminders = ##t
  \set stanzaReminderText = #make-bracket-markup
  Je -- su, mei -- ne Freu -- de,
  mei -- nes Her -- zens Wei -- de,
  \set stanzaReminderText = "(1st)"
  Je -- su, mei -- ne Zier.
}
```



La commande `\noStanza` réinitialise le numéro de couplet de telle sorte qu'aucun rappel n'apparaisse à nouveau.

```
\layout {
  \set stanzaReminders = ##t
}

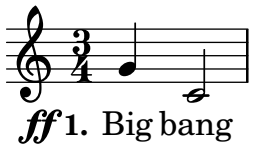
\new Voice \relative {
  \repeat volta 2 {
    a'4 e a b
    c d b2
    a\fermata c4 b
    \break
    a b c d
    e1\fermata
  }
  e4 f g c, \break
  f e d2
  c2 \fermata
}
\addlyrics {
  \stanza "1."
  Herr, ich ha -- be miss -- ge -- han -- delt,
  ja mich drückt der Sün -- den Last;
  \noStanza
  und itzt wollt' ich gern aus Schre -- cken
}
\addlyrics {
  \stanza "2."
  ich bin nicht den Weg ge -- wan -- delt,
  den du mir ge -- zei -- get hast;
}
```



9.3.3 Indication de nuance dans les couplets

Lorsque des couplets ont des nuances différentes, vous pouvez ajouter une nuance en regard de chacun d'eux. Bien qu'il n'y ait pas de procédé dédié pour traiter ce cas, elles peuvent s'insérer facilement et directement dans le numéro du couplet :

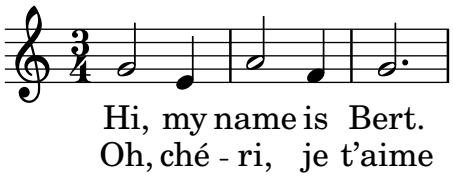
```
<<
  \new Voice = tune {
    \time 3/4
    g'4 c'2
  }
  \new Lyrics \lyricsto tune {
    \stanza \markup { \dynamic ff 1. }
     Big bang
  }
>>
```



9.3.4 Indication du personnage et couplets

On peut également ajouter le nom de chaque rôle ; ils s'imprimeront au début de chaque ligne comme les noms d'instrument. Il faut pour cela définir `vocalName`, et `shortVocalName` pour une version abrégée.

```
\new Voice \relative {
  \time 3/4 g'2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = "Bert"
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = "Ernie"
  Oh, ché -- ri, je t'aime
}
```



9.3.5 Rythme différent selon le couplet

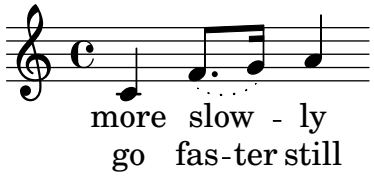
Il arrive assez souvent que le paroles de différents couplets, bien qu'attachées à une même mélodie, ne s'articulent pas de la même manière. La commande `\lyricsto` est cependant capable de gérer de telles situations.

Mélismes dans certaines strophes seulement

Il peut survenir que les paroles comportent un mélisme pour l'un des couplets, mais plusieurs syllabes pour d'autres. Une solution consiste à ignorer temporairement les mélismes dans le couplet ayant le plus de syllabes. Il suffit pour cela de définir la propriété `ignoreMelismata` à l'intérieur du contexte `Lyrics`.

Petit détail qui a son importance : l'activation de `ignoreMelismata` doit **précéder** la syllabe à partir de laquelle elle s'appliquera :

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    go
    \set ignoreMelismata = ##t
    fas -- ter
    \unset ignoreMelismata
    still
  }
>>
```



Problèmes connus et avertissements
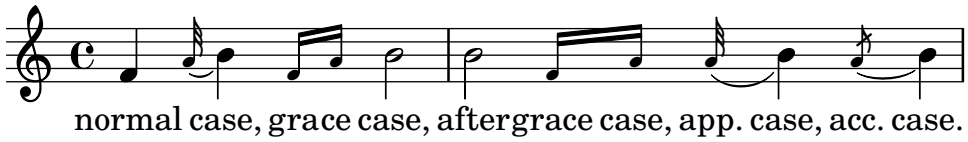
Contrairement aux autres utilisations de l'instruction `\set`, il n'est pas possible de la faire précéder d'un `\once` dans le cas de `\set ignoreMelismata`. Les paroles affectées par la propriété `ignoreMelismata` **doivent** être encadrées respectivement d'un `\set` et d'un `\unset`.

Syllabe sur note de passage

L'utilisation de la commande `\lyricsto` ne permet pas, par défaut, d'assigner une syllabe à des notes d'ornement – introduites par la commande `\grace`. Vous pouvez cependant y parvenir grâce à la propriété `includeGraceNotes` :

```
<<
  \new Voice = melody \relative {
    f'4 \appoggiatura a32 b4
    \grace { f16 a16 } b2
    \afterGrace b2 { f16[ a16] }
    \appoggiatura a32 b4
    \acciaccatura a8 b4
  }
  \new Lyrics
  \lyricsto melody {
    normal
    \set includeGraceNotes = ##t
    case,
    gra -- ce case,
    after -- grace case,
    \set ignoreMelismata = ##t
    app. case,
```
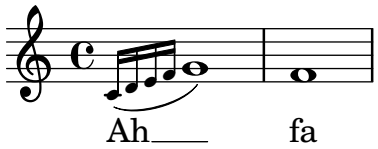
```
    acc. case.
  }
>>
```



Problèmes connus et avertissements

Tout comme pour la propriété `associatedVoice`, la propriété `includeGraceNotes` doit être activée au moins une syllabe avant celle qui viendra s'attacher à la note d'ornement. Dans le cas où cette note se trouve être la première de la pièce, vous devrez recourir à une clause `\with` ou introduire une section `\context` dans le bloc `\layout` :

```
<<
  \new Voice = melody \relative c' {
    \grace { c16( d e f }
    g1) f
  }
  \new Lyrics \with { includeGraceNotes = ##t }
  \lyricsto melody {
    Ah __ fa
  }
>>
```

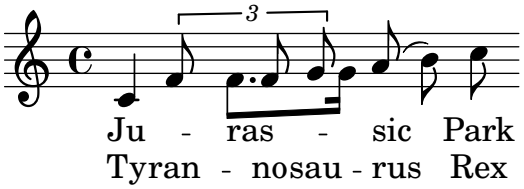


Basculer vers une mélodie alternative

On peut créer des variations plus complexes à partir d'une mélodie à plusieurs voix. Les paroles peuvent suivre l'une ou l'autre des lignes mélodiques, et même basculer de l'une à l'autre si l'on modifie la propriété `associatedVoice`. Dans cet exemple,

```
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c'4
    <<
      \new Voice = "alternative" {
        \voiceOne
        \tuplet 3/2 {
          % show associations clearly.
          \override NoteColumn.force-hshift = -3
          f8 f g
        }
      }
      {
        \voiceTwo
        f8.[ g16]
        \oneVoice
      } >>
```

```
    a8( b) c
  }
  \new Lyrics \lyricsto "lahlah" {
    Ju -- ras -- sic Park
  }
  \new Lyrics \lyricsto "lahlah" {
    % Tricky: need to set associatedVoice
    % one syllable too soon!
    \set associatedVoice = alternative % applies to "ran"
    Ty --
    ran --
    no --
    \set associatedVoice = lahlah % applies to "rus"
    sau -- rus Rex
  }
>>
```



le texte du premier couplet s'aligne de manière habituelle sur la mélodie nommée « lahlah ». Mais le second couplet, tout d'abord rattaché au contexte `lahlah`, bascule sur la mélodie `alternative` pour les syllabes « ran » à « sau » grâce aux lignes

```
\new Lyrics \lyricsto "lahlah" {
  \set associatedVoice = alternative % s'applique à "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = lahlah % s'applique à "rus"
  sau -- rus Rex
}
```

où `alternative` désigne le nom du contexte `Voice` qui contient le triolet.

Notez bien où apparaît la commande `\set associatedVoice` – une syllabe en avance, ce qui est tout à fait correct.

> **Note :** La commande `\set associatedVoice` **doit** intervenir une syllabe *avant* celle qui sera suivie par la bascule. Autrement dit, une modification de la voix associée sera effective une syllabe plus tard que prévu. Il ne s'agit en aucun cas d'une bogue, la raison en est purement technique.

9.3.6 Paroles en fin de partition

Il peut parfois s'avérer opportun d'aligner un seul couplet sur la mélodie et de présenter tous les autres en bloc à la fin du morceau. Ces couplets additionnels peuvent être inclus dans une section `\markup` en dehors du bloc `\score` principal. Vous pourrez noter qu'il existe différentes méthodes pour indiquer les sauts de ligne en mode *markup*. On peut saisir du texte en continu avec `string-lines` et y insérer des `\n` ou couper automatiquement selon la saisie, ou bien utiliser

\wordwrap-string. Dans le cas de formatages différent au sein du bloc, il vaut mieux utiliser une combinaison de \line et \column.

```
melody = \relative {
  \time 2/4
  g'4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

text = \lyricmode {
  \stanza "1."
  À la clai- re fon- tai- ne,
  M'en al- lant pro- me- ner...
}

\score{ <<
  \new Voice = "one" { \melody }
  \new Lyrics \lyricsto "one" \text
>>
  \layout { }
}

\markup \column \string-lines
  "Couplet 2. \n Sous les feuilles d'un chêne \n Je me suis fait sécher..."
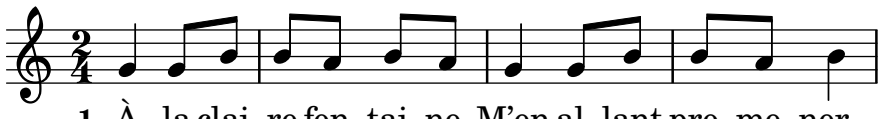
\markup \column \string-lines
  "Couplet 3.
   Chante, rossignol, chante,
   Toi qui as le cœur gai..."

\markup \column {
    \line \italic { Couplet 4. }
    \line { J'ai perdu mon ami }
    \line { Sans l'avoir mérité... }
}

\markup \wordwrap-string "
  Couplet 5.

  Je voudrais que la rose

  Fût encore au rosier..."
```



Couplet 2.
Sous les feuilles d'un chêne
Je me suis fait sécher...

Couplet 3.
Chante, rossignol, chante,
Toi qui as le cœur gai...

Couplet 4.
J'ai perdu mon ami
Sans l'avoir mérité...

Couplet 5.
Je voudrais que la rose
Fût encore au rosier...

9.3.7 Paroles sur plusieurs colonnes en fin de partition

Si les couplets sont vraiment nombreux, il est possible de les imprimer sur plusieurs colonnes. L'exemple suivant vous montre comment procéder pour que le numéro du couplet soit en retrait à gauche, comme c'est traditionnellement le cas.
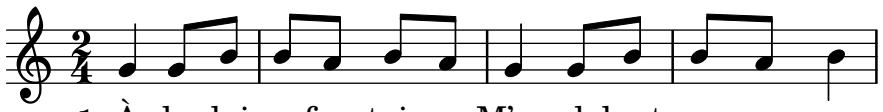
```
melody = \relative {
  \time 2/4
  g'4 g8 b | b a b a |
  g4 g8 b | b a b4 |
}

text = \lyricmode {
  \stanza "1." À la clai- re fon- tai- ne,
  M'en al- lant pro- me- ner...
}

\score{ <<
    \new Voice = "one" { \melody }
    \new Lyrics \lyricsto "one" \text
   >>
  \layout { }
}

\markup {
  \fill-line {
    % décalage par rapport à la marge de gauche
    % peut être supprimé si l'espace sur la page est réduit
    \hspace #0.1
    \column {
      \line { \bold "2."
        \column {
          "Sous les feuilles d'un chêne"
          "Je me suis fait sécher..."
        }
      }
      % ajout d'espace vertical entre les couplets
      \combine \null \vspace #0.1
      \line { \bold "3."
        \column {
          "Chante, rossignol, chante,"
          "Toi qui as le cœur gai..."
```

```
        }
      }
    }
    % ajout d'espace horizontal entre les colonnes
    \hspace #0.1
    \column {
      \line { \bold "4."
        \column {
          "J'ai perdu mon ami"
          "Sans l'avoir mérité..."
        }
      }
      % ajout d'espace vertical entre les couplets
      \combine \null \vspace #0.1
      \line { \bold "5."
        \column {
          "Je voudrais que la rose"
          "Fût encore au rosier..."
        }
      }
    }
    % décalage par rapport à la marge de droite
    % peut être supprimé si l'espace sur la page est réduit
    \hspace #0.1
  }
}
```



1. À la clai- re fon- tai- ne, M'en al- lant pro- me- ner...

2. Sous les feuilles d'un chêne
Je me suis fait sécher...

3. Chante, rossignol, chante,
Toi qui as le cœur gai...

4. J'ai perdu mon ami
Sans l'avoir mérité...

5. Je voudrais que la rose
Fût encore au rosier...

Voir aussi

Référence des propriétés internes : Section "LyricText" dans *Référence des propriétés internes*, Section "StanzaNumber" dans *Référence des propriétés internes*.

9.4 Chansons

9.4.1 Références en matière de chanson

Une chanson se présente la plupart du temps sous la forme de trois portées : une pour la mélodie surmontant un système pianistique pour l'accompagnement ; les paroles du premier couplet s'accolent sous la mélodie. S'il n'y a que deux ou trois couplets, et que cela n'est pas gênant au niveau de l'aspect général, tous peuvent prendre place entre la mélodie et l'accompagnement. Dans le cas contraire, le premier couplet sera imprimé sous la mélodie et les suivants après la partition, sous forme de blocs de texte indépendants.

Tous les éléments qui permettent d'imprimer des chansons sont examinés à différents endroits de la documentation de LilyPond :

- L'agencement des portées est abordé au chapitre Section 6.1 [Gravure des portées], page 248.
- Les spécificités du piano sont abordées au chapitre Chapitre 10 [Instruments utilisant des portées multiples], page 426.
- L'affectation de paroles à une ligne mélodique est abordée au chapitre Section 9.1 [Vue d'ensemble de la musique vocale], page 355.
- Le positionnement des paroles fait l'objet d'une Section 9.2.2 [Positionnement vertical des paroles], page 370.
- La gestion des couplets est abordée dans un Section 9.3 [Couplets], page 388.
- L'harmonisation d'une chanson est souvent indiquée par des noms d'accord en surplomb de la mélodie. Ceci est abordé au chapitre Section 15.2 [Gravure des accords], page 527.
- L'impression de diagrammes d'accord, lorsque l'accompagnement est fait à la guitare, est expliqué à la rubrique « Tablatures sous forme d'étiquette », au chapitre Section 12.1 [Vue d'ensemble des cordes frettées], page 444.

Voir aussi

Manuel d'initiation : Section "Chansons" dans *Manuel d'initiation*.

Manuel de notation : Section 9.3 [Couplets], page 388, Section 15.2 [Gravure des accords], page 527, Section 6.1 [Gravure des portées], page 248, Chapitre 10 [Instruments utilisant des portées multiples], page 426, Section 9.2.2 [Positionnement vertical des paroles], page 370, Section 9.1 [Vue d'ensemble de la musique vocale], page 355.

Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

9.4.2 Feuille de chant

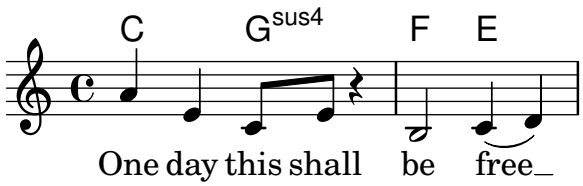
Une simple feuille de chant s'obtient en combinant une partie vocale et son harmonisation. La syntaxe appropriée est expliquée en détails au chapitre Chapitre 15 [Notation des accords], page 522.

Morceaux choisis

Chanson simple

Assembler des noms d'accords, une mélodie et des paroles permet d'obtenir la partition d'une chanson.

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Manuel de notation : Chapitre 15 [Notation des accords], page 522.

9.5 Chorale

Nous allons voir, dans les paragraphes qui suivent, les particularités de la musique chorale, qu'il s'agisse de motet, d'oratorio ou de simple partie de chœur.

9.5.1 Références en matière de chorale

Une partition pour chœur comporte habituellement de deux à quatre portées regroupées dans un `ChoirStaff`. L'accompagnement, s'il y en a un, se présente sous la forme d'un système pianistique – un `PianoStaff` – en dessous du chœur ; il s'agira d'une simple réduction dans le cas d'une œuvre *a capella*. Les notes de chaque pupitre font l'objet d'un contexte `Voice` distinct. Ces contextes `Voice` peuvent se voir groupés ou non sur une même portée.

Les paroles sont traitées dans des contextes `Lyrics` qui viendront se placer tantôt sous la portée, tantôt au-dessus et au-dessous de la portée si elle contient deux voix.

Certaines composantes d'une partition pour chœur sont examinées à différents endroits de la documentation de LilyPond :

- La création pas à pas d'une partition pour chœur se trouve dans le manuel d'initiation, à la rubrique Section "Partition pour chœur à quatre voix mixtes" dans *Manuel d'initiation*. LilyPond dispose aussi d'un canevas automatisé qui simplifie grandement la saisie d'une partition pour chœur SATB disponible à la rubrique Section "Gabarits préprogrammés" dans *Manuel d'initiation*.
- Plusieurs exemples et canevas sont regroupés dans le manuel d'initiation, à la rubrique Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*.
- Des informations détaillées sur les contextes `ChoirStaff` et `PianoStaff` sont disponibles au chapitre Section 6.1.2 [Regroupement de portées], page 250.
- Les formes de notation particulière, telle que celle utilisée en *Sacred Harp* et assimilées, sont abordées au chapitre Section 1.4.3 [Têtes de note à forme variable], page 48.
- Lorsque plusieurs pupitres sont regroupés sur la même portée, les hampes, liaisons, etc. de la voix supérieure sont orientées vers le haut, et inversement pour la voix inférieure. L'utilisation de `\voiceOne` et `\voiceTwo` est expliquée au chapitre Section 5.2.1 [Polyphonie sur une portée], page 227.
- La division temporaire d'un pupitre, ce qui correspond à un passage polyphonique temporaire, est expliquée à la section Section 5.2.1 [Polyphonie sur une portée], page 227.

Commandes prédéfinies

`\oneVoice`, `\voiceOne`, `\voiceTwo`.

Voir aussi

Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*, Section "Partition pour chœur à quatre voix mixtes" dans *Manuel d'initiation*.

Manuel de notation : Section 33.7 [Ordonnancement des contextes], page 779, Section 5.2.1 [Polyphonie sur une portée], page 227, Section 6.1.2 [Regroupement de portées], page 250, Section 1.4.3 [Têtes de note à forme variable], page 48.

Morceaux choisis. . . : Section "Musique vocale" dans *Morceaux choisis*.

Référence des propriétés internes. . . : Section "ChoirStaff" dans *Référence des propriétés internes*, Section "Lyrics" dans *Référence des propriétés internes*, Section "PianoStaff" dans *Référence des propriétés internes*.

9.5.2 Mise en forme d'une partition chorale

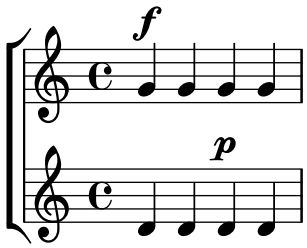
Une partition pour chœur sur quatre portées, avec ou sans accompagnement, présente traditionnellement deux systèmes par page. Selon la taille du papier, vous pourrez être amené à effectuer quelques ajustements aux réglages par défaut, notamment en raison des points suivants :

- La taille des portées a des répercussions sur l'ensemble des éléments de la partition. Voir Section 27.2 [Définition de la taille de portée], page 707.
- La distance séparant les systèmes, les portées et les paroles peut s'ajuster de manière séparée, comme expliqué au chapitre Chapitre 29 [Espacement vertical], page 719.
- La mise en évidence des différentes dimensions permet d'appréhender avec finesse le réglage des variables d'espacement vertical et, pourquoi pas, de faire tenir la partition sur moins de pages, comme l'explique la rubrique Chapitre 31 [Réduction du nombre de pages de la partition], page 752.
- Lorsque le nombre de systèmes varie d'une page à l'autre, il est judicieux de l'indiquer visuellement, en suivant les instructions de la rubrique Section 6.1.4 [Séparation des systèmes], page 257.
- Pour de plus amples détails quant aux propriétés liées au formatage, consultez le chapitre Chapitre 26 [Mise en forme de la page], page 691.

Les indications de nuance se placent traditionnellement sous la portée, ce qui n'est pas le cas en matière de musique vocale dans le but d'éviter toute collision avec les paroles. La commande prédéfinie `\dynamicUp` attachée à un contexte `Voice` permet de positionner les nuances au-dessus de la portée. Dans le cas où il y en aurait plusieurs, cette commande devra apparaître dans chacun des contextes `Voice` qui le requiert. Vous pouvez aussi opter pour la forme développée, comme dans l'exemple ci-dessous, pour que cela s'applique à toutes les portées de la partition – changez `\Score` en `\ChoirStaff` s'il y a d'autres parties que celles du chœur.

```
\score {
  \new ChoirStaff <<
    \new Staff {
      \new Voice {
        \relative { g'4\f g g g }
      }
    }
    \new Staff {
      \new Voice {
        \relative { d'4 d d\p d }
      }
```

```
    }
  >>
  \layout {
    \context {
      \Score
      \override DynamicText.direction = #UP
      \override DynamicLineSpanner.direction = #UP
    }
  }
}
```



Commandes prédéfinies

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Voir aussi

Manuel de notation : Section 27.2 [Définition de la taille de portée], page 707, Chapitre 29 [Espacement vertical], page 719, Section 31.1 [Mise en évidence de l'espacement], page 752, Chapitre 26 [Mise en forme de la page], page 691, Chapitre 27 [Mise en forme de la partition], page 705, Section 31.2 [Modification de l'espacement], page 753, Chapitre 31 [Réduction du nombre de pages de la partition], page 752, Chapitre 28 [Sauts], page 711, Section 6.1.4 [Séparation des systèmes], page 257.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

Morceaux choisis

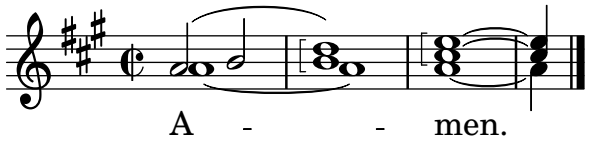
Utilisation d'un crochet pour rendre les divisions plus évidentes

La commande `nonArpeggiato` permet de mettre en évidence les divisions d'un pupitre en l'absence de hampe, comme on le voit régulièrement dans les partitions pour chœur.

```
\include "english.ly"

\score {
  \relative c'' {
    \key a \major
    \time 2/2
    <<
      \new Voice = "upper" <<
        {
          \voiceOne
          a2( b2
          <b d>1\nonArpeggiato)
          <cs e>\nonArpeggiato ~
          <cs e>4
```

```
        \fine
      }
      \addlyrics { \lyricmode { A -- men. } }
    >>
    \new Voice = "lower" {
      \voiceTwo
      a1 ~
      a
      a ~
      a4
      \fine
    }
  >>
 }
}
```



Voir aussi

Manuel de notation : Section 3.3 [Signes d'interprétation sous forme de ligne], page 181.

9.6 Opéras et musiques de scène

Tout ce qui permet d'exécuter un opéra ou une œuvre scénique accompagnée de musique se présente généralement sous l'une ou plusieurs des formes suivantes :

- Un *Conducteur* destiné au chef d'orchestre. Il comprend l'intégralité des parties d'orchestre et des chanteurs, ainsi que des citations du livret pour les passages déclamés.
- Un *matériel d'orchestre*, autrement dit une partition pour chacun des pupitres de l'orchestre ou de l'ensemble instrumental.
- Une *partition de chœur* regroupant toutes les parties vocales avec accompagnement au piano. Cet accompagnement est souvent une réduction d'orchestre où les différents instruments sont indiqués. Les partitions de chœur comprennent parfois des indications de mise en scène ainsi que des extraits du livret.
- Une *partition de choriste* qui ne comprend que les parties vocales – donc sans accompagnement. Elle peut être augmentée du livret.
- Un *livret* contenant l'intégralité des dialogues et le texte des passages chantés. On y trouve aussi très souvent les indications de mise en scène. Bien que LilyPond soit capable de « typographier » un livret, n'oubliez pas qu'il n'y a dans ce cas pas de musique, et que d'autres outils pourraient être mieux appropriés.

La plupart de ce qui est nécessaire à la mise en forme d'un opéra ou d'une musique de scène est disséminé dans la somme documentaire de LilyPond. Nous commencerons par rappeler ces différents éléments, avant que d'en examiner certaines particularités adaptées aux styles opératique et scénique.

9.6.1 Références en matière d'opéra et musique de scène

En plus des formations scéniques ou vocales, la plupart des notions qui suivent peuvent s'appliquer aux orchestres et autres musiques d'ensemble.

- Un conducteur contient un certain nombre de portées et de nombreuses paroles. Les manières d'agencer les portées sont indiquées à la rubrique Section 6.1.2 [Regroupement de portées], page 250, et les façons de les combiner à la rubrique Section 6.1.3 [Imbrication de regroupements de portées], page 254.
- Les portées vides sont la plupart du temps éliminées d'un conducteur ou d'une partition de chœur. La réalisation d'une telle partition – les anglophones la disent « à la française » – est expliquée à la rubrique Section 6.2.3 [Masquage de portées], page 265.
- La génération d'un matériel d'orchestre fait l'objet de la rubrique Section 6.3 [Écriture de parties séparées], page 270. D'autres parties du chapitre consacré à la notation spécialisée vous seront utiles selon l'orchestration de la pièce. Tous les instruments ne sont pas accordés pareil ; vous trouverez des informations à ce sujet à la rubrique Section 1.3.4 [Instruments transpositeurs], page 30.
- Lorsque le nombre de systèmes varie d'une page à l'autre, il peut être judicieux de les mettre en évidence, en suivant les indications de la rubrique Section 6.1.4 [Séparation des systèmes], page 257.
- Les différentes propriétés impliquées dans la mise en page sont répertoriées au chapitre Chapitre 26 [Mise en forme de la page], page 691.
- L'insertion de dialogues et d'indications de mise en scène peuvent se réaliser à l'aide de *markups*, en suivant les directives fournies aux chapitres Section 21.4 [Notes de bas de page], page 629, et Chapitre 8 [Texte], page 317. Les indications de mise en scène peuvent s'intercaler entre deux blocs `\score` selon les préceptes de la rubrique Section 8.1.6 [Texte indépendant], page 328.

Voir aussi

Glossaire musicologique : Section "Partition à la française" dans *Glossaire*, Section "Frenched staves" dans *Glossaire*, Section "instrument transpositeur" dans *Glossaire*.

Manuel de notation : Section 8.1 [Ajout de texte], page 317, Section 6.3 [Écriture de parties séparées], page 270, Section 6.1.3 [Imbrication de regroupements de portées], page 254, Section 1.3.4 [Instruments transpositeurs], page 30, Section 6.2.3 [Masquage de portées], page 265, Chapitre 26 [Mise en forme de la page], page 691, Section 21.4 [Notes de bas de page], page 629, Section 6.1.2 [Regroupement de portées], page 250, Section 6.1.4 [Séparation des systèmes], page 257, Section 1.2.2 [Transposition], page 14.

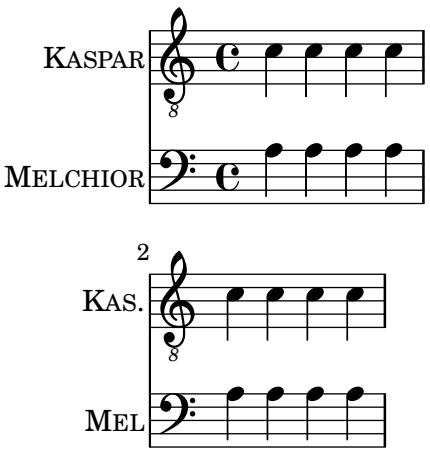
Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

9.6.2 Indication du rôle

Lorsqu'un rôle est distribué sur une portée spécifique, vous pouvez l'indiquer en regard de cette portée :
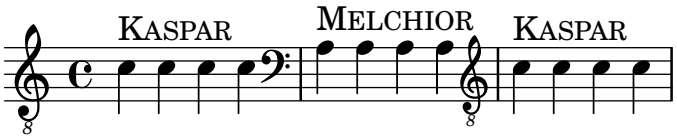
```
\score {
  <<
    \new Staff {
      \set Staff.vocalName = \markup \smallCaps Kaspar
      \set Staff.shortVocalName = \markup \smallCaps Kas.
      \relative {
        \clef "G_8"
        c'4 c c c
        \break
        c4 c c c
      }
    }
    \new Staff {
```

```
      \set Staff.vocalName = \markup \smallCaps Melchior
      \set Staff.shortVocalName = \markup \smallCaps Mel
      \clef "bass"
      \relative {
        a4 a a a
        a4 a a a
      }
    }
  >>
}
```



Lorsque la même portée sert à plusieurs personnages, leur nom est généralement imprimé en surplomb de la portée, à chaque changement de rôle. L'utilisation d'un *markup* – dans une fonte réservée à cet effet – vous permettra de générer ces indications :

```
\relative c' {
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
  \clef "bass"
  a4^\markup \fontsize #1 \smallCaps Melchior
  a a a
  \clef "G_8"
  c4^\markup \fontsize #1 \smallCaps Kaspar
  c c c
}
```
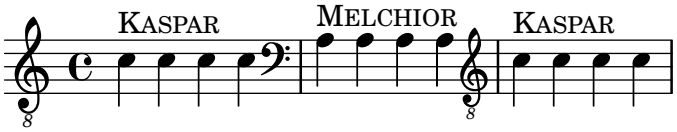


Dans le cas où les changements de personnage se multiplient, il peut s'avérer pratique d'affecter à des variables la définition de chacun des rôles afin de simplifier la gestion des différentes interventions de l'un ou de l'autre.

```
kaspar = {
  \clef "G_8"
  \set Staff.shortVocalName = "Kas."
  \set Staff.midiInstrument = "voice oohs"
  <>^\markup \smallCaps "Kaspar"
}

melchior = {
  \clef "bass"
  \set Staff.shortVocalName = "Mel."
  \set Staff.midiInstrument = "choir aahs"
  <>^\markup \smallCaps "Melchior"
}

\relative c' {
  \kaspar
  c4 c c c
  \melchior
  a4 a a a
  \kaspar
  c4 c c c
}
```



Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section A.1 [Commandes pour *markup*], page 837, Chapitre 8 [Texte], page 317.

9.6.3 Citation-repère

Les citations d'instruments insérées dans les parties vocales, les partitions de chœur ou les partitions d'un pupitre permettent d'indiquer ce qui se passe ailleurs juste avant une entrée. On les retrouve souvent dans la réduction pour piano, ce qui fournit au chef de chœur de précieuses indications sur qui joue quoi, lorsqu'il ne dispose pas d'un conducteur en bonne et due forme.

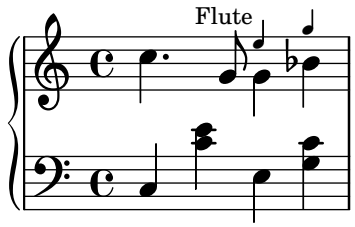
Les mécanismes de base permettant d'insérer des citations sont expliqués en détail aux rubriques Section 6.3.2 [Citation d'autres voix], page 274, et Section 6.3.3 [Mise en forme d'une citation], page 277. Dans le cas où les citations concernent différents instruments, faire mention de celui qui intervient devient une nécessité ; voici une illustration de la manière de procéder en pareil cas :

```
flute = \relative {
  s4 s4 e'' g
}
\addQuote "flute" { \flute }

pianoRH = \relative {
  c''4. g8
  % position name of cue-ing instrument just before the cue notes,
```

```
  % and above the staff
  <>^\markup { \right-align { \tiny "Flute" } }
  \cueDuring "flute" #UP { g4 bes4 }
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  \new PianoStaff <<
    \new Staff {
      \pianoRH
    }
    \new Staff {
      \clef "bass"
      \pianoLH
    }
  >>
}
```



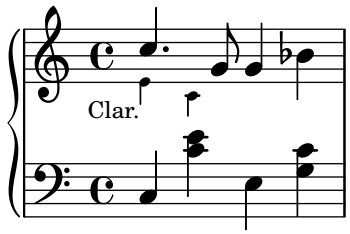
La citation peut concerner un instrument transpositeur, auquel cas il faudra mentionner sa tonalité dans sa définition, afin que ses hauteurs soient automatiquement converties dans la réplique. Ceci est illustré par l'exemple ci-dessous, dans lequel il est fait appel à une clarinette en si bémol. Dans la mesure où les notes citées se trouvent vers le bas de la portée, nous affectons un `DOWN` à la commande `\cueDuring`, de telle sorte que les hampes aillent vers le bas et que le nom de l'instrument cité soit en dessous de la portée.

```
clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

pianoRH = \relative c'' {
  \transposition c'
  % position name of cue-ing instrument below the staff
  <>_\markup { \right-align { \tiny "Clar." } }
  \cueDuring "clarinet" #DOWN { c4. g8 }
  g4 bes4
}
pianoLH = \relative { c4 <c' e> e, <g c> }

\score {
  <<
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
```

```
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Il est clair, au vu de ces deux exemples, que multiplier le nombre de citations dans une partition vocale demande un travail fastidieux et que relire la partie de piano deviendra vite un cauchemar. Vous pouvez néanmoins, comme l'illustre l'exemple suivant, définir une fonction musicale dans le but de vous épargner de la saisie tout en améliorant la lisibilité des notes du piano.

Morceaux choisis

Indication de l'instrument cité dans l'accompagnement d'une partition pour chœur

Lorsque le nombre d'instruments cités dans la réduction pour piano se multiplie, vous pourriez avoir intérêt à créer votre propre fonction pour gérer ces repères. La fonction musicale `\cueWhile` prend quatre arguments : la musique d'où provient la citation, telle que définie par `\addQuote`, le nom qui sera mentionné en regard de cette citation, son positionnement – `UP` ou `DOWN` selon qu'il sera attribué à `\voiceOne` et placé au-dessus ou `\voiceTwo` et placé en dessous – et enfin la musique du piano qui interviendra en parallèle. Le nom de l'instrument en question viendra s'aligner sur la gauche de la citation. Bien que vous puissiez effectuer plusieurs citations, elle ne peuvent être simultanées.

```
cueWhile =
#(define-music-function
   (instrument name dir music)
   (string? string? ly:dir? ly:music?)
   #{
     \cueDuring $instrument #dir {
       \once \override TextScript.self-alignment-X = #RIGHT
       \once \override TextScript.direction = $dir
       <>-\markup { \tiny #name }
       $music
     }
   #})

flute = \relative c'' {
  \transposition c'
  s4 s4 e g
}
```
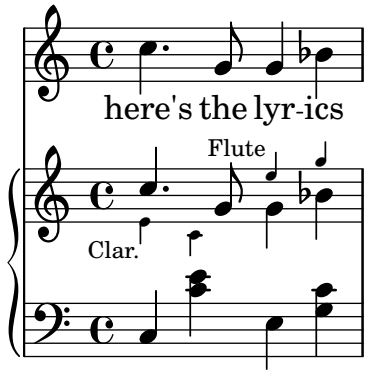
```
\addQuote "flute" { \flute }

clarinet = \relative c' {
  \transposition bes
  fis4 d d c
}
\addQuote "clarinet" { \clarinet }

singer = \relative c'' { c4. g8 g4 bes4 }
words = \lyricmode { here's the lyr -- ics }

pianoRH = \relative c'' {
  \transposition c'
  \cueWhile "clarinet" "Clar." #DOWN { c4. g8 }
  \cueWhile "flute" "Flute" #UP { g4 bes4 }
}
pianoLH = \relative c { c4 <c' e> e, <g c> }

\score {
  <<
    \new Staff {
      \new Voice = "singer" {
        \singer
      }
    }
    \new Lyrics {
      \lyricsto "singer"
      \words
    }
    \new PianoStaff <<
      \new Staff {
        \new Voice {
          \pianoRH
        }
      }
      \new Staff {
        \clef "bass"
        \pianoLH
      }
    >>
  >>
}
```



Voir aussi

Glossaire musicologique : Section “cue-notes” dans *Glossaire*.

Manuel de notation : Section 36.9 [Alignement des objets], page 820, Section 6.3.2 [Citation d'autres voix], page 274, Section 36.1 [Direction et positionnement], page 801, Section 6.3.3 [Mise en forme d'une citation], page 277, Section 22.3 [Utilisation de fonctions musicales], page 658.

Morceaux choisis : Section “Musique vocale” dans *Morceaux choisis*.

Référence des propriétés internes : Section “CueVoice” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

`\cueDuring` crée automatiquement un contexte `CueVoice` qui accueillera toutes les notes répliquées. Il est par conséquent impossible de faire se superposer des citations à l'aide de la technique simplifiée telle que nous venons de le voir. La superposition de fragments cités requiert que les contextes `CueVoice` soient explicitement déclarés, ainsi que l'utilisation de la commande `\quoteDuring` pour extraire et insérer les notes répliquées.

9.6.4 Musique parlée
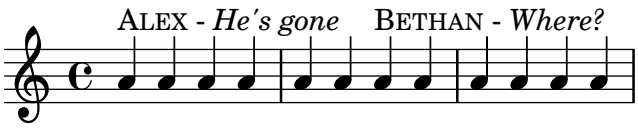
Le *parlato* – ou *Sprechgesang* pour les germanistes – est du texte scandé en rythme, mais sans hauteurs définies ; il est indiqué par des notes en croix, à l'instar des percussions – voir Section 1.4.1 [Têtes de note spécifiques], page 45.

9.6.5 Dialogue et musique

Les dialogues que l'on ajoute à la musique sont traditionnellement imprimés en italique au-dessus des portées, au moment même où ils surviennent.

Une courte intervention peut se formuler à l'aide d'un simple *markup* :

```
\relative {
  a'4^\markup { \smallCaps { Alex - } \italic { He's gone } } a a a
  a4 a a^\markup { \smallCaps { Bethan - } \italic Where? } a
  a4 a a a
}
```



Une intervention un peu plus longue peut nécessiter d'étirer la musique de telle sorte que le texte ait suffisamment de place. LilyPond ne disposant d'aucun mécanisme permettant d'automatiser l'étirement, vous devrez probablement ajuster vous même la mise en forme.

Dans le cas d'une phrase entière ou de passages relativement denses, le recours à un contexte Lyrics peut donner de meilleurs résultats. Le contexte Lyrics en question ne doit être rattaché à aucune voix musicale ; chaque fragment de dialogue devra donc comporter des durées explicites. Lorsque les dialogues comportent des pauses, le dernier mot devra être séparé du reste et les durées individualisées pour obtenir un espacement harmonieux de la musique.

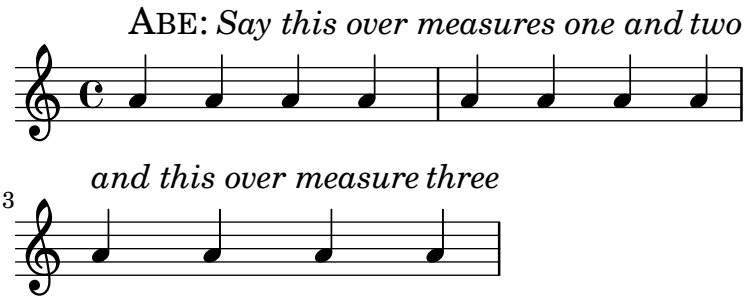
Des dialogues qui s'étendent sur plus d'une ligne vous obligeront à insérer des \break et ajuster leur placement pour éviter qu'ils ne débordent dans la marge droite. Le dernier mot de la dernière mesure d'une ligne doit être saisi sur une ligne à part.

Voici une illustration de tout ce que nous venons de voir :

```
music = \relative {
  \*3 { a'4 a a a }
}

dialogue = \lyricmode {
  \markup {
    \fontsize #1 \upright \smallCaps Abe:
    "Say this over measures one and"
  }4*7
  "two"4 |
  \break
  "and this over measure"4*3
  "three"4 |
}

\score {
  <<
    \new Lyrics \with {
      \override LyricText.font-shape = #'italic
      \override LyricText.self-alignment-X = #LEFT
    }
    { \dialogue }
    \new Staff {
      \new Voice { \music }
    }
  >>
}
```



Voir aussi

Manuel de notation : Section 9.1.5 [Durée explicite des syllabes], page 362, Chapitre 8 [Texte], page 317.

Référence des propriétés internes : Section "LyricText" dans *Référence des propriétés internes*.

9.7 Chants liturgiques

Selon les chapelles, la mise en forme des cantiques, psaumes et hymnes répond à des canons bien établis. Bien que différents de par leur présentation, nous verrons dans ce qui suit que les problèmes qui surviennent en matière de typographie se ressemblent, quelle que soit l'obédience.

9.7.1 Références en matière de chant liturgique

La présentation du plain-chant et du grégorien selon différents styles est abordée au chapitre Chapitre 17 [Notations anciennes], page 552.

Voir aussi

Manuel de notation : Chapitre 17 [Notations anciennes], page 552.

Morceaux choisis : Section "Musique vocale" dans *Morceaux choisis*.

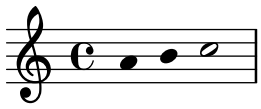
9.7.2 Cantiques et hymnes

La mise en forme contemporaine de cantiques utilise à la fois la notation moderne et un certain nombre d'éléments propres aux notations anciennes. Nous allons examiner quelques uns de ces éléments et la méthode consacrée pour les mettre en œuvre.

Les cantiques utilisent souvent des noires dépourvues de hampe pour indiquer les hauteurs ; le rythme de la mélodie est donné par le rythme et l'accentuation des paroles elles-mêmes.

```
stemOff = { \hide Staff.Stem }

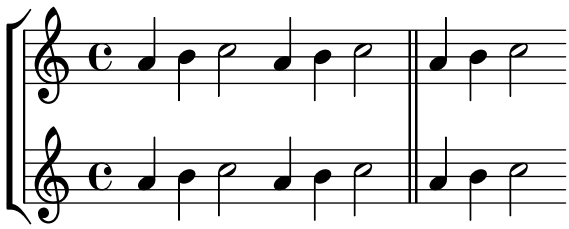
\relative c' {
  \stemOff
  a'4 b c2 |
}
```



Les barres de mesure sont absentes dans la plupart des cas ; celles que vous rencontrerez seront raccourcies ou en pointillé, dans le but d'indiquer une « respiration ». Pour obtenir un chant non mesuré, voir Section 2.3.5 [Musique sans métrique], page 94. Pour préserver tous les effets d'une métrique tout en désactivant les barres de mesure automatiques, il faudra régler `measureBarType` à `'()`. Sans barre de mesure à intervalle régulier, vous devrez gérer manuellement l'espacement horizontal – voir Section 28.1 [Sauts de ligne], page 711.

```
\score {
  \new StaffGroup <<
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 | \section
        a4 b c2 |
      }
    }
    \new Staff {
      \relative {
        a'4 b c2 |
        a4 b c2 | \section
        a4 b c2 |
      }
```
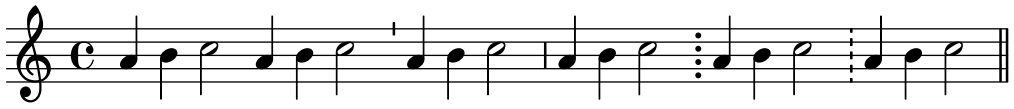
```
    }
  >>
  \layout {
    \context {
      \Staff
      measureBarType = #'()
      forbidBreakBetweenBarLines = ##f
    }
  }
}
```



L'absence de barre de mesure peut ne concerner que certaines portées – voir Section 33.4 [Modification des greffons de contexte], page 769.

Dans la mélodie d'un cantique, les silences ou pauses s'indiquent à l'aide de barres de mesure spécifiques :
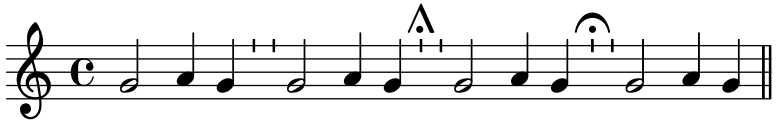
```
\relative a' {
  a4
  \cadenzaOn
  b c2
  a4 b c2
  \bar "'"
  a4 b c2
  \bar ","
  a4 b c2
  \bar ";"
  a4 b c2
  \bar "!"
  a4 b c2
  \bar "||"
}
```



Vous pouvez accessoirement, bien qu'il s'agisse de notation moderne, emprunter au grégorien des indications de pause et silence.
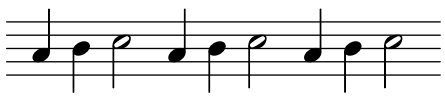
```
\score {
  \relative {
    g'2 a4 g
    \divisioMinima
    g2 a4 g
    \divisioMaior
    g2 a4 g
    \divisioMaxima
```

```
    g2 a4 g
    \finalis
  }
  \layout {
    \context {
      \Staff
      \remove Bar_engraver
      \consists Divisio_engraver
      caesuraType = #'((breath . chantquarterbar))
    }
  }
}
```



De nombreux cantiques sont dépourvus de métrique, voire même de clef.

```
\score {
  \new Staff {
    \relative {
      a'4 b c2 |
      a4 b c2 |
      a4 b c2 |
    }
  }
  \layout {
    \context {
      \Staff
      \remove Time_signature_engraver
      \remove Clef_engraver
      measureBarType = #'()
      forbidBreakBetweenBarLines = ##f
    }
  }
}
```



L'une des traditions anglicanes est de chanter les psaumes sur la base d'un fragment de sept mesures – forme *single* ou simple – ou de deux fragments toujours de sept mesures – forme *double*. Chaque fragment est divisé en deux parties correspondant aux deux moitiés de chaque verset et généralement séparées par une double barre. Il n'est fait usage que de rondes et de blanches, et la première mesure de chaque moitié contient un simple accord de rondes. Il s'agit donc des notes correspondant au « récitatif ». Ces cantiques sont traditionnellement centrés sur la page.

```
SopranoMusic = \relative {
  g'1 | c2 b | a1 |
  a1 | d2 c | c b | c1 |
}

AltoMusic = \relative {
  e'1 | g2 g | f1 |
  f1 | f2 e | d d | e1 |
}

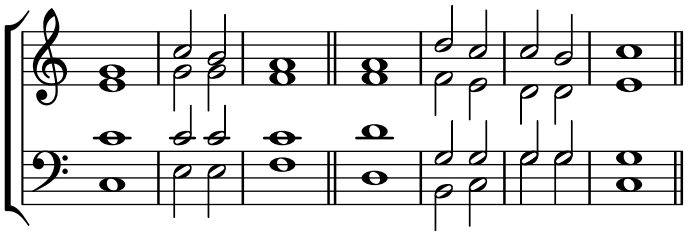
TenorMusic = \relative {
  c'1 | c2 c | c1 |
  d1 | g,2 g | g g | g1 |
}

BassMusic =  \relative {
  c1 | e2 e | f1 |
  d1 | b2 c | g' g | c,1 |
}

global = {
  \time 2/2
  \skip 1*3 \section
  \skip 1*4 \fine
}

% Use markup to center the chant on the page
\markup {
  \fill-line {
    \score {  % centered
      <<
        \new ChoirStaff <<
          \new Staff <<
            \global
            \clef "treble"
            \new Voice = "Soprano" <<
              \voiceOne
              \SopranoMusic
            >>
            \new Voice = "Alto" <<
              \voiceTwo
              \AltoMusic
            >>
          >>
          \new Staff <<
            \clef "bass"
            \global
            \new Voice = "Tenor" <<
              \voiceOne
              \TenorMusic
            >>
            \new Voice = "Bass" <<
```

```
            \voiceTwo
            \BassMusic
          >>
        >>
      >>
    >>
    \layout {
      \context {
        \Score
        \override SpacingSpanner.base-shortest-duration =
          \musicLength 2
        fineBarType = "||"
      }
      \context {
        \Staff
        \remove Time_signature_engraver
      }
    }
  } % End score
}
} % End markup
```



D'autres approches d'une telle mise en forme font l'objet du premier des exemples qui suivent.

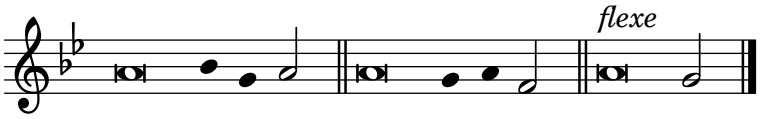
Morceaux choisis

Notation pour psalmodie

Ce style de notation permet d'indiquer la mélodie d'une psalmodie lorsque les strophes sont de longueur inégale.

```
stemOff = \hide Staff.Stem
stemOn  = \undo \stemOff

\score {
  \new Staff \with { \remove "Time_signature_engraver" }
  {
    \key g \minor
    \cadenzaOn
    \stemOff a'\breve bes'4 g'4
    \stemOn a'2 \section
    \stemOff a'\breve g'4 a'4
    \stemOn f'2 \section
    \stemOff a'\breve^\markup { \italic flexe }
    \stemOn g'2 \fine
  }
}
```



Cantiques et autres textes liturgiques peuvent être mis en forme avec une grande liberté et parfois emprunter des éléments de notation ancienne. Le texte apparaît souvent sous la mélodie, les mots alors alignés sur les notes. En pareil cas, les notes sont espacées selon les syllabes et non leur durée.

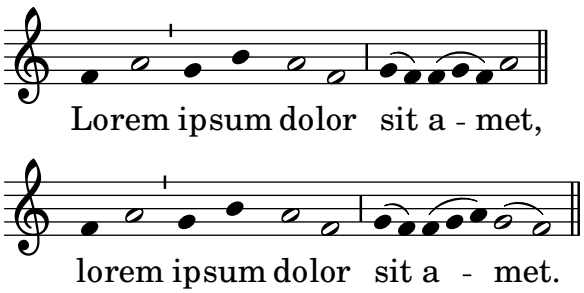
Exemples de notation ancienne – transcription moderne de musique grégorienne

Voici comment vous pourriez transcrire du grégorien. Pour mémoire, il n'y a en grégorien ni découpage en mesure, ni hampe ; seules sont utilisées des têtes de note blanches ou noires, ainsi que des signes spécifiques permettant d'indiquer des silences de différentes durées.

```
chant = \relative c' {
  \set Score.timing = ##f
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g f) a2 \finalis \break
  f4 a2 \divisioMinima
  g4 b a2 f2 \divisioMaior
  g4( f) f( g a) g2( f) \finalis
}

verba = \lyricmode {
  Lo -- rem ip -- sum do -- lor sit a -- met,
  lo -- rem ip -- sum do -- lor sit a -- met.
}

\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "melody" \chant
    \new GregorianTranscriptionLyrics = "one" \lyricsto melody \verba
  >>
}
```



Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*, Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*.

Manuel de notation : Section 2.5.1 [Barres de mesure], page 123, Section 33.4 [Modification des greffons de contexte], page 769, Section 2.3.5 [Musique sans métrique], page 94, Chapitre 17 [Notations anciennes], page 552, Section 17.4 [Typographie du chant grégorien], page 569, Section 36.7 [Visibilité des objets], page 812.

9.7.3 Psalmodie

Les versets d'un psaume anglican sont habituellement centrées sous la mélodie.

Dans le cas d'un chant simple, les sept mesures qui le composent sont répétées pour chaque verset. Dans le cas d'un chant double, les quatorze mesures se répètent par couple de versets. Des marques insérées dans le texte indiquent comment il s'articule par rapport à la mélodie. Chaque verset est séparé en deux, et la rupture est indiquée par un caractère deux points (:) correspondant à la double barre de la mélodie. Le texte précédant les deux points se chante sur les trois premières mesures, celui qui suit sur les quatre dernières mesures.

De simples barres verticales – remplacées par des virgules inversées dans certains psautiers – représentent les barres de mesures portées sur la mélodie. En mode *markup*, ces barres s'obtiennent en saisissant le même caractère | qui sert pour les contrôles de mesure.

```
\markup {
  \fill-line {
    \column {
      \line { O come let us sing | unto the | Lord : let }
      \line { us heartily rejoice \bold \italic in the | strength of | our }
      \line { sal- | -vation. }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

Vous pourriez tout à fait utiliser d'autres symboles disponibles au travers des glyphes de la fonte `fetaMusic` – voir le chapitre Section 8.3 [Fontes], page 347, pour plus de détails.

```
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \line { O come let us sing \tick unto the \tick Lord : let }
      \line { us heartily rejoice \bold \italic in the \tick strength of \tick our }
      \line { sal \tick vation. }
    }
  }
}
```

O come let us sing ' unto the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Lorsqu'une mesure ne comporte qu'une ronde, le texte correspondant à cette mesure est chanté sur cette même note, selon le rythme naturel de la phrase. Lorsque la mesure comporte deux notes, celles-ci correspondent en général à une ou deux syllabes ; dans le cas contraire, le changement de note est indiqué par un point.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \line { O come let us sing \tick unto \dot the \tick Lord : let }
      \line { us heartily rejoice \bold \italic in the \tick strength of \tick our }
      \line { sal \tick vation. }
    }
  }
}
```

O come let us sing ' unto • the ' Lord : let
us heartily rejoice in the ' strength of ' our
sal ' vation.

Certains psautiers font apparaître, pour indiquer une césure, une astérisque au lieu d'une virgule, ainsi que des caractères gras pour les syllabes accentuées ou allongées.

```
dot = \markup {
  \raise #0.7 \musicglyph "dots.dot"
}
tick = \markup {
  \raise #1 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
      \line { Today if ye will hear his voice * }
      \line { \concat { \bold hard en } |
              not your | hearts : as \bold \italic in the pro- }
      \line { vocation * and as \bold \italic in the \bold day of tempt- | }
      \line { -ation | \bold \italic in the | wilderness. }
    }
  }
}
```

Today if ye will hear his voice *
harden | not your | hearts : as **in** the pro-
vocation * and as **in** the **day** of tempt- |
-ation | in the | wilderness.

D'autres psautiers indiquent une syllabe accentuée en la surchargeant d'un accent.

```
tick = \markup {
  \raise #2 \fontsize #-5 \musicglyph "scripts.rvarcomma"
}
\markup {
  \fill-line {
    \column {
```

```
      \line { O come let us \concat { si \combine \tick ng } |
              unto the | Lord : let }
      \line { us heartily \concat { rejo \combine \tick ice }
              in the | strength of | our }
      \line { sal- | -vation. }
    }
  }
}
```

O come let us sing | unto the | Lord : let
us heartily rejoice in the | strength of | our
sal- | -vation.

L'utilisation du mode *markup* pour centrer le texte et agencer les lignes est abordée en détails au chapitre Section 8.2 [Mise en forme du texte], page 329.

La plupart de ces éléments sont regroupés dans l'un des versets du modèle présenté à la rubrique Section "Psalmodie" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Modèles pour ensemble vocal" dans *Manuel d'initiation*, Section "Psalmodie" dans *Manuel d'initiation*.

Manuel de notation : Section 8.3 [Fontes], page 347, Section 8.2 [Mise en forme du texte], page 329.

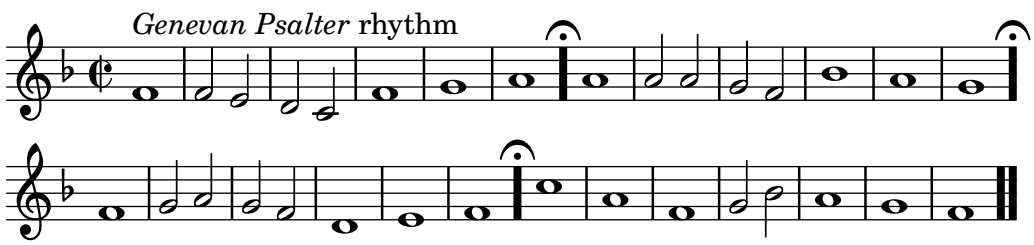
9.7.4 Barre de phrase en musique liturgique
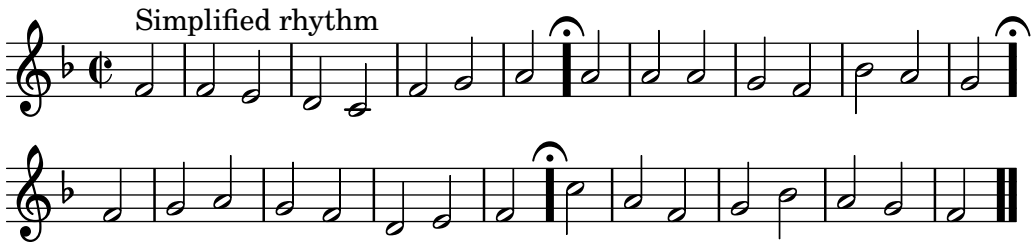
La commande `\caesura` peut être configurée de sorte à créer des barres de phrase qui interagissent correctement avec les autres barres automatiques – voir Section 2.5.2 [Barres de mesure automatiques], page 133.

Dans les exemples suivants, le code source de la mélodie dans `old-hundredth-example.ly` utilise `\caesura` pour séparer les lignes poétiques et `\fine` à la fin. Il est aussi dépourvu des commandes `\fermata` ou `\bar` ; ces symboles apparaissent en raison de la configuration de la césure.

The Boston Handel and Haydn Society Collection of Church Music (1830) positionne une barre épaisse à la fin de chaque phrase. Dans cet hymne, il y a aussi des points d'orgue sur les barres de mesure :

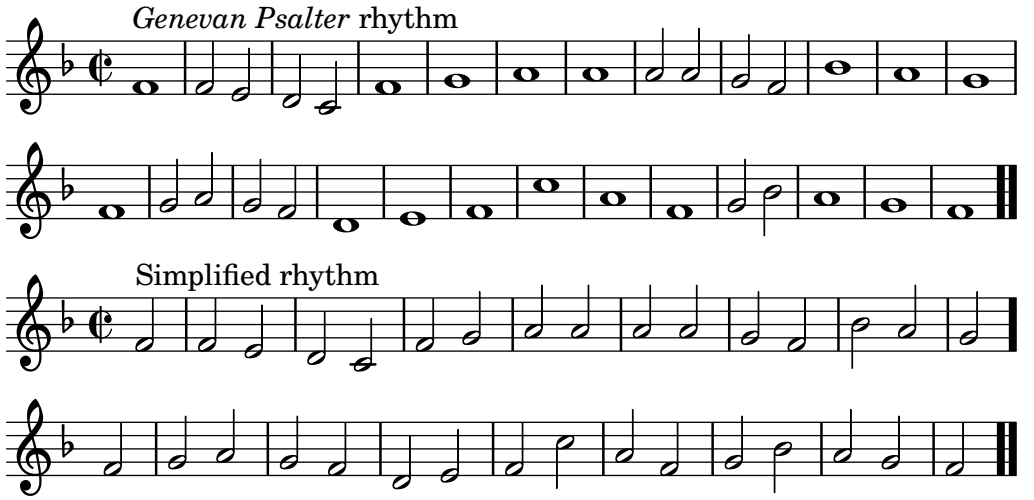
```
\layout {
  \context {
    \Score
    caesuraType = #'((bar-line . ".")
                     (scripts . (fermata)))
    fineBarType = ".."
  }
}
\include "old-hundredth-example.ly"
```





J.S. James, dans *Original Sacred Harp* (1911), dispose une barre épaisse lorsqu'un saut de ligne intervient en cours de mesure à la fin d'une phrase.

```
\layout {
  \context {
    \Score
    caesuraType = #'((underlying-bar-line . "x-."))
    fineBarType = ".."
  }
}
\include "old-hundredth-example.ly"
```



9.7.5 Mesure incomplète et musique liturgique

Il arrive fréquemment que les chants liturgiques comportent des mesures incomplètes aussi bien en début qu'en fin de ligne, de telle sorte qu'à une portée corresponde une ligne de texte. Ceci requiert donc l'utilisation de la commande `\partial` en début de partition et d'une barre de mesure à la fin de chaque ligne.

Modèle pour cantique

Le code ci-dessous illustre la manière d'agencer un cantique liturgique dans lequel chaque ligne débute et se termine par une mesure incomplète. Vous noterez par ailleurs l'affichage des paroles indépendamment de la musique.

```
Timeline = {
  \time 4/4
  \tempo 4=96
  \partial 2
  s2 | s1 | s2 \breathe s2 | s1 | s2 \caesura \break
  s2 | s1 | s2 \breathe s2 | s1 | s2 \fine
}
```

```
SopranoMusic = \relative g' {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

AltoMusic = \relative c' {
  d4 d | d d d d | d d d d | d d d d | d2
  d4 d | d d d d | d d d d | d d d d | d2
}

TenorMusic = \relative a {
  b4 b | b b b b | b b b b | b b b b | b2
  b4 b | b b b b | b b b b | b b b b | b2
}

BassMusic = \relative g {
  g4 g | g g g g | g g g g | g g g g | g2
  g4 g | g g g g | g g g g | g g g g | g2
}

global = {
 \key g \major
}

\score {  % Start score
  \new PianoStaff <<  % Start pianostaff
    \new Staff <<  % Start Staff = RH
      \global
      \clef "treble"
      \new Voice = "Soprano" <<  % Start Voice = "Soprano"
        \Timeline
        \voiceOne
        \SopranoMusic
      >>  % End Voice = "Soprano"
      \new Voice = "Alto" <<  % Start Voice = "Alto"
        \Timeline
        \voiceTwo
        \AltoMusic
      >>  % End Voice = "Alto"
    >>  % End Staff = RH

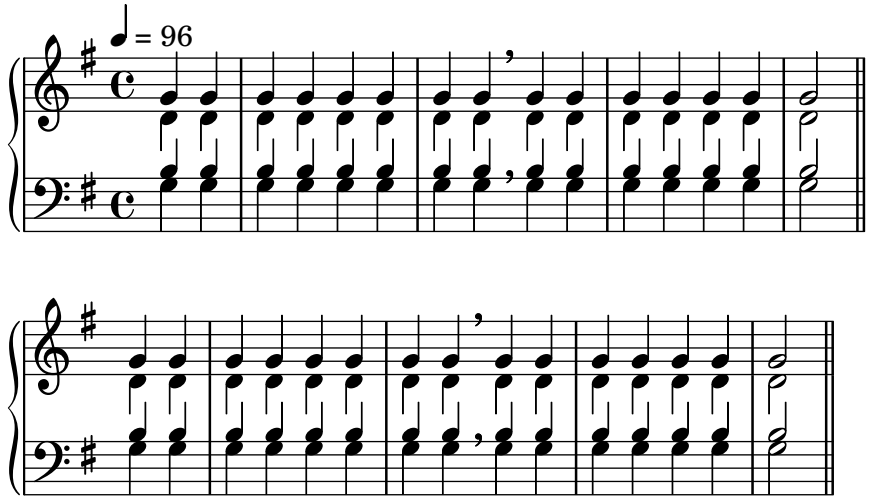
    \new Staff <<  % Start Staff = LH
      \global
      \clef "bass"
      \new Voice = "Tenor" <<  % Start Voice = "Tenor"
        \Timeline
        \voiceOne
        \TenorMusic
      >>  % End Voice = "Tenor"
      \new Voice = "Bass" <<  % Start Voice = "Bass"
        \Timeline
        \voiceTwo
```

```
        \BassMusic
      >>  % End Voice = "Bass"
    >>  % End Staff = LH
  >>  % End pianostaff
}  % End score

\markup \fill-line {
  \left-column {
    "This is line one of the first verse"
    "This is line two of the same"
    \null
    "And here's line one of the second verse"
    "And the next line of the same"
  }
}

\layout {
  \context {
    \Score
    caesuraType = #'((bar-line . "||"))
    fineBarType = "||"
  }
}

\paper {  % Start paper block
  indent = 0         % don't indent first system
  line-width = 130   % shorten line length to suit music
  tagline = ##f      % Don't print tag line, can be removed
}  % End paper block
```



This is line one of the first verse
This is line two of the same

And here's line one of the second verse
And the next line of the same
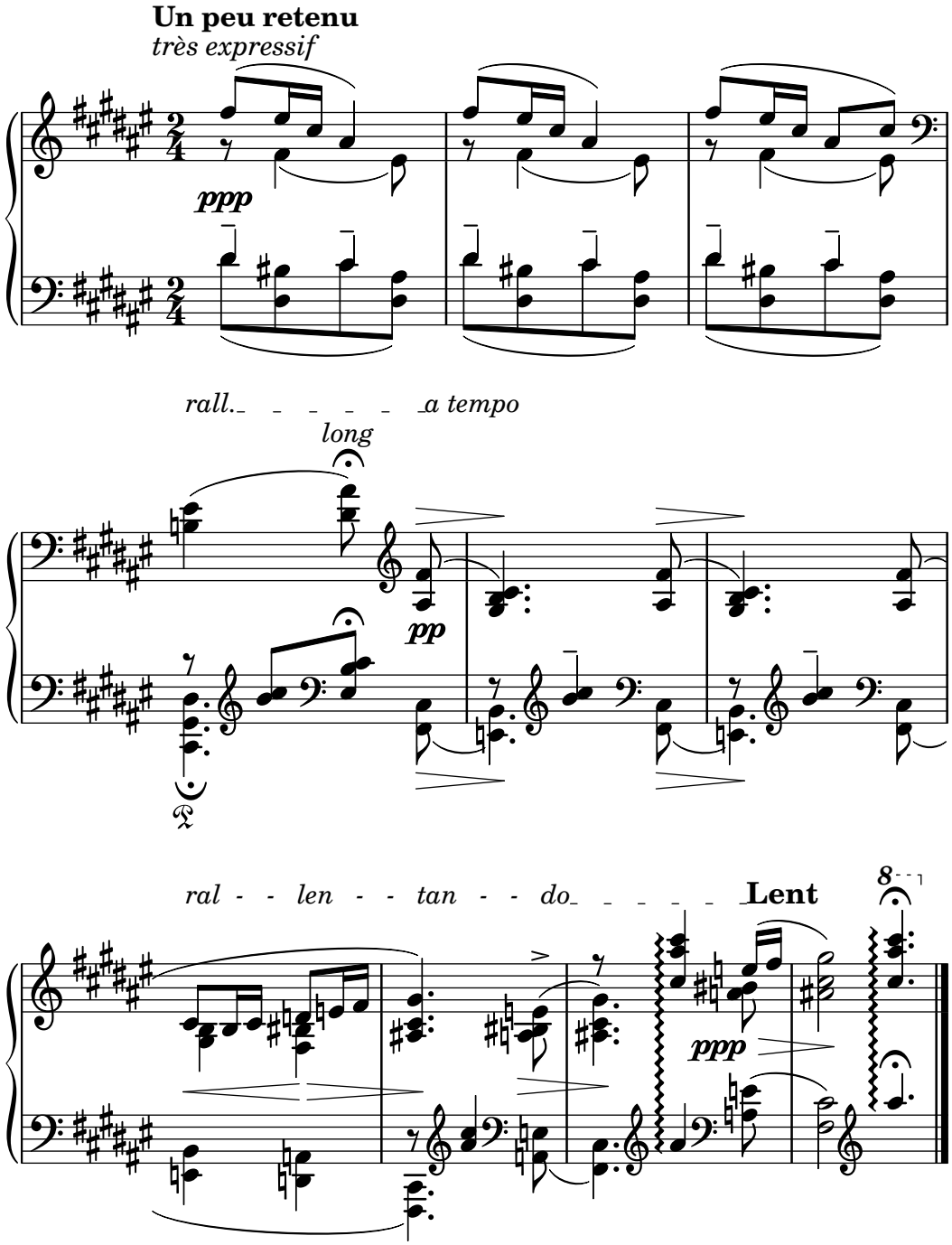
9.8 Musique vocale ancienne

LilyPond prend en charge la musique vocale ancienne. Elle est abordée en détails au chapitre Chapitre 17 [Notations anciennes], page 552.

Voir aussi

Manuel de notation : Chapitre 17 [Notations anciennes], page 552.

10 Instruments utilisant des portées multiples



Ce chapitre traite des différents aspects de la notation que l'on rencontre particulièrement avec les instruments qui ont recours à plusieurs portées, tels que ceux disposant de claviers, la harpe ou le vibraphone. Pour les besoins du discours, et pour simplifier, nous parlerons ici de « clavier » bien que le recours à des portées multiples concerne aussi des instruments qui en sont dépourvus.

10.1 Vue d'ensemble des claviers

Nous allons examiner ici les problèmes qui peuvent survenir en matière de notation pour la plupart des instrument à cordes multiples.

10.1.1 Généralités sur les instruments à clavier

La notation pour instrument à clavier est en règle générale présentée sous la forme d'un système pour piano, autrement dit deux portées normales ou plus réunies par une accolade. Cette notation sert également à la harpe ou à d'autres instruments à clés. L'organiste, quant à lui, lira une partition composée de deux portées au sein d'un système pianistique auquel vient s'adjoindre une portée normale pour le pédalier.

Les portées sont largement autonomes, mais il arrive que des voix passent de l'une à l'autre. Le contexte `PianoStaff` est précisément conçu pour gérer la notation spécifique au piano et autres instruments à clavier, notamment ces croisements.

Certaines particularités de la notation pour claviers sont abordées dans d'autres chapitres :

- Les claviers ont régulièrement recours à plusieurs voix dont le nombre peut varier. Voir à ce sujet Section 5.2.3 [Résolution des collisions], page 233.
- On peut écrire la musique pour claviers de façon parallèle, comme l'explique Section 5.2.6 [Saisie de musique en parallèle], page 244.
- Les nuances peuvent se gérer dans un contexte `Dynamics` qu'il suffira d'insérer entre les deux contextes `Staff` pour qu'elles apparaissent sur leur propre ligne centrée entre les deux portées. Voir à ce sujet Section 3.1.2 [Nuances], page 162.
- Les indications de doigté sont abordées dans Section 7.1.2 [Doigtés], page 290.
- Les indications en matière de pédalier d'orgue sont traitées comme des articulations. Voir à ce sujet Section 10.3.1 [Indication de pédalier d'orgue], page 436, et Section B.13 [Liste des signes d'articulation], page 943.
- Pour générer des repères verticaux, voir Section 7.2.3 [Quadrillage temporel], page 311.
- En plus des *Laissez vibrer*, les liaisons en matière de clavier peuvent intervenir sur des accords arpégés ou des trémolos. Reportez-vous au chapitre Section 2.1.4 [Liaisons de prolongation], page 64.
- Le traitement des arpèges couvrant plusieurs voix ou portées est abordé au chapitre Section 3.3.2 [Arpèges], page 186.
- Une description des indications de trémolo est disponible au chapitre Section 4.2.2 [Répétitions en trémolo], page 217.
- Certaines retouches particulières au monde des claviers sont abordées au chapitre Section "Exemple concret" dans *Manuel d'initiation*.
- Des notes fantômes permettent d'introduire des liaisons de tenue qui passent d'une voix à l'autre, comme le montre Section "Autres utilisations des retouches" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Autres utilisations des retouches" dans *Manuel d'initiation*, Section "Exemple concret" dans *Manuel d'initiation*.

Manuel de notation : Section 3.3.2 [Arpèges], page 186, Section 7.1.2 [Doigtés], page 290, Section 2.1.4 [Liaisons de prolongation], page 64, Section B.13 [Liste des signes d'articulation], page 943, Section 6.3.1 [Noms d'instrument], page 270, Section 7.2.3 [Quadrillage temporel], page 311, Section 6.1.2 [Regroupement de portées], page 250, Section 4.2.2 [Répétitions en trémolo], page 217, Section 5.2.3 [Résolution des collisions], page 233, Section 5.2.6 [Saisie de musique en parallèle], page 244.

Morceaux choisis : Section "Claviers et autres instruments à portées multiples" dans *Morceaux choisis*.

Référence des propriétés internes : Section "PianoStaff" dans *Référence des propriétés internes*.

10.1.2 Changement de portée manuel
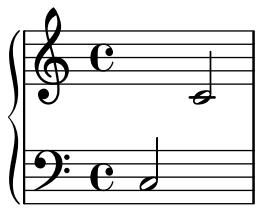
Il est possible de passer d'une portée à l'autre de façon manuelle, au moyen de la commande

```
\change Staff = nomDeLaPortee
```

La valeur *nomDeLaPortee* est le nom de la portée sur laquelle va se déplacer la voix courante. Pour des raisons pratiques, on nomme la portée supérieure `"haut"` et la portée inférieure `"bas"`, donc *nomDeLaPortee* désigne habituellement `"haut"`, `"bas"`, `"MD"` ou `"MG"`.

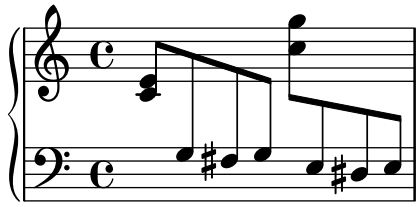
Dans tous les cas, le contexte de portée ainsi utilisé doit exister au préalable. Le cas échéant, vous devrez « garder actives » certaines portées – voir Section 33.3 [Conservation d'un contexte], page 767, à ce propos, ou bien explicitement instanciées – en recourant par exemple à un accord vide <> (voir Section 5.1.1 [Notes en accords], page 221).

```
\new PianoStaff <<
  \new Staff = "up" {
    % enforce creation of all contexts at this point of time
    <>
    \change Staff = "down" c2
    \change Staff = "up" c'2
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```



Pour ligaturer automatiquement des notes entre deux portées, procédez ainsi :

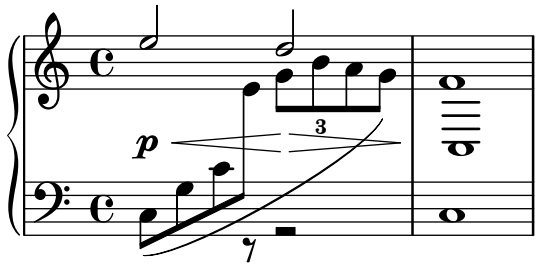
```
\new PianoStaff <<
  \new Staff = "up" {
    <e' c'>8
    \change Staff = "down"
    g8 fis g
    \change Staff = "up"
    <g'' c''>8
    \change Staff = "down"
    e8 dis e
    \change Staff = "up"
  }
  \new Staff = "down" {
    \clef bass
    % keep staff alive
    s1
  }
>>
```



Si les ligatures demandaient à être retouchées, commencez par modifier la direction des hampes. L'emplacement des ligatures sera alors calculé à partir du centre de la portée la plus proche – voir Section "Correction des collisions d'objets" dans *Manuel d'initiation* pour un exemple de retouche sur des ligatures.

Une voix qui change de portée peut entraîner des collisions :

```
\new PianoStaff <<
  \new Staff = "up" {
    \voiceOne
    % Make space for fingering in the cross-staff voice
    \once\override DynamicLineSpanner.staff-padding = 4
    e''2\p\< d''\>
    c1\!
  }
  \new Staff = "down" <<
    {
      \clef bass
      s4. e,8\rest g,2\rest
      c1
    } \\ {
      c8\( g c'
      \change Staff = "up"
      e' g' b'-3 a' g'\)
      f'1
    }
  >>
>>
```



Hampes et liaisons viennent en surimpression sur la ligne des nuances parce que la résolution automatique des collisions est inactivée pour ce qui relie les notes de différentes portées ainsi que pour les hampes ou extenseurs affectés à des notes incluses dans un changement de portée. Lorsque des collisions surviennent en pareil cas, il vous faudra les résoudre, en suivant les directives du chapitre Section "Correction des collisions d'objets" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Correction des collisions d'objets" dans *Manuel d'initiation*.

Manuel de notation : Section 2.4.1 [Barres de ligature automatiques], page 104, Section 33.3 [Conservation d'un contexte], page 767, Section 7.1.9 [Hampes], page 306.

Morceaux choisis : Section “Claviers et autres instruments à portées multiples” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Beam” dans *Référence des propriétés internes*, Section “ContextChange” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Dans la mesure où l'évitement de collision des ligatures ne fonctionne pas lorsqu'une ligature automatique prend fin juste avant un changement de portée, utilisez alors une ligature manuelle.

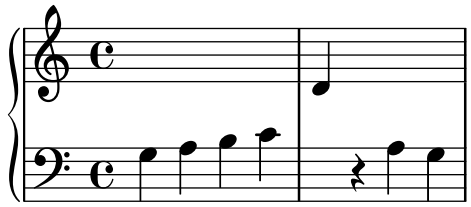
10.1.3 Changement de portée automatique

Les voix peuvent passer automatiquement d'une portée à l'autre, au moyen de la syntaxe suivante :

```
\autoChange ...musique...
```

Deux portées seront alors créées au sein du contexte `PianoStaff`, nommées respectivement `"up"` et `"down"`. La portée du bas sera par défaut en clef de fa. La commande `\autoChange` bascule les notes d'une portée à l'autre en fonction de leur hauteur (le do du milieu servant de pivot), et place les silences en fonction des notes qui les suivront. Ainsi :

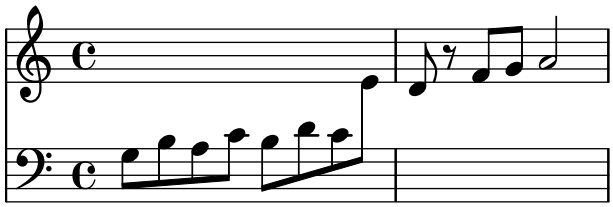
```
\new PianoStaff {
  \autoChange {
    g4 a b c'
    d'4 r a g
  }
}
```
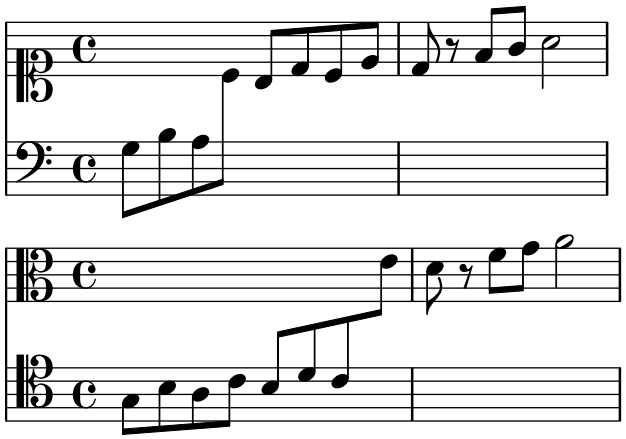


Il est tout à fait possible de déterminer une autre hauteur charnière. Dès lors qu'aucune portée n'a été spécifiquement instanciée, d'autres clefs peuvent être utilisées.

```
music = {
  g8 b a c' b8 d' c'8 e'
  d'8 r f' g' a'2
}

\autoChange d' \music
\autoChange b \with { \clef soprano } \music
\autoChange d' \with { \clef alto } \with { \clef tenor } \music
```

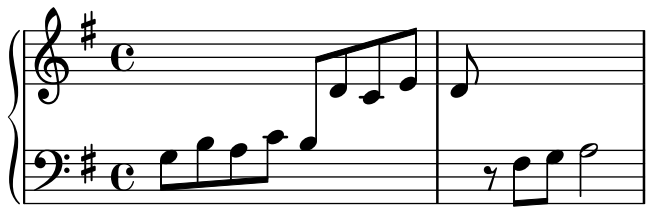




Une section en mode `\relative` se situant en dehors de la commande `\autoChange` n'aura pas d'effet sur les hauteurs de l'expression *musique*. Il est donc préférable d'introduire la directive `\relative` **après** `\autoChange`.

Lorsque des contrôles particuliers doivent s'appliquer aux portées, mieux vaut les nommer explicitement – attention : sous peine d'effet indésirable quant au résultat, la portée supérieure doit s'appeler `"up"` et l'inférieure `"down"` ; *ceci est à notre connaissance le seul cas où ces noms de variable sont figés*. Cette procédure sert, entre autres, à indiquer l'armure sur la portée inférieure :

```
\new PianoStaff <<
  \new Staff = "up" {
    \new Voice = "melodieUn" {
      \key g \major
      \autoChange \relative {
        g8 b a c b d c e
        d8 r fis, g a2
      }
    }
  }
  \new Staff = "down" {
    \key g \major
    \clef bass
  }
>>
```



Voir aussi

Manuel de notation : Section 10.1.2 [Changement de portée manuel], page 428.

Morceaux choisis : Section "Claviers et autres instruments à portées multiples" dans *Morceaux choisis*.

Problèmes connus et avertissements

Les changements de portée automatiques n'interviennent pas toujours à l'endroit le plus opportun. Pour un résultat de meilleure qualité, il vaut mieux indiquer soi-même ces changements.

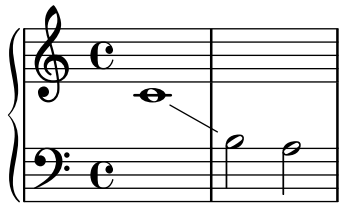
Un accord ne peut se répartir sur plusieurs portées ; sa portée d'affectation sera déterminée par la première hauteur mentionnée dans la construction de cet accord.

`\autoChange` ne peut intervenir à l'intérieur d'une commande `\tuplet`.

10.1.4 Lignes de changement de portée

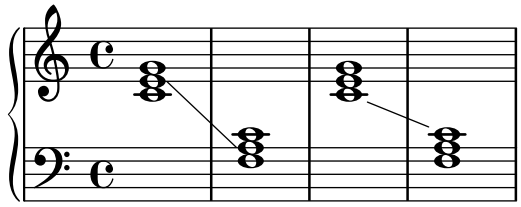
Lorsqu'une voix change de portée, il est possible d'imprimer automatiquement une ligne reliant les notes, en faisant appel à la commande `\showStaffSwitch` :

```
\new PianoStaff <<
  \new Staff = "one" {
    \showStaffSwitch
    c'1
    \change Staff = "two"
    b2 a
  }
  \new Staff = "two" {
    \clef bass
    s1*2
  }
>>
```



Dans le cas d'accords, cette ligne connectera la **dernière** hauteur de chacun d'entre eux selon leur ordre d'apparition dans le fichier source ; ceci permet d'ajuster rapidement les positions de départ et d'arrivée de cette ligne.

```
\new PianoStaff <<
  \new Staff = "one" {
    <c' e' g'>1
    \showStaffSwitch
    \change Staff = "two"
    <a c' f>1
    \hideStaffSwitch
    \change Staff = "one"
    <e' g' c'>1
    \showStaffSwitch
    \change Staff = "two"
    <f a c'>1
  }
  \new Staff = "two" {
    \clef bass
    s1*4
  }
>>
```



Commandes prédéfinies

`\showStaffSwitch`, `\hideStaffSwitch`.

Voir aussi

Morceaux choisis : Section “Claviers et autres instruments à portées multiples” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Note_head_line_engraver” dans *Référence des propriétés internes*, Section “VoiceFollower” dans *Référence des propriétés internes*.

Morceaux choisis

Hampes interportées

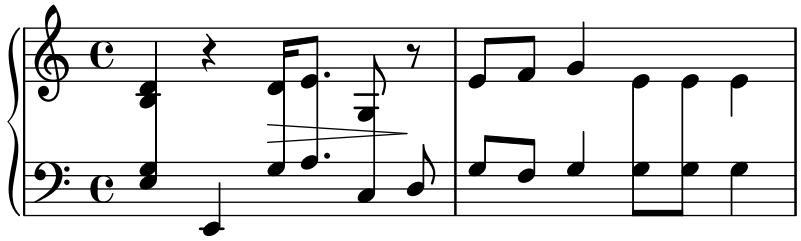
L’exemple ci-dessous illustre l’utilisation du `Span_stem_engraver` et de la commande `\crossStaff` afin de connecter des hampes entre les portées.

Nul n’est besoin de spécifier la taille des hampes ; le graveur calcule automatiquement la distance relative des têtes de note avec les portées. Il est toutefois important d’appliquer `\crossStaff` dans le contxte de voix ou de portée approprié – autrement dit en opposition à l’endroit où se trouvera une hampe) pour obtenir l’effet attendu.

```
\layout {
  \context {
    \PianoStaff
    \consists "Span_stem_engraver"
  }
}

\new PianoStaff <<
  \new Staff {
    <b d'>4 r d'16\> e'8. g8 r\! |
    e'8 f' g'4
      \voiceTwo
      % Down to lower staff
      \crossStaff { e'8 e'8 } e'4 |
  }

  \new Staff {
    \clef bass
    \voiceOne
    % Up to upper staff
    \crossStaff { <e g>4 e, g16 a8. c8 } d |
    g8 f g4 \voiceTwo g8 g g4 |
  }
>>
```



Indication d'un accord à cheval sur deux portées par un crochet

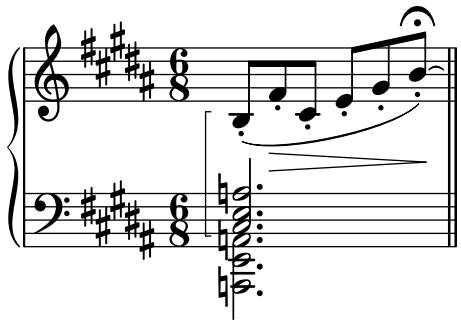
Un crochet « non-arpeggiato » peut indiquer que des notes réparties sur deux portées différentes doivent être jouées par la même main. Le contexte `PianoStaff` doit accepter ces crochets « distribués ».

Est ici reproduite la mesure 65 du prélude de Debussy *Les collines d'Anacapri*.

```
\new PianoStaff <<
  \once \set PianoStaff.connectChordBrackets = ##t

  \new Staff \relative c' {
    \key b \major
    \time 6/8
    b8-.(\nonArpeggiato fis'-.\> cis-.
      e-. gis-. b-.)\!\fermata^\laissezVibrer
    \section
  }

  \new Staff \relative c' {
    \clef bass
    \key b \major
    << { <a e cis>2.\nonArpeggiato } \\
       { <a, e a,>2. } >>
    \section
  }
>>
```



Voir aussi

Morceaux choisis : Section "Claviers et autres instruments à portées multiples" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Stem" dans *Référence des propriétés internes*.
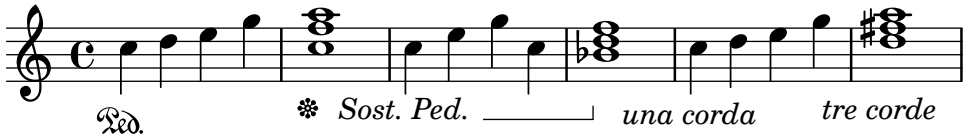
10.2 Piano

Ce chapitre traite des aspects de la notation directement liés au piano.

10.2.1 Pédales de piano

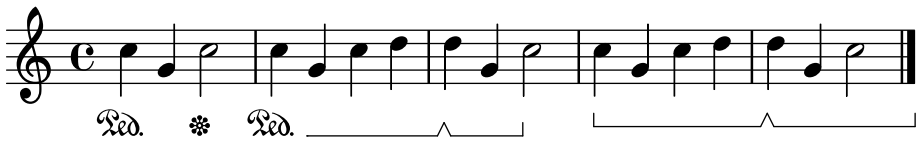
Le piano possède deux pédales, parfois trois, permettant de modifier l'émission du son : une pédale de *tenue* (*sustain*), une pédale de *sourdine* (*una corda* ou *U.C.*) et une pédale *tonale* (*sostenuto* ou *sos.*). La pédale *sustain* se rencontre aussi sur les vibraphones et celestas.

```
\relative {
  c''4\sustainOn d e g
  <c, f a>1\sustainOff
  c4\sostenutoOn e g c,
  <bes d f>1\sostenutoOff
  c4\unaCorda d e g
  <d fis a>1\treCorde
}
```



Trois styles sont à votre disposition pour imprimer les indications de pédale : sous forme de texte, de crochet, ou une combinaison des deux. `text` est le style de notation par défaut pour les pédales de tenue ou de sourdine – le traditionnel « *Ped. ». La pédale tonale, en revanche, utilise `mixed` par défaut.

```
\relative {
  c''4\sustainOn g c2\sustainOff
  \set Staff.pedalSustainStyle = #'mixed
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2\sustainOff
  \set Staff.pedalSustainStyle = #'bracket
  c4\sustainOn g c d
  d\sustainOff\sustainOn g, c2
  \bar "|."
}
```



Le placement des commandes de pédale correspond au mouvement de la pédale de tenue pendant le jeu. Garder une pédale enclenchée jusqu'à la barre finale s'obtient tout simplement en omettant la commande de relâcher.

Les indications de pédale peuvent s'inscrire dans un contexte `Dynamics`, ce qui aura pour effet de leur attribuer une ligne en propre.

Voir aussi

Manuel de notation : Section 2.1.4 [Liaisons de prolongation], page 64.

Morceaux choisis : Section "Claviers et autres instruments à portées multiples" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Piano_pedal_engraver" dans *Référence des propriétés internes*, Section "PianoPedalBracket" dans *Référence des propriétés internes*, Section "SustainEvent" dans *Référence des propriétés internes*, Section "SostenutoPedal" dans *Référence des propriétés internes*, Section "SustainPedal" dans *Référence des propriétés*

internes, Section “SustainPedalLineSpanner” dans *Référence des propriétés internes*, Section “SostenutoEvent” dans *Référence des propriétés internes*, Section “SostenutoPedalLineSpanner” dans *Référence des propriétés internes*, Section “UnaCordaPedal” dans *Référence des propriétés internes*, Section “UnaCordaEvent” dans *Référence des propriétés internes*, Section “UnaCordaPedalLineSpanner” dans *Référence des propriétés internes*.

10.3 Orgue

Cette section aborde les problématiques de la musique pour orgue d’église.

10.3.1 Indication de pédalier d’orgue

pointe pointe, pdaliertalon talon, pdalier

On trouve, dans la majorité des méthodes pour orgue, des indication d’utilisation des pointe et talon des pieds droit et gauche sur la portée du pédalier. Selon le style de l’ouvrage apparaîtront différents signes. Toutefois, et dans la plupart des cas, les indications de pédale sont positionnées au-dessus de la portée pour le pied droit, et en dessous de la portée pour le pied gauche.

Les principales commandes permettant d’imprimer ces indications sont `\rtoe`, `\ltoe`, `\rheel` et `\lheel`. La propriété de contexte `toeHeelStyle` permet de sélectionner le style à adopter. Les différentes options sont les suivantes :

`default` Valeur par défaut en début de pièce, pour compatibilité ascendante.

`standard` Il s’agit de la nomenclature la plus courante de nos jours. L’apparence des indications est identique des deux côtés de la portée.

`reversed` L’extrémité « fermée » du glyphe pointe vers la portée.

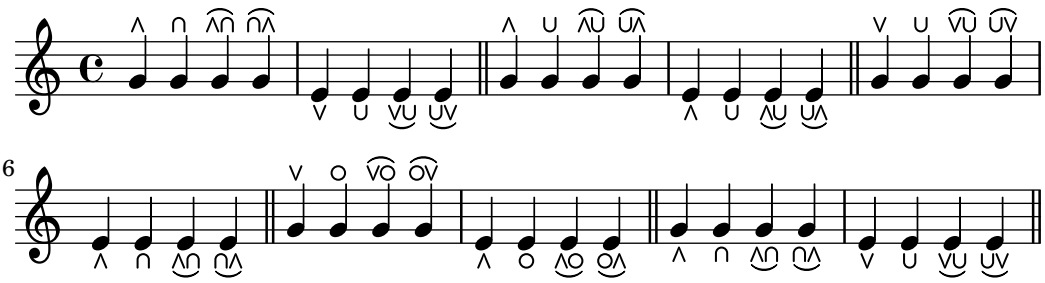
`circleheels` L’indication de talon utilise un glyphe circulaire.

`below` Cette option répond aux rares cas dans lesquels le pédalier apparaît sur la même portée que la main gauche, ce qui rend nécessaire que les indications du pied droit et du pied gauche se placent sous la portée.

Pour des substitutions pointe-talon et talon-pointe sont disponibles les instructions `\rtoeheel`, `\ltoeheel`, `\rheeltoe` et `\lheeltoe`.

```
music = { g'4_\rtoe g'\rheel g'\rtoeheel g'\rheeltoe |
          e'4\ltoe e'\lheel e'\ltoeheel e'\lheeltoe \bar "||" }

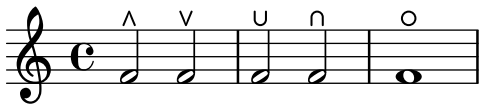
{
  \music              % #'default
  \set toeHeelStyle = #'standard \music
  \set toeHeelStyle = #'reversed \music
  \set toeHeelStyle = #'circleheels \music
  \set toeHeelStyle = #'below \music
}
```



\rtoe et ses semblables ressemblent aux autres articulations telles que \flageolet or \prall, puisqu'il s'agit d'objets graphiques Script. Cependant, ils ignorent les modifications de positionnement – autrement dit, les '^', '_' ou \tweak direction sont ignorés. L'idée sous-jacente est de séparer le contenu de la représentation en laissant le style gérer le positionnement des indications.

S'il est besoin de positionner individuellement les indication de pédalier en surplomb ou au-dessous de la portée, doivent s'utiliser les instruction d'articulation standard \toe, \vartoe, \heel, \varheel et \heelcircle.

```
{
  f'2\toe f'2\vartoe |
  f'2\heel f'2\varheel |
  f'1\heelcircle
}
```



10.4 Accordéon

10.4.1 Symboles de jeux

De nombreux accordéons possèdent plusieurs jeux d'anches capables de jouer à l'unisson ou bien à l'octave inférieure ou supérieure par rapport aux notes écrites. Chaque facteur d'accordéon donne des noms différents aux *tirettes* (*shifts*) qui permettent de sélectionner les différents jeux d'anches tels que *hautbois*, *musette* ou *bandonéon*, de telle sorte qu'un système de symbole a fini par voir le jour afin de faciliter les instructions pour exécuter un morceau.

Une liste des différents symboles est disponible à l'annexe Section A.1.7 [Accordion registers], page 898.

Morceaux choisis

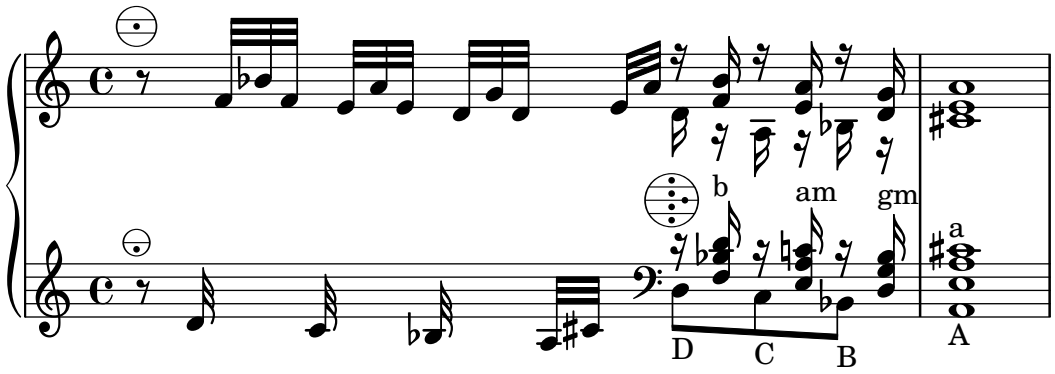
Symboles de registre pour accordéon

Les symboles spécifiques aux registres d'accordéon sont disponible tant sous forme de \markup que d'événements musicaux indépendants – un changement de registre a tendance à intervenir entre des événements musicaux réels. Les registres de basse ne sont pas très standardisés. Les différentes commandes disponibles sont regroupées à l'annexe Section "Registres d'accordéon" dans *Manuel de notation*.

```
#(use-modules (lily accreg))

\new PianoStaff <<
  \new Staff \relative {
    \clef treble
    \discant "10"
    r8 s32 f'[ bes f] s e[ a e] s d[ g d] s16 e32[ a]
    <<
      { r16 <f bes> r <e a> r <d g> }
      \\
      { d r a r bes r }
    >> |
    <cis e a>1
  }
```

```
  \new Staff \relative {
    \clef treble
    \freeBass "1"
    r8 d'32 s16. c32 s16. bes32 s16. a32[ cis] s16
    \clef bass \stdBass "Master"
    <<
      { r16 <f, bes d>^"b" r <e a c>^"am" r <d g bes>^"gm" |
      <e a cis>1^"a" }
      \\
      { d8_"D" c_"C" bes_"B" | a1_"A" }
    >>
  }
>>
```



Voir aussi

Morceaux choisis : Section “Claviers et autres instruments à portées multiples” dans *Morceaux choisis*.

10.5 Harpe

Cette partie s’intéresse aux particularités en matière de notation pour la harpe.

10.5.1 Généralités sur la harpe

Certaines caractéristiques de la musique pour harpes sont abordées dans d’autres chapitres, tels que :

- Les glissandos, l’une des techniques spécifique à la harpe, Section 3.3.1 [Glissando], page 181.
- Le *bisbigliando*, qui s’écrit comme un trémolo, Section 4.2.2 [Répétitions en trémolo], page 217.
- Les harmoniques naturelles sont présentées dans Section 11.1.3 [Harmoniques], page 441.
- L’interprétation des arpeggios est abordée dans Section 3.3.2 [Arpèges], page 186.

Voir aussi
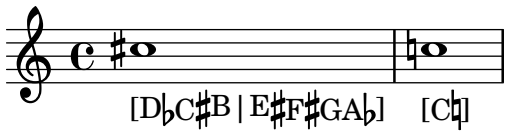
Manuel de notation : Section 3.3.2 [Arpèges], page 186, Section 3.3.1 [Glissando], page 181, Section 11.1.3 [Harmoniques], page 441, Section 4.2.2 [Répétitions en trémolo], page 217.

10.5.2 Pédales de harpe

Les harpes comportent sept cordes par octave qui peuvent sonner naturel, dièse ou bémol. Si chacune des cordes de la harpe celtique (*lever harp*) est accordée individuellement, celles d’une harpe à pédalier ayant la même note de base sont contrôlées par une seule pédale. De gauche à droite, elles correspondent aux notes ré, do, si, et mi, fa, sol, la pour la grande harpe. Les

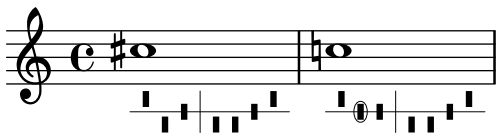
trois premières pédales sont réservées au pied gauche, les quatre dernières au pied droit. Leur position peut être indiquée par une marque textuelle :

```
\textLengthOn
cis''1_\markup \concat \vcenter {
  [D \flat C \sharp B|E \sharp F \sharp G A \flat] }
c''!1_\markup \concat \vcenter {
  [ C \natural ]}
```



ou bien sous forme de diagramme :

```
\textLengthOn
cis''1_\markup { \harp-pedal "^v-|vv-^" }
c''!1_\markup { \harp-pedal "^o--|vv-^" }
```

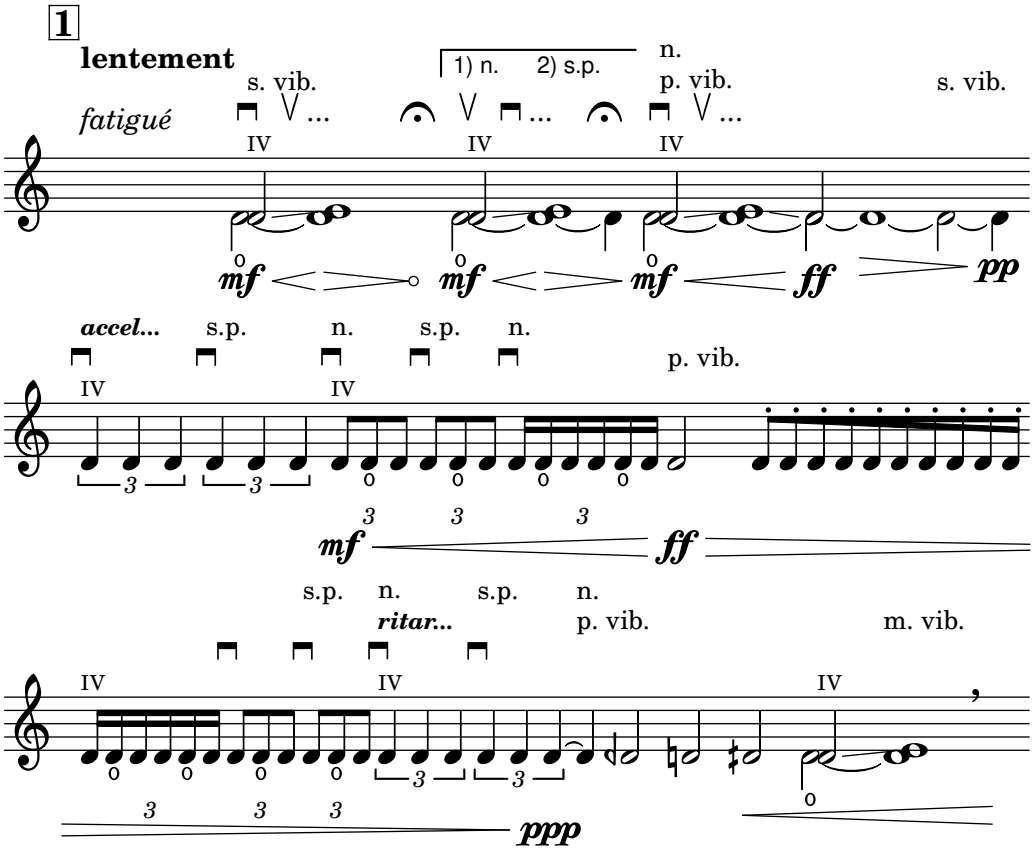


Bémol si la pédale est relâchée (ou en haut), bécarre si elle est bloquée sur le cran du milieu, et dièse si elle est tout à fait enfoncée. La commande `\harp-pedal` prend en argument une chaîne de caractères, où `^` indique que la pédale est relâchée ou en haut (bémol), `-` qu'elle est bloquée sur le cran du milieu (bécarre), `v` qu'elle est tout à fait enfoncée (dièse) et `|` représente le séparateur (entre gauche et droite de l'instrumentiste). Faire précéder un symbole par un `o` permet de l'inscrire dans un cercle.

Voir aussi

Manuel de notation : Section 8.1.2 [Commentaires textuels], page 320, Section A.1.6 [Instrument-specific markup], page 892.

11 Cordes non frettées



Cette section dispense des informations supplémentaires et utiles à l'écriture pour les cordes non frettées, et tout spécialement pour les cordes d'orchestre.

11.1 Vue d'ensemble de la notation pour cordes non frettées

Il existe peu d'éléments spécifiques à l'écriture pour instruments à cordes non frettées. La musique est notée sur une portée et ne nécessite généralement qu'une seule voix. Le recours à deux voix distinctes peut cependant permettre de traiter efficacement des doubles notes ou des divisions.

11.1.1 Références en matière de cordes non frettées

La majorité des éléments de notation nécessaires en matière de cordes d'orchestre et autres instruments à archet est abordée dans d'autres chapitres de la documentation :

- Les indications textuelles telles que « pizz. » ou « arco » sont ajoutées comme du texte simple – voir à ce sujet Section 8.1.2 [Commentaires textuels], page 320.
- Les indications de doigtés, incluant les indications du pouce, sont décrites dans Section 7.1.2 [Doigtés], page 290.
- Des numéros de corde peuvent être ajoutés (généralement en chiffre romain pour les instruments à archet) – voir à ce sujet Section 12.1.2 [Indications du numéro de corde], page 445.
- Les doubles notes sont généralement indiquées en écrivant un accord, voir Section 5.1.1 [Notes en accords], page 221. Les précisions pour jouer des accords peuvent être ajoutées, comme l'indique Section 3.3.2 [Arpèges], page 186.
- Un squelette de partition est disponible à l'annexe Section "Modèles pour quatuor à cordes" dans *Manuel d'initiation*. D'autres informations se trouvent dans les exemples de code.

Voir aussi

Manuel d'initiation : Section "Modèles pour quatuor à cordes" dans *Manuel d'initiation*.

Manuel de notation : Section 3.3.2 [Arpèges], page 186, Section 8.1.2 [Commentaires textuels], page 320, Section 7.1.2 [Doigtés], page 290, Section 5.1.1 [Notes en accords], page 221.

Morceaux choisis : Section "Cordes non frettées" dans *Morceaux choisis*.
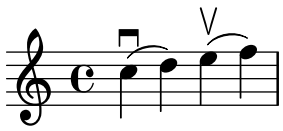
11.1.2 Indications d'archet

pouss pouss, indication d'archettir tir, indication d'archet

Les indications d'archet se créent comme des articulations, elles sont décrites dans Section 3.1.1 [Articulations et ornements], page 158.

Les indications d'archet, poussé (`\upbow`) et tiré (`\downbow`), peuvent se combiner à des liaisons comme ici :
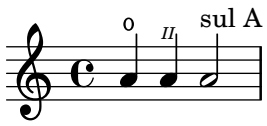
```
\relative { c''4(\downbow d) e(\upbow f) }
```



Des chiffres romains peuvent s'ajouter pour indiquer les numéros de corde (en lieu et place de chiffres arabes cerclés), comme expliqué dans Section 12.1.2 [Indications du numéro de corde], page 445.

Alternativement, les indications de corde peuvent se traiter sous forme de *markup*, et un script indiquer une corde à vide.

```
a'4 \open
\romanStringNumbers
a'\2
a'2^\markup { \small "sul A" }
```



Commandes prédéfinies

`\downbow`, `\upbow`, `\open`, `\romanStringNumbers`.

Voir aussi
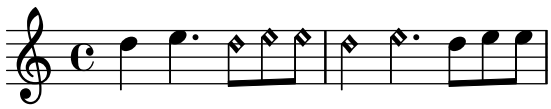
Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Section 12.1.2 [Indications du numéro de corde], page 445, Section 3.2.1 [Liaisons d'articulation], page 174.

11.1.3 Harmoniques

Harmoniques naturels
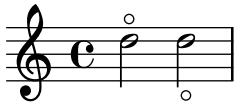
Les harmoniques naturels sont indiqués de différentes manières : une tête de note en forme de losange signifie généralement que vous devez effleurer la corde à l'endroit où vous l'auriez pincée si la note avait été normale.

```
\relative d'' {
  d4 e4.
  \harmonicsOn
  d8 e e
  d4 e4.
  \harmonicsOff
  d8 e e
}
```



Une autre façon de procéder consiste à faire surmonter la note normale d'un petit cercle. Ceci indique que la note écrite doit être jouée en harmonique :

```
d''2^\flageolet d''_\flageolet
```
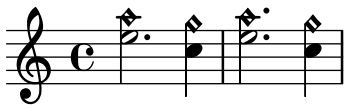


Harmoniques artificiels

Les harmoniques artificiels sont indiqués par une double tête de note : l'une normale, indique la note à pincer, et l'autre, en forme de losange, indique l'endroit où la corde doit être effleurée.

La propriété `harmonicDots`, lorsqu'elle est activée, permet d'ajouter un point aux notes pointées affublées d'un `\harmonic`.

```
<e a\harmonic>2.  <c g'\harmonic>4
\set harmonicDots = ##t
<e a\harmonic>2.  <c g'\harmonic>4
```



Voir aussi

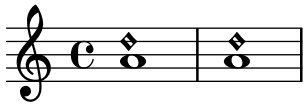
Glossaire musicologique : Section "harmonics" dans *Glossaire*.

Manuel de notation : Section 11.1.1 [Références en matière de cordes non frettées], page 440, Section 1.4.1 [Têtes de note spécifiques], page 45.

Problèmes connus et avertissements

Une harmonique sera centrée sur une ronde dès lors qu'est adoptée une construction d'accord.

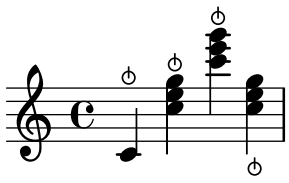
```
{
  << { d''1\harmonic } \\ { a'1 } >>
  <a' d''\harmonic>1
}
```
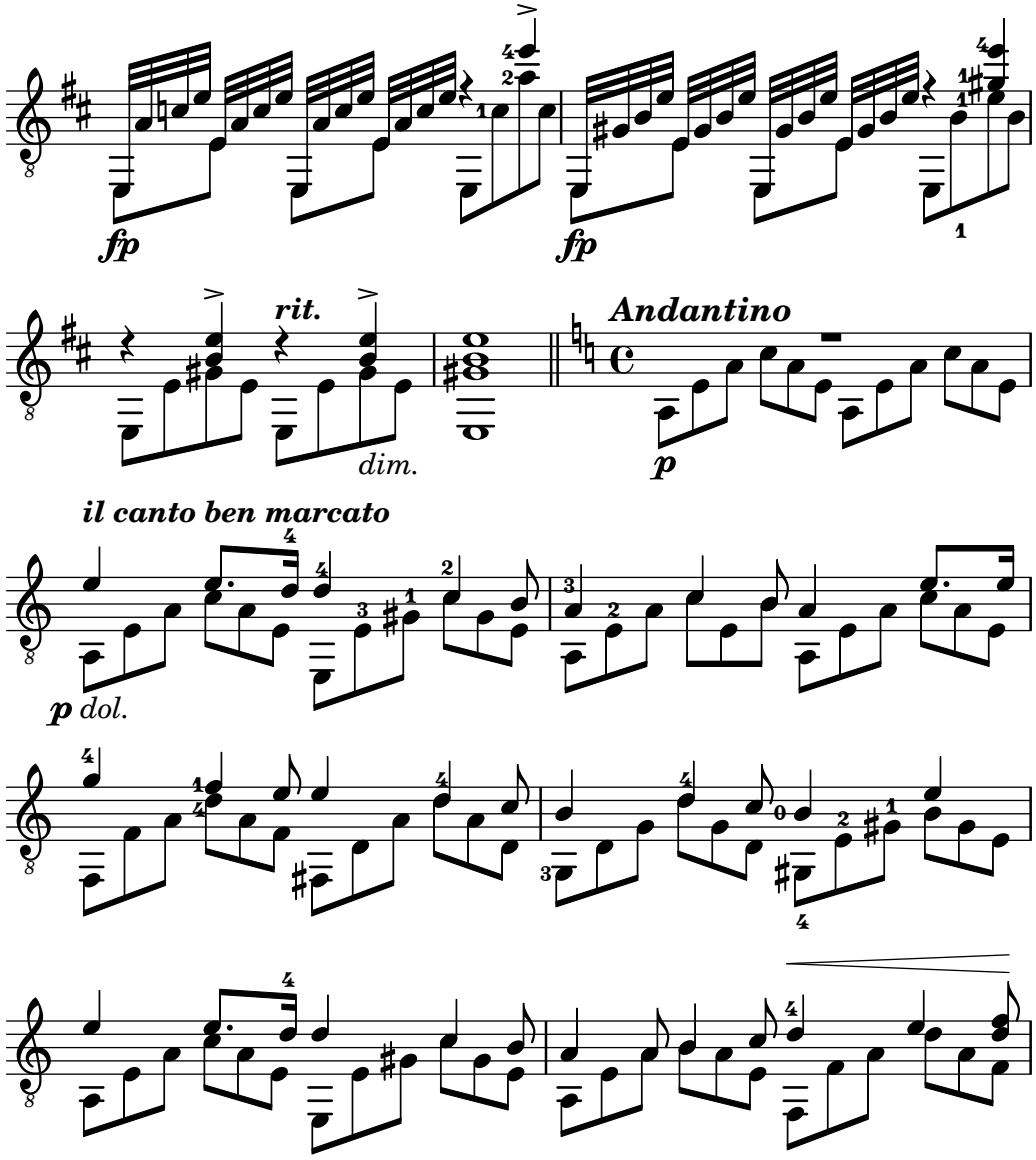


11.1.4 Snap (Bartók) pizzicato

Un *snap pizzicato*, aussi appelé « Bartok pizz » est un type de pizzicato pour lequel la corde est tirée vers le haut (plutôt que sur le côté) de telle sorte qu'elle vienne frapper le manche.

```
\relative {
  c'4\snappizzicato
  <c' e g>4\snappizzicato
  <c' e g>4^\snappizzicato
  <c, e g>4_\snappizzicato
}
```



12 Instruments à cordes frettées



Cette section traite de différents aspects de la notation propre aux instruments à cordes frettées.

12.1 Vue d'ensemble des cordes frettées

Nous allons aborder, dans les paragraphes qui suivent, les particularités communes aux différents instruments à cordes frettées.

12.1.1 Références en matière de cordes frettées

La musique pour instruments à cordes frettées est généralement notée sur une seule portée, en notation traditionnelle ou en tablature, les deux étant parfois combinées. Il est aussi courant en musique populaire d'utiliser des diagrammes d'accord au-dessus de la portée traditionnelle. La guitare et le banjo sont des instruments transpositeurs, sonnant une octave au-dessous de ce qui est écrit. Les partitions pour ces instruments utilisent donc la clé de sol *ottava bassa* `"treble_8"` – ou une instruction `\transposition c` pour un rendu MIDI correct. Vous pourrez trouver ailleurs dans la documentation d'autres éléments aussi utilisés par les instruments à cordes frettées :

- Les doigtés s'obtiennent comme indiqué au chapitre Section 7.1.2 [Doigtés], page 290.

- En plus des *Laissez vibrer*, les liaisons peuvent intervenir sur des accords arpégés ou des trémolos. Reportez-vous au chapitre Section 2.1.4 [Liaisons de prolongation], page 64.
- Des indications quant à la prise en charge de la polyphonie se trouvent au chapitre Section 5.2.3 [Résolution des collisions], page 233.
- La notation des sons harmoniques se trouve à la section Section 11.1.3 [Harmoniques], page 441.

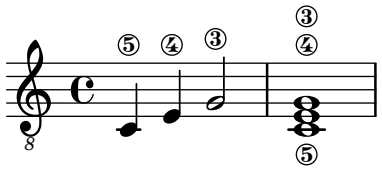
Voir aussi

Manuel de notation : Section 3.3.2 [Arpèges], page 186, Section 1.3.1 [Clefs], page 20, Section 7.1.2 [Doigtés], page 290, Section 1.3.4 [Instruments transpositeurs], page 30, Section 2.1.4 [Liaisons de prolongation], page 64, Section B.13 [Liste des signes d'articulation], page 943, Section 6.3.1 [Noms d'instrument], page 270, Section 5.2.3 [Résolution des collisions], page 233, Section 5.2.6 [Saisie de musique en parallèle], page 244.

12.1.2 Indications du numéro de corde

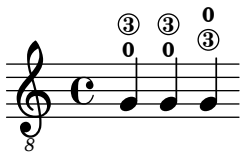
La corde sur laquelle une note doit être jouée peut être indiquée en attachant `\numéro` à cette note prise dans une construction de type accord <>.

```
\clef "treble_8"
c4\5 e\4 g2\3
<c\5 e\4 g\3>1
```



Quand les indications de doigté et de numéro de corde sont attachées à une même note, leur positionnement respectif peut se gérer à l'aide de la propriété `script-priority` – voir [Contrôle de l'ordre vertical des articulations et ornements], page 161, et Section B.18 [Valeurs par défaut de `script-priority`], page 953. Leur ordre d'apparition dans la source n'a donc aucune importance.

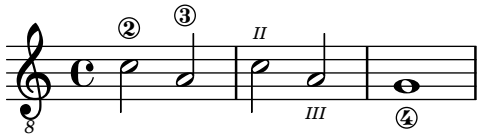
```
\clef "treble_8"
g4\3-0
g-0\3
g-\tweak script-priority 200 -0 \3
```



Les numéros de corde peuvent aussi, comme traditionnellement pour les cordes non frettées, s'imprimer en chiffres romains placés sous la portée plutôt qu'en surplomb.

```
\clef "treble_8"
c'2\2
a\3
\romanStringNumbers
c'\2
\set stringNumberOrientations = #'(down)
a\3
\arabicStringNumbers
```

```
g1\4
```



Le traitement des numéros de corde, objets `StringNumber`, et notamment leur positionnement, est comparable à celui des doigtés – voir Section 7.1.2 [Doigtés], page 290.

Commandes prédéfinies

`\arabicStringNumbers`, `\romanStringNumbers`.

Voir aussi

Manuel de notation : Section 7.1.2 [Doigtés], page 290.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

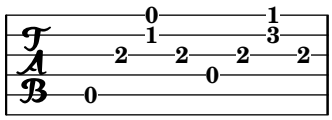
Référence des propriétés internes : Section “StringNumber” dans *Référence des propriétés internes*, Section “Fingering” dans *Référence des propriétés internes*.

12.1.3 Tablatures par défaut

La notation en tablature est utilisée pour certains instruments à cordes pincées. Les hauteurs n’y sont pas indiquées par des têtes de note mais par des chiffres ou autres symboles qui indiquent sur quelle corde et à quelle case chaque note doit être jouée. Des notes devant être jouées simultanément seront alors alignées verticalement.

Par défaut, la première corde est la plus aiguë et correspond à la ligne supérieure du `TabStaff`. Les cordes suivent par défaut l’accordage traditionnel d’une guitare (à six cordes). Les notes sont imprimées sous forme de tablature dans les contextes `TabStaff` et `TabVoice`, qui comportent une clef spécifique ajoutée automatiquement.

```
\new TabStaff \relative {
  a,8 a' <c e> a
  d,8 a' <d f> a
}
```

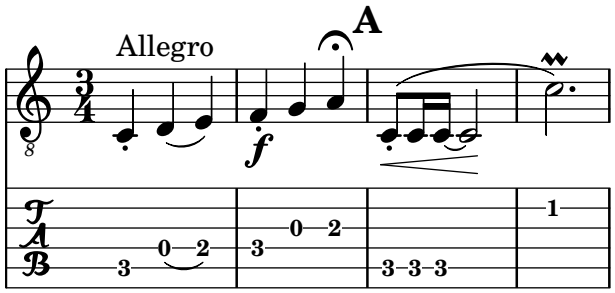


Par défaut, les tablatures ne comportent aucune marque de durée ni de symbole musical tel que des nuances.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c ~ 2\!
  c'2.\prall\)
}

\score {
  <<
    \new Staff { \clef "G_8" \symbols }
```

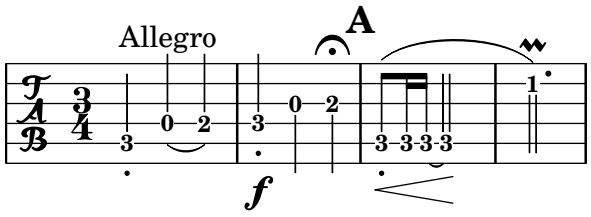
```
    \new TabStaff   { \symbols }
  >>
}
```



Pour obtenir les symboles de notation habituelle dans une tablature, il faut appliquer la commande `\tabFullNotation` au contexte `TabStaff`. Vous noterez que les blanches sont affublées d'une double hampe afin qu'elles ne se confondent pas avec des noires.

```
symbols = {
  \time 3/4
  c4-.^"Allegro" d( e)
  f4-.\f g a^\fermata
  \mark \default
  c8_.\<\( c16 c ~ 2\!
  c'2.\prall\)
}

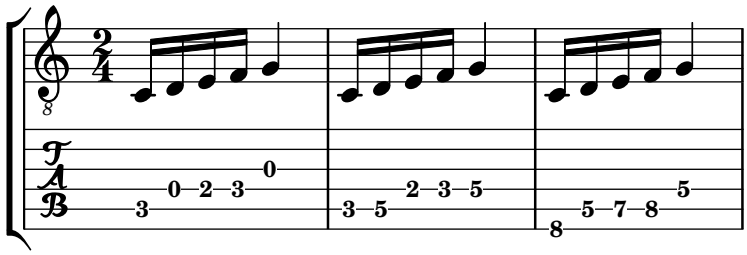
\score {
  \new TabStaff {
    \tabFullNotation
    \symbols
  }
}
```



Quand aucune corde n'est précisée, LilyPond choisit automatiquement la corde pour laquelle la position est la moins élevée, avec une préférence pour une corde à vide. Vous pouvez préférer qu'une note donnée soit jouée sur une corde particulière, auquel cas l'affectation directe du numéro de corde suffit. L'absence d'indication des numéros de corde en notation traditionnelle se gère au niveau des stencils. Il est cependant plus facile de jouer sur la propriété `minimumFret`, dont la valeur par défaut est fixée à 0, ce qui correspond aux cordes à vide.

Cependant, et en dépit d'une affectation de `minimumFret`, une corde à vide aura toujours préséance. Ce comportement se modifie par l'activation de `restrainOpenStrings`.

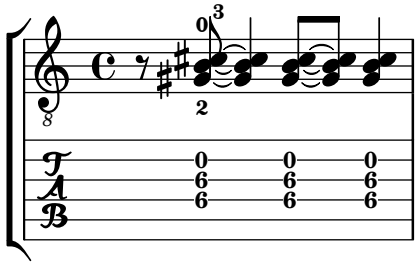
```
\layout { \omit Voice.StringNumber }
\new StaffGroup <<
   \new Staff \relative {
     \clef "treble_8"
     \time 2/4
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     c,16 d e f g4
   }
   \new TabStaff \relative {
     c16 d e f g4
     c,16\5 d\5 e\4 f\4 g4\4
     \set TabStaff.minimumFret = 5
     \set TabStaff.restrainOpenStrings = ##t
     c,16 d e f g4
   }
>>
```



La répétition d'une construction en accord s'indique par un `q` – voir Section 5.1.2 [Répétition d'accords], page 223. Cette fonctionnalité, bien qu'opérationnelle en mode tablature, supprime entre autres les numéros de corde et doigtés, ce qui peut conduire à des résultats inattendus, en particulier des changement de fret. L'instruction `\tabChordRepeats` permet la persistance des doigtés au fil des répétitions. Dans l'exemple suivant, le doigté par défaut pour cet accord – sans qu'il soit affiché – serait sol dièse sur la quatrième corde, si sur la troisième, et do dièse sur la deuxième. Puisqu'est mentionné `b-0` dans la saisie, le si est sur la deuxième corde et le do dièse va sur la troisième. La présence du `\tabChordRepeats` permet de maintenir le même doigté pour les accords `q` qui suivent.

```
guitar = \relative {
  r8 <gis-2 cis-3 b-0>~ q4 q8~ 8 q4
}

\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \guitar
  }
  \new TabStaff {
    \tabChordRepeats \guitar
  }
>>
```

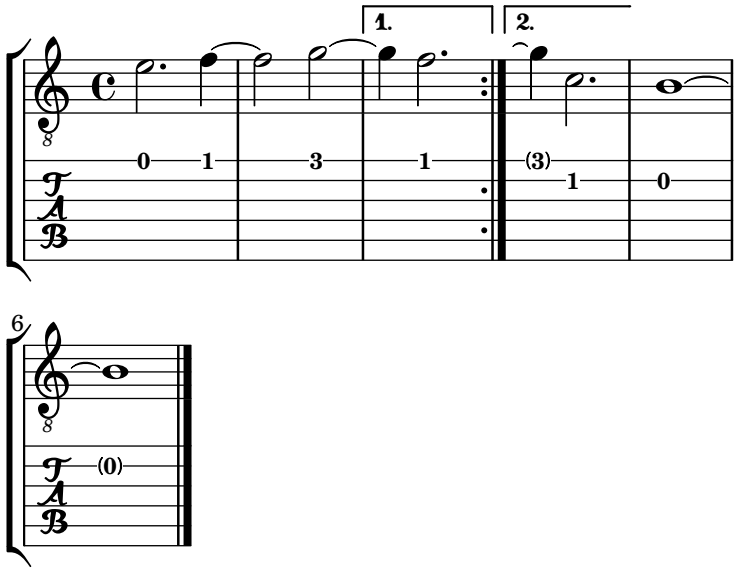


Lorsqu'une liaison de prolongation intervient à l'occasion d'un saut de ligne, la note est répétée, entre parenthèses. Il en va de même pour la seconde alternative d'une répétition.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~
  }
  \alternative {
     \volta 1 { g4 f2. }
     \volta 2 { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
        \ties
      }
    >>
  >>
  \layout {
```

```
  indent = 0
  ragged-right = ##t
  }
}
```

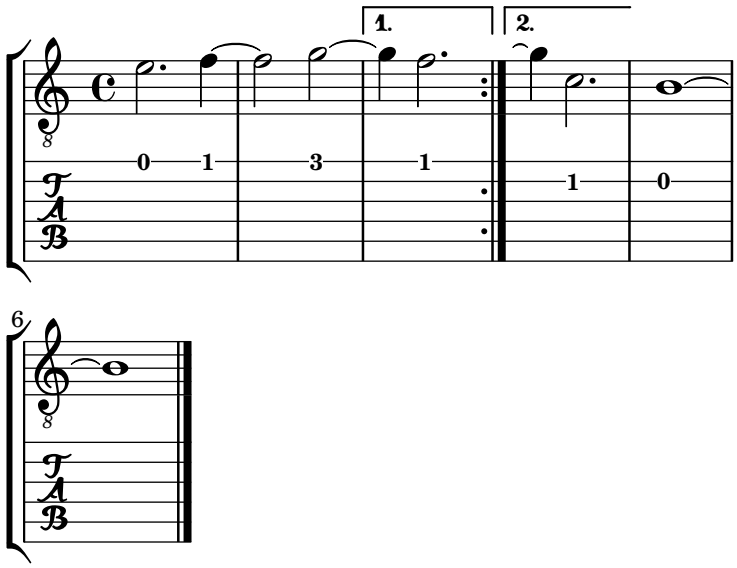


La commande \hideSplitTiedTabNotes permet d'éviter d'imprimer ces cases entre parenthèses.

```
ties = \relative {
  \repeat volta 2 {
    e'2. f4~
    2 g2~ }
  \alternative {
    { g4 f2. }
    { g4\repeatTie c,2. }
  }
  b1~
  \break
  b1
  \bar "|."
}

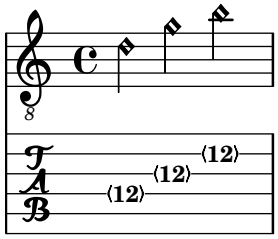
\score {
  <<
    \new StaffGroup  <<
      \new Staff {
        \clef "treble_8"
        \ties
      }
      \new TabStaff {
      \hideSplitTiedTabNotes
        \ties
      }
    >>
  >>
  \layout {
```

```
  indent = 0
  ragged-right = ##t
  }
}
```



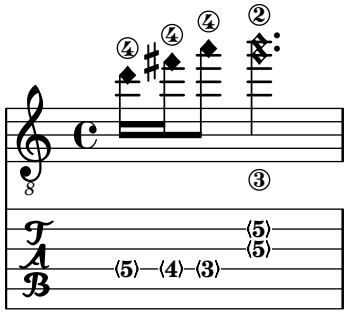
Les indications de sons harmoniques ainsi que les glissandos peuvent être ajoutés aux tablatures.

```
\layout { \omit Voice.StringNumber }
firstHarmonic = {
  d'4\4\harmonic
  g'4\3\harmonic
  b'2\2\harmonic
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \firstHarmonic
    }
    \new TabStaff { \firstHarmonic }
  >>
}
```



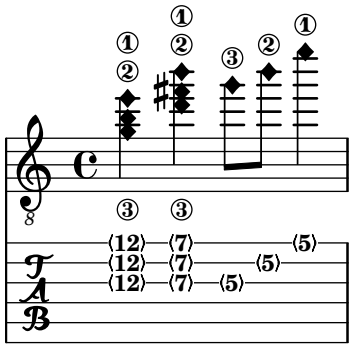
Vous noterez que la commande \harmonic s'attache toujours à une note unique (parfois contenue dans un accord) et non à l'ensemble d'un accord. Ceci ne fonctionne donc que pour une harmonique au douzième fret d'une corde à vide. Toute autre harmonique devrait être défini directement par LilyPond. Vous pouvez l'y aider en indiquant la case où le doigt viendrait se placer sur le manche.

```
fretHarmonics = {
  \harmonicByFret 5 d16\4
  \harmonicByFret 4 d16\4
  \harmonicByFret 3 d8\4
  \harmonicByFret 5 <g\3 b\2>2.
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \fretHarmonics
    }
    \new TabStaff { \fretHarmonics }
  >>
}
```



Une harmonique peut accessoirement se calculer à partir de la longueur de corde par rapport au doigté de cet harmonique.

```
ratioHarmonics = {
  \harmonicByRatio #1/2 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/3 <g\3 b\2 e'\1>4
  \harmonicByRatio #1/4 { g8\3 b8\2 e'4\1 }
}
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \ratioHarmonics
    }
    \new TabStaff { \ratioHarmonics }
  >>
}
```



Des désinences peuvent s'ajouter en notation de tablature. Elles s'indiquent par adjonction d'un \^ à la note ou à l'accord de départ, et se terminent à la note ou l'accord suivant. Différents styles sont disponibles : le style par défaut imprime une courbe terminée par une flèche vers le haut ou le bas, le style `'hold` une ligne horizontale discontinue, `'pre-bend` une ligne verticale terminée en flèche, et `'pre-bend-hold` une ligne verticale terminée en flèche suivie d'une horizontale discontinue.

Les commandes \bendHold, \preBend et \preBendHold sont des raccourcis pour saisir le style de désinence.

```
bend-styles = {
  <>^"default"
  f'4\^ g'4\^ f'2

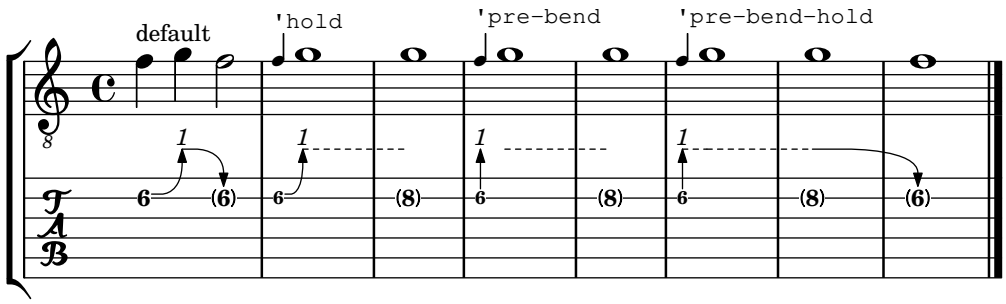
  <>^\markup \typewriter "'hold"
  \grace f'4\^ g'1\bendHold \^ g'1

  <>^\markup \typewriter "'pre-bend"
  \grace f'4\preBend \^ g'1\bendHold \^ g'1

  <>^\markup \typewriter "'pre-bend-hold"
  \grace f'4\preBendHold \^ g'1\bendHold \^ g'1\^ f'

  \bar "|."
}

\score {
  \new StaffGroup
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \bend-styles
    }
    \new TabStaff \bend-styles
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabStaff
      minimumFret = 5
    }
  }
}
```



Les cordes à vide ne font habituellement pas l'objet d'une chute ou d'un saut. Il faut, pour qu'une corde soit concernée, basculer sa propriété `bend-me` à `#t`, et à `#f` pour exclure les autres notes de la désinence.

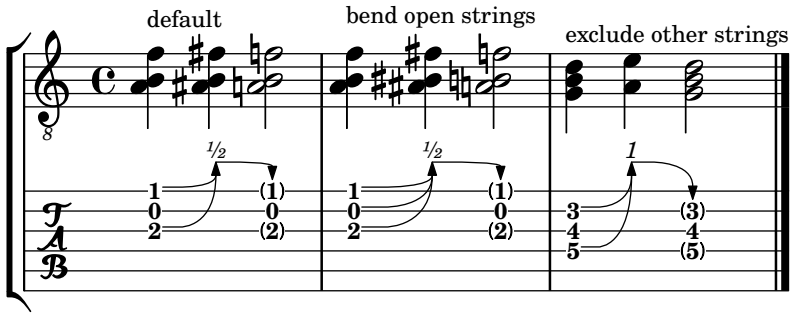
```
mus = {
  <>^"default"
  <a b f'>4\^
  <ais b fis'>\^
  <a b f'>2

  <>^"bend open strings"
  <a \tweak bend-me ##t b f'>4\^
  <ais \tweak bend-me ##t bis fis'>\^
  <a b f'>2

  <>^"exclude other strings"
  <g \tweak bend-me ##f b\3 d'>4\^
  <a e'\2 >\^
  <g \tweak bend-me ##f b\3 d'>2

  \bar "|."
}

\score {
  \new StaffGroup
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \mus
    }
    \new TabStaff \mus
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
  }
}
```



Dans une succession de chutes ou sauts, la désinence de départ pourra nécessiter un réglage particulier de `details.successive-level`. La fonction `bendStartLevel`, qui prend en argument un entier, est là pour le gérer.

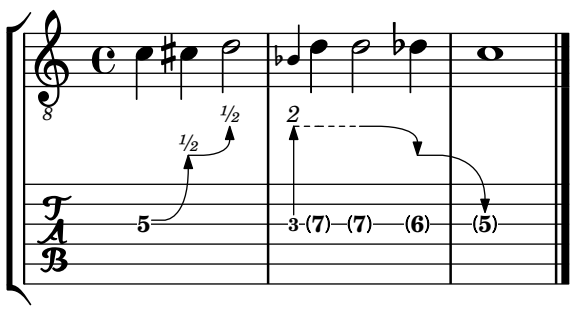
```
printNext = -\tweak details.target-visibility ##t \etc

mus = {
  c'4\3\^ cis'\3 \^ d'2\3

  \grace bes4\3\preBendHold \bendStartLevel 2 \printNext \^
  d'4\3\bendHold \^ d'2\3\^ des'4\3 \^ c'1\3

  \bar "|."
}

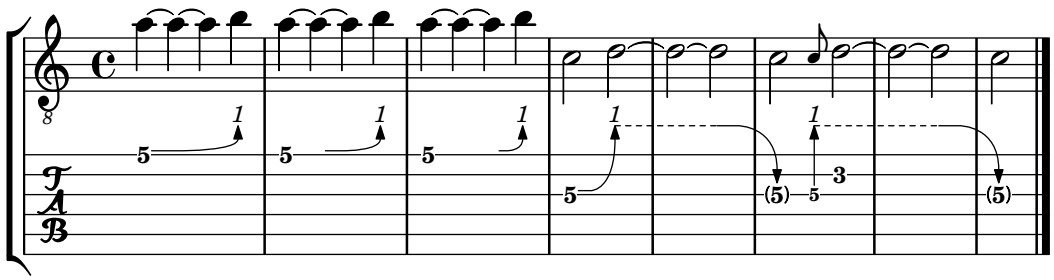
\score {
  \new StaffGroup
  <<
    \new Staff {
      \override TextScript.font-size = -2
      \clef "G_8"
      \mus
    }
    \new TabStaff \mus
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
  }
}
```



L'extension du BendSpanner se termine par défaut sur la note ou l'accord qui suit, même en présence d'une liaison de prolongation. Un élément NoteColumn particulier peut être sauté à l'aide d'un \skipNC. Un groupe de NoteColumns peut se sauter en le faisant précéder d'un \skipNCs et suivre d'un \endSkipNCs.

```
bends-with-ties-and-skips = {
  a'4~\^ \skipNC a'4~ \skipNC a'4 b'4
  a'4~ a'4~\^ \skipNC a'4 b'4
  a'4~ a'4~ a'4\^ b'4
  c'2\^ d'~ \bendHold \^ \skipNC d'~ d'\^ c'
  \grace { c'8-\preBendHold \^ }
  \skipNCs d'2~ d'2~ \endSkipNCs d'\^ c'2
  \bar "|."
}

\score {
  \new StaffGroup
  <<
    \new Staff {
      \clef "G_8"
      \bends-with-ties-and-skips
    }
    \new TabVoice \bends-with-ties-and-skips
  >>
  \layout {
    \context {
      \Voice
      \omit StringNumber
    }
    \context {
      \TabStaff
      minimumFret = 3
      restrainOpenStrings = ##t
    }
  }
}
```



Commandes prédéfinies

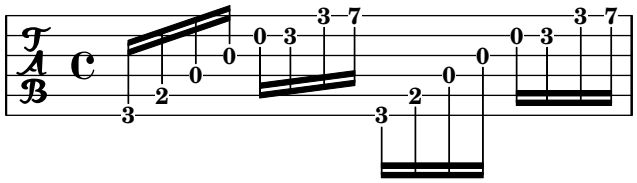
\skipNCs, \skipNC, \endSkipNCs.

Morceaux choisis

Hampes et ligatures en mode tablature

La direction des hampes se gère dans les tablatures tout comme en notation traditionnelle. Les ligatures peuvent être mises à l'horizontale comme le montre cet exemple.

```
\new TabStaff {
  \relative c {
    \tabFullNotation
    g16 b d g b d g b
    \stemDown
    \override Beam.concaveness = 10000
    g,,16 b d g b d g b
  }
}
```



Polyphonie en mode tablature

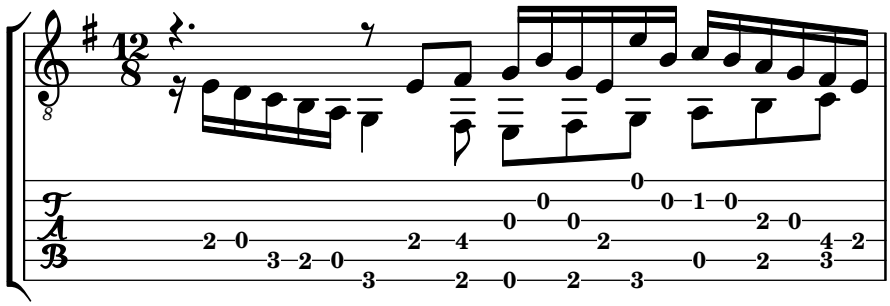
Une section polyphonique s'obtient dans un TabStaff de la même manière que dans une portée normale.

```
upper = \relative c' {
  \time 12/8
  \key e \minor
  \voiceOne
  r4. r8 e, fis g16 b g e e' b c b a g fis e
}

lower = \relative c {
  \key e \minor
  \voiceTwo
  r16 e d c b a g4 fis8 e fis g a b c
}

\score {
  \new StaffGroup = "tab with traditional" <<
    \new Staff = "guitar traditional" <<
      \clef "treble_8"
      \new Voice = "upper" \upper
      \new Voice = "lower" \lower
    >>

    \new TabStaff = "guitar tab" <<
      \new TabVoice = "upper" \upper
      \new TabVoice = "lower" \lower
    >>
  >>
}
```



Harmoniques sur corde à vide en tablature

Voici comment obtenir des harmoniques sur corde à vide (harmoniques naturelles) dans une tablature.

```
openStringHarmonics = {
  \textSpannerDown
  \override TextSpanner.staff-padding = 3
  \override TextSpanner.dash-fraction = 0.3
  \override TextSpanner.dash-period = 1

  % first harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "1st harm. "
  \harmonicByFret 12 e,2\6\startTextSpan
  \harmonicByRatio #1/2 e,\6\stopTextSpan

  % second harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "2nd harm. "
  \harmonicByFret 7 e,\6\startTextSpan
  \harmonicByRatio #1/3 e,\6
  \harmonicByFret 19 e,\6
  \harmonicByRatio #2/3 e,\6\stopTextSpan
  %\harmonicByFret 19 < e,\6 a,\5 d\4 >
  %\harmonicByRatio #2/3 < e,\6 a,\5 d\4 >

  % third harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "3rd harm. "
  \harmonicByFret 5 e,\6\startTextSpan
  \harmonicByRatio #1/4 e,\6
  \harmonicByFret 24 e,\6
  \harmonicByRatio #3/4 e,\6\stopTextSpan
  \break

  % fourth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "4th harm. "
  \harmonicByFret 4 e,\6\startTextSpan
  \harmonicByRatio #1/5 e,\6
  \harmonicByFret 9 e,\6
  \harmonicByRatio #2/5 e,\6
  \harmonicByFret 16 e,\6
```

```
  \harmonicByRatio #3/5 e,\6\stopTextSpan

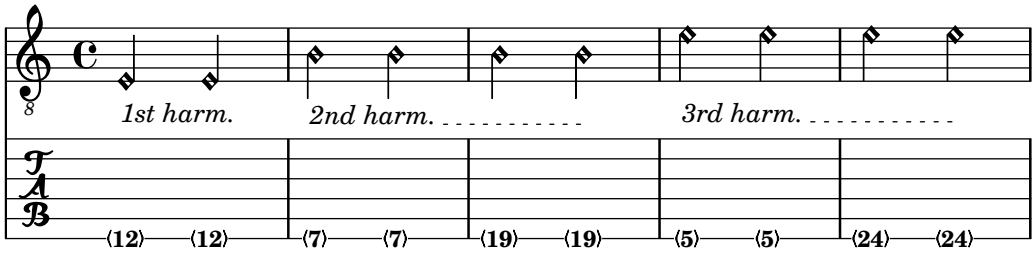
  % fifth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "5th harm. "
  \harmonicByFret 3 e,\6\startTextSpan
  \harmonicByRatio #1/6 e,\6\stopTextSpan
  \break

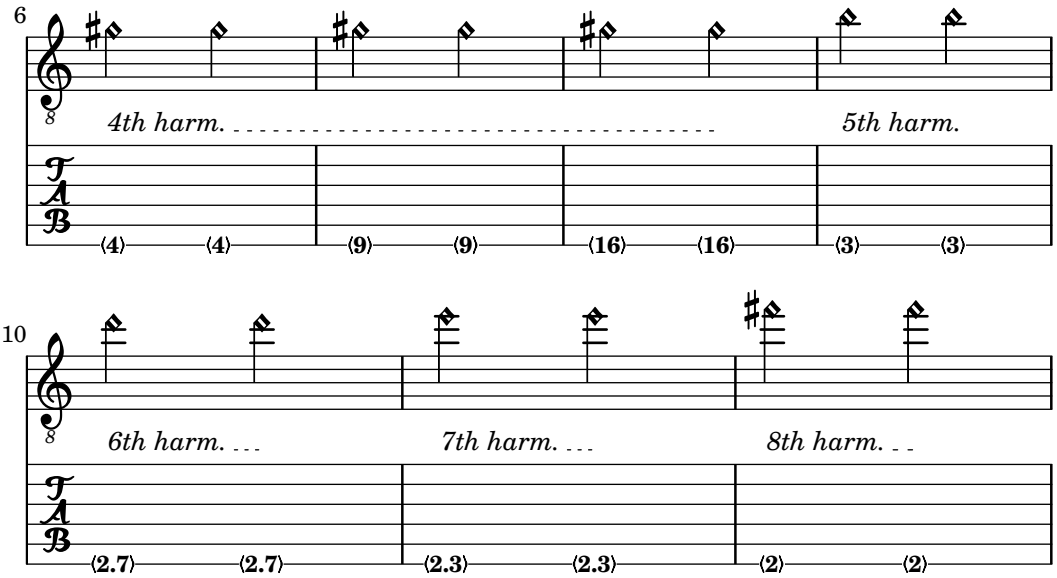
  % sixth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "6th harm. "
  \harmonicByFret 2.7 e,\6\startTextSpan
  \harmonicByRatio #1/7 e,\6\stopTextSpan

  % seventh harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "7th harm. "
  \harmonicByFret 2.3 e,\6\startTextSpan
  \harmonicByRatio #1/8 e,\6\stopTextSpan

  % eighth harmonic
  \override TextSpanner.bound-details.left.text =
    \markup\small "8th harm. "
  \harmonicByFret 2 e,\6\startTextSpan
  \harmonicByRatio #1/9 e,\6\stopTextSpan
}

\score {
  <<
    \new Staff \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \openStringHarmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \openStringHarmonics
      }
    }
  >>
}
```





Harmoniques et tablature

Le code suivant indique comment imprimer des harmoniques dan une tablature.

```
pinchedHarmonics = {
  \textSpannerDown
  \override TextSpanner.bound-details.left.text =
    \markup {\halign #-0.5 \teeny "PH" }
  \override TextSpanner.style = #'dashed-line
  \override TextSpanner.dash-period = 0.6
  \override TextSpanner.bound-details.right.attach-dir = 1
  \override TextSpanner.bound-details.right.text =
    \markup { \draw-line #'(0 . 1) }
  \override TextSpanner.bound-details.right.padding = -0.5
}

harmonics = {
  % artificial harmonics (AH)
  \textLengthOn
  <\parenthesize b b'\harmonic>4_\markup { \teeny "AH 16" }
  <\parenthesize g g'\harmonic>4_\markup { \teeny "AH 17" }
  <\parenthesize d' d''\harmonic>2_\markup { \teeny "AH 19" }

  % pinched harmonics (PH)
  \pinchedHarmonics
  <a'\harmonic>2\startTextSpan
  <d''\harmonic>4
  <e'\harmonic>4\stopTextSpan

  % tapped harmonics (TH)
  <\parenthesize g\4 g'\harmonic>4_\markup { \teeny "TH 17" }
  <\parenthesize a\4 a'\harmonic>4_\markup { \teeny "TH 19" }
  <\parenthesize c'\3 c''\harmonic>2_\markup { \teeny "TH 17" }

  % touch harmonics (TCH)
  a4( <e''\harmonic>2. )_\markup { \teeny "TCH" }
```

```
}

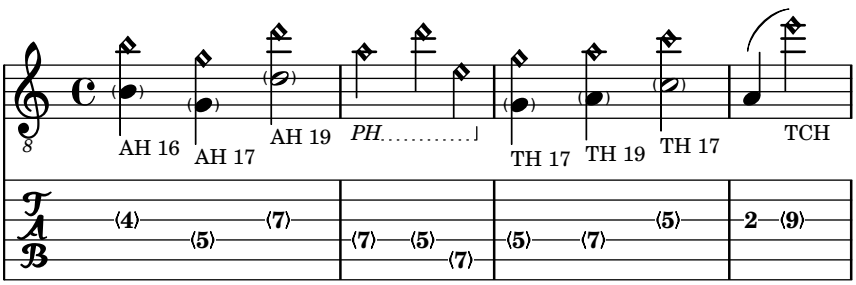
frettedStrings = {
  % artificial harmonics (AH)
  \harmonicByFret 4 g4\3
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 g2\3

  % pinched harmonics (PH)
  \harmonicByFret 7 d2\4
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 a4\5

  % tapped harmonics (TH)
  \harmonicByFret 5 d4\4
  \harmonicByFret 7 d4\4
  \harmonicByFret 5 g2\3

  % touch harmonics (TCH)
  a4 \harmonicByFret 9 g2.\3
}

\score {
  <<
    \new Staff
    \with { \omit StringNumber } {
      \new Voice {
        \clef "treble_8"
        \harmonics
      }
    }
    \new TabStaff {
      \new TabVoice {
        \frettedStrings
      }
    }
  >>
}
```
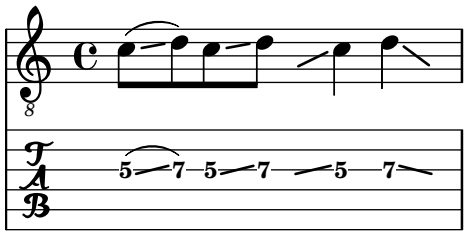


Glissando et tablature

Un glissando s'indique dans un `TabStaff` tout comme dans un `Staff`.

```
slides = {
  c'8\3(\glissando d'8\3)
  c'8\3\glissando d'8\3
  \hideNotes
  \grace { g16\glissando }
  \unHideNotes
  c'4\3
  \afterGrace d'4\3\glissando {
  \stemDown \hideNotes
  g16 }
  \unHideNotes
}

\score {
  <<
    \new Staff { \clef "treble_8" \slides }
    \new TabStaff { \slides }
  >>

  \layout {
    \context {
      \Score
      \override Glissando.minimum-length = 4
      \override Glissando.springs-and-rods =
                  #ly:spanner::set-spacing-rods
      \override Glissando.thickness = 2
      \omit StringNumber
      % or:
      %\override StringNumber.stencil = ##f
    }
  }
}
```



Glissando d'accords et tablature

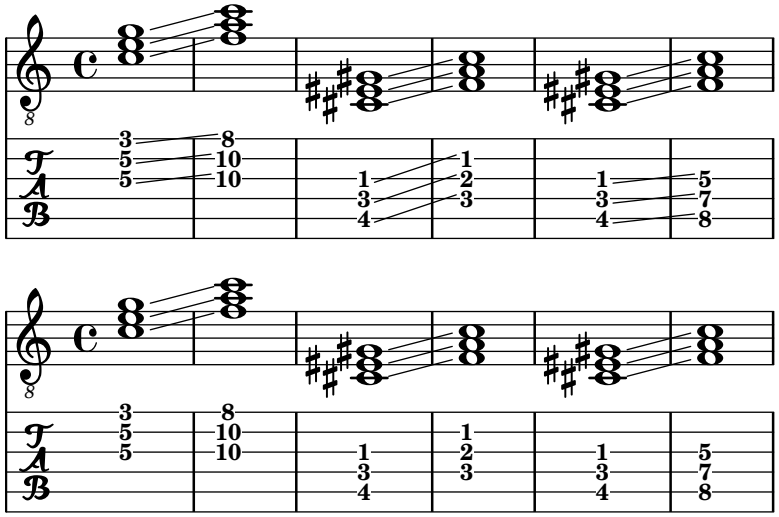
Un glissando sur des accords s'indique dans un `TabStaff` de la même manière que dans un `Staff`, à ceci près que nous pourrions avoir besoin des numéros de corde afin de déterminer correctement les frets d'arrivée.

```
myMusic = \relative c' {
  <c e g>1 \glissando <f a c>
  <cis, eis gis>1 \glissando <f a c>
  <cis eis gis>1 \glissando <f a c\3>
}

\score {
```

```
  <<
    \new Staff {
      \clef "treble_8"
      \omit StringNumber
      \myMusic
    }
    \new TabStaff \myMusic
  >>
}

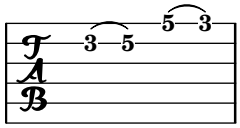
\score {
  <<
    \new Staff {
      \clef "treble_8"
      \omit StringNumber
      \myMusic
    }
    \new TabStaff \with { \override Glissando.style = #'none } {
      \myMusic
    }
  >>
}
```



Hammer on et pull off

Hammer-on et *pull-off* peuvent s'indiquer par des liaisons.

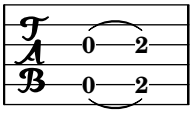
```
\new TabStaff {
  \relative c' {
    d4( e\2)
    a( g)
  }
}
```



Hammer on et pull off gérés par les voix

L'arc des *hammer-on* et *pull-off* est ascendant dans les voix une et trois, et descendant dans les voix deux et quatre.

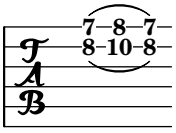
```
\new TabStaff {
  \relative c' {
    << { \voiceOne g2( a) }
    \\ { \voiceTwo a,( b) }
    >> \oneVoice
  }
}
```



Hammer on et pull off avec accords

Dans le cadre de notes en accord, les *hammer-on* et *pull-off* sont indiqués par un arc simple. Vous obtiendrez néanmoins un arc double en réglant la propriété `doubleSlurs` sur `#t`.

```
\new TabStaff {
  \relative c' {
    % chord hammer-on and pull-off
    \set doubleSlurs = ##t
    <g' b>8( <a c> <g b>)
  }
}
```



Voir aussi

Manuel de notation : Section 3.3.1 [Glissando], page 181, Section 7.1.9 [Hampes], page 306, Section 11.1.3 [Harmoniques], page 441, Section 5.1.2 [Répétition d'accords], page 223, Section 4.1.1 [Répétitions explicites], page 193.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

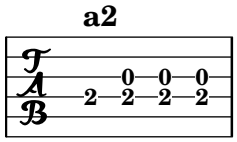
Référence des propriétés internes : Section "Beam" dans *Référence des propriétés internes*, Section "TabNoteHead" dans *Référence des propriétés internes*, Section "TabStaff" dans *Référence des propriétés internes*, Section "TabVoice" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les accords ne subissent aucun traitement particulier ; de ce fait, la sélection automatique des cordes peut attribuer une même corde pour deux notes différentes de l'accord.

Afin que `\partCombine` fonctionne avec des tablatures, on doit ajouter au contexte `TabStaff` des voix fantômes :

```
melodia = \partCombine { e4 g g g } { e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodia }
  >>
>>
```



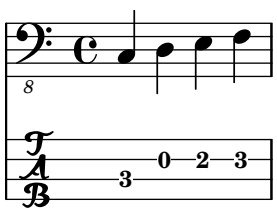
Le support des modes de jeu propres à la guitare se limite aux sons harmoniques et aux glissandos.

12.1.4 Tablatures personnalisées

Avec LilyPond, la case correspondant à une note jouée sur une corde donnée est calculée automatiquement. Pour ce faire, l'accordage doit être spécifié. L'accordage des cordes est donné par la propriété `stringTunings`.

LilyPond possède des accordages prédéfinis pour le banjo, la mandoline, la guitare et la guitare basse ainsi que le ukulele et les cordes d'orchestre. LilyPond calcule automatiquement la transposition correspondant à ces accordages. L'exemple ci-dessous est pour guitare basse, il sonnera donc une octave en dessous de ce qui est écrit.

```
<<
  \new Voice \with {
    \omit StringNumber
  } {
    \clef "bass_8"
    \relative {
      c,4 d e f
    }
  }
  \new TabStaff \with {
    stringTunings = #bass-tuning
  } {
    \relative {
      c,4 d e f
    }
  }
>>
```



L'accordage par défaut est `guitar-tuning_`; il correspond à l'accordage standard d'une guitare : mi la ré sol si mi (EADGBE). D'autres accordages prédéfinis sont disponibles : `guitar-`

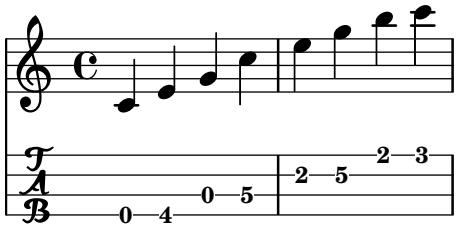
open-g-tuning, mandolin-tuning et banjo-open-g-tuning. Les accordages prédéfinis sont répertoriés dans le fichier ly/string-tunings-init.ly.

LilyPond vous permet de créer n'importe quel accordage. L'accordage du contexte en cours se détermine à l'aide de la fonction \stringTuning. Celle-ci prend deux arguments : une représentation symbolique qui gardera l'accordage en mémoire, et une construction d'accord définissant la hauteur des différentes cordes. Les hauteurs fournies s'expriment impérativement en mode absolu – voir Section 1.1.1 [Hauteurs avec octave absolue], page 3. La corde ayant le numéro le plus élevé (généralement la note la plus basse) est mentionnée en premier.

Nous pouvons ainsi définir par exemple l'accordage d'un instrument à quatre cordes accordées do sol ré la, soit en anglais a'', d'', g', et c' :

```
mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
  \new TabStaff {
    #(define custom-tuning #{ \stringTuning <c' g' d'' a''> #})
    \set Staff.stringTunings = #custom-tuning
    \mynotes
  }
>>
```



La propriété stringTunings permet aussi au FretBoards de calculer automatiquement les diagrammes de frets.

L'accordage fait partie de la clé permettant d'identifier les diagrammes prédéfinis – voir Section 12.1.6 [Tablatures prédéfinies], page 479.

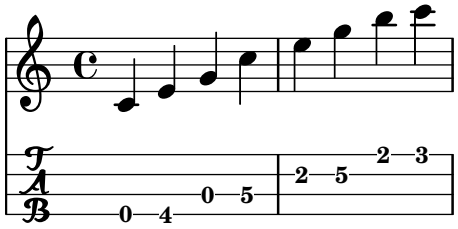
Nous pourrions donc écrire l'exemple précédent ainsi :

```
custom-tuning = \stringTuning <c' g' d'' a''>

mynotes = {
  c'4 e' g' c'' |
  e''4 g'' b'' c'''
}

<<
  \new Staff {
    \clef treble
    \mynotes
  }
```

```
    \new TabStaff {
      \set TabStaff.stringTunings = #custom-tuning
      \mynotes
    }
  >>
```



L'accordage est constitué, en interne, par une liste Scheme des hauteurs de note correspondant aux cordes à vide, une note pour chaque corde, classée par numéro de corde de 1 à n, où la corde 1 est la plus haute dans la tablature et n la plus basse. Cela revient généralement à classer les cordes de la plus aiguë à la plus grave, mais certains instruments (comme le ukulele) n'ont pas les cordes classées par hauteur.

Chaque hauteur de corde incluse dans un accordage est un objet LilyPond de type *pitch*. Les objets *pitch* sont créés par la fonction `ly:make-pitch` – voir Section "Scheme functions" dans *Référence des propriétés internes*.

La fonction `\stringTuning` permet de créer de tels objets à partir de la saisie d'un accord.

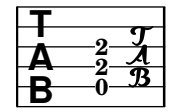
LilyPond calcule automatiquement le nombre de cordes à représenter dans la tablature (`TabStaff`) ainsi que dans le `FretBoard` en comptant le nombre d'éléments définis dans le `stringTunings`.

Les différents contextes `TabStaff` utiliseront par défaut un même accordage personnalisé dès lors que votre fichier comportera une clause

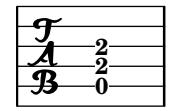
```
\layout {
  \context {
    \TabStaff
    stringTunings = \stringTuning <c' g' d'' a''>
  }
}
```

LilyPond dispose d'une clef de tablature moderne.

```
\new TabStaff {
  \clef moderntab
  <a, e a>1
  \break
  \clef tab
  <a, e a>1
}
```



2



Cette clef moderne prend en charge les tablatures de quatre à sept cordes.

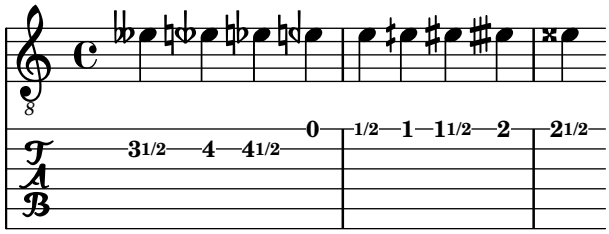
Un TabStaff peut contenir des micro-intervalles tels les quarts de ton, qui interviennent dans les chutes ou sauts. L'assertion supportNonIntegerFret = ##t devra se placer au niveau du contexte Score. Les micro-intervalles ne sont toutefois pas pris en charge dans un contexte FretBoards.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

custom-tuning = \stringTuning <e, a, d ges beh eeh'>

mus = \relative {
  eeses'4
  eeseh
  ees
  eeh
  e
  eih
  eis
  eisih
  eisis
}

<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \with { stringTunings = \custom-tuning } \mus
>>
```



Voir aussi

Manuel de notation : Section 1.1.1 [Hauteurs avec octave absolue], page 3, Section 12.1.6 [Tablatures prédéfinies], page 479.

Fichiers d'initialisation : ly/string-tunings-init.ly, scm/tablature.scm.

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Scheme functions" dans *Référence des propriétés internes*, Section "Tab_note_heads_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

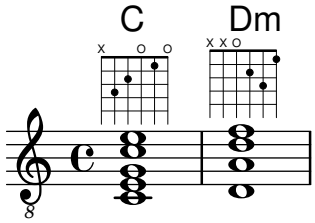
Le calcul automatique des tablatures se révèle inapproprié pour les instruments dont l'ordre des cordes ne correspond pas à l'ordre des hauteurs, comme le ukulele.

12.1.5 Tablatures sous forme d'étiquettes

On peut ajouter des diagrammes d'accord au-dessus de n'importe quelle note, en tant qu'objets \markup. Ces objets contiennent toutes les informations du diagramme d'accord. Il y a pour les définir trois interfaces « diagramme d'accord » (*fret-diagram* en anglais) : *standard*, *terse* et *verbose*. Ces trois interfaces produiront des diagrammes similaires mais permettent des niveaux de personnalisation différents. Des détails à propos des interfaces de type \markup peuvent être trouvés à l'annexe Section A.1.6 [Instrument-specific markup], page 892.

Dans l'interface standard des diagrammes d'accord, on indiquera le numéro de corde et le numéro de case pour chaque point placé sur une corde. Les cordes à vide et étouffées peuvent aussi être indiquées.

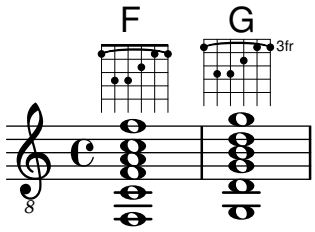
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



Les indications de barré peuvent aussi être ajoutées au diagramme d'accord dans l'interface standard :

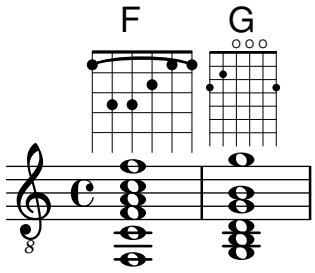
```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram "c:6-1-3;6-3;5-5;4-5;3-4;2-3;1-3;"
    }
  }
```

```
>>
```



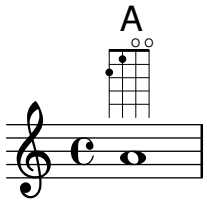
La taille du diagramme d'accord ainsi que le nombre de cases représentées peuvent aussi être modifiés dans l'interface standard.

```
<<
  \new ChordNames {
     \chordmode {
       f1 g
     }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram "s:1.5;c:6-1-1;6-1;5-3;4-3;3-2;2-1;1-1;"
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram "h:6;6-3;5-2;4-o;3-o;2-o;1-3;"
    }
  }
>>
```



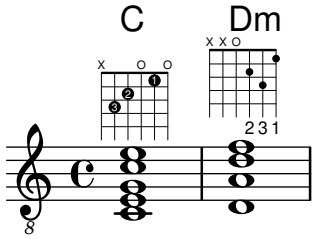
Le nombre de cordes dans les diagrammes d'accord peut être modifié dans l'interface standard pour s'adapter à différents instruments tels que le banjo et le ukulele.

```
<<
  \new ChordNames {
    \chordmode {
      a1
    }
  }
  \new Staff {
    % An 'A' chord for ukulele
    a'1^\markup {
      \fret-diagram "w:4;4-2-2;3-1-1;2-o;1-o;"
    }
  }
>>
```



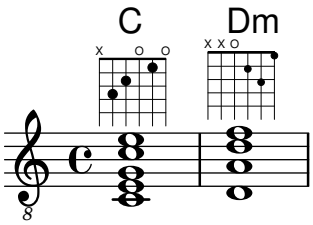
Des indications de doigtés peuvent être ajoutées, et le positionnement de ces doigtés peut être modifié dans l'interface standard.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "f:1;6-x;5-3-3;4-2-2;3-o;2-1-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "f:2;6-x;5-x;4-o;3-2-2;2-3-3;1-1-1;"
    }
  }
>>
```



La taille ainsi que la position des points peuvent être contrôlées dans l'interface standard.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram "d:0.35;6-x;5-3;4-2;3-o;2-1;1-o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram "p:0.2;6-x;5-x;4-o;3-2;2-3;1-1;"
    }
  }
>>
```



Pour les gauchers qui inversent les cordes, LilyPond permet d'imprimer les diagrammes de fret correctement.

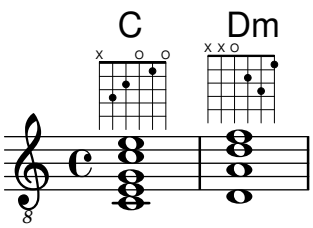
```
\markup
  \center-column {
    "C"
    "(gaucher)"
     \override #`(fret-diagram-details . ((handedness . ,LEFT)))
     \fret-diagram "6-x;5-3-3;4-2-2;3-o;2-1;1-o;"
  }
```

C
(gaucher)



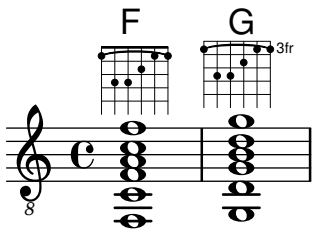
Dans l'interface `fret-diagram-terse`, les numéros de corde sont omis ; les numéros de corde sont induits par la présence de points-virgules. Il y a un point-virgule pour chaque corde du diagramme. Le premier point-virgule correspondant au plus haut numéro de corde, le dernier à la première corde. Les cordes étouffées, les cordes à vide ainsi que les numéros de case peuvent y être indiqués.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3;2;o;1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2;3;1;"
    }
  }
>>
```



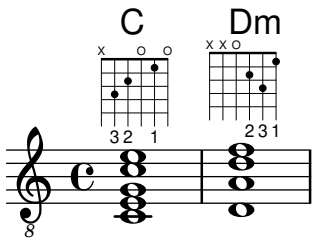
Les indications de barré peuvent être incluses dans l'interface `fret-diagram-terse`.

```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1^\markup {
      \fret-diagram-terse "1-(;3;3;2;1;1-);"
    }
    <g, d g b d' g'>1^\markup {
      \fret-diagram-terse "3-(;5;5;4;3;3-);"
    }
  }
>>
```



Les indications de doigtés peuvent être incluses dans l'interface `fret-diagram-terse`.

```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new Staff {
    \override Voice.TextScript.fret-diagram-details.finger-code =
      #'below-string
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-terse "x;3-3;2-2;o;1-1;o;"
    }
    <d a d' f'>1^\markup {
      \fret-diagram-terse "x;x;o;2-2;3-3;1-1;"
    }
  }
>>
```

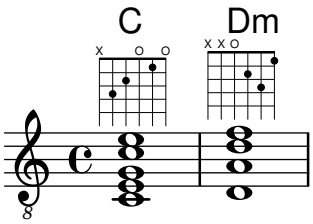


Les autres propriétés des diagrammes d'accord doivent être ajustées en utilisant la commande `\override` dans l'interface `fret-diagram-terse`.

Il n'est possible d'inclure qu'une seule indication par corde dans un *markup* `fret-diagram-terse`. Il faudra, pour en inclure plusieurs, utiliser un *markup* `fret-diagram` ou `fret-diagram-verbose`.

L'interface `fret-diagram-verbose` est au format d'une liste Scheme. Chaque élément de la liste décrit un objet devant être placé dans le diagramme d'accord.

```
<<
    \new ChordNames {
      \chordmode {
        c1 d:m
      }
    }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (place-fret 5 3)
        (place-fret 4 2)
        (open 3)
        (place-fret 2 1)
        (open 1)
      )
    }
    <d a d' f'>1^\markup {
      \fret-diagram-verbose #'(
        (mute 6)
        (mute 5)
        (open 4)
        (place-fret 3 2)
        (place-fret 2 3)
        (place-fret 1 1)
      )
    }
  }
>>
```



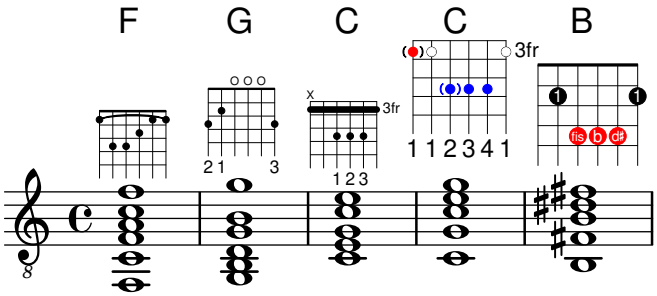
Les indications de doigté et de barré peuvent être décrites dans l'interface `fret-diagram-verbose`. Particularité propre à l'interface `fret-diagram-verbose` : l'indication de capodastre dans le diagramme d'accord. L'indication de capodastre est une petite ligne transversale aux cordes. La case avec le capodastre est la case la plus basse du diagramme d'accord.

Les points d'indication de doigté peuvent se colorier ou être mis entre parenthèses ; la couleur des parenthèses est indépendante de celle du point.

Des *markups* peuvent par ailleurs venir s'insérer dans les points.

```
<<
    \new ChordNames {
      \chordmode {
        f1 g c c b
      }
    }
  \new Staff {
    \clef "treble_8"
    \override Voice.TextScript.fret-diagram-details.finger-code =
      #'below-string
    <f, c f a c' f'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 1)
        (place-fret 5 3)
        (place-fret 4 3)
        (place-fret 3 2)
        (place-fret 2 1)
        (place-fret 1 1)
        (barre 6 1 1)
      )
    }
    <g, b, d g b g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 2)
        (place-fret 5 2 1)
        (open 4)
        (open 3)
        (open 2)
        (place-fret 1 3 3)
      )
    }
    <c e g c' e'>1^\markup {
      \fret-diagram-verbose #'(
        (capo 3)
        (mute 6)
        (place-fret 4 5 1)
        (place-fret 3 5 2)
        (place-fret 2 5 3)
      )
    }
    \override Voice.TextScript.size = 1.4
    <c g c' e' g'>1^\markup {
      \fret-diagram-verbose #'(
        (place-fret 6 3 1 red parenthesized default-paren-color)
        (place-fret 5 3 1 inverted)
        (place-fret 4 5 2 blue parenthesized)
        (place-fret 3 5 3 blue)
        (place-fret 2 5 4 blue)
        (place-fret 1 3 1 inverted)
      )
```

```
    }
    \override Voice.TextScript.size = 1.5
    <b, fis b dis' fis'>1^\markup
      \override #'(fret-diagram-details . ((finger-code . in-dot)))
      \fret-diagram-verbose #`(
        (place-fret 5 2 1)
        (place-fret 4 4 "fis" red)
        (place-fret 3 4 "b" red)
        (place-fret
          2 4
          ,#{ \markup
                \concat {
                  \vcenter "d"
                  \fontsize #-5
                  \musicglyph "accidentals.sharp"} #}
          red)
        (place-fret 1 2 1)
      )
  }
>>
```



Toutes les autres propriétés du diagramme d'accord doivent être indiquées en utilisant la commande `\override` lorsque l'on utilise l'interface `fret-diagram-verbose`.

La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur grâce aux propriétés de l'interface `fret-diagram-interface`. Des détails se trouvent dans Section "fret-diagram-interface" dans *Référence des propriétés internes*. Voir aussi Section "Construction et développement de diagrammes de fret" dans *Morceaux choisis* pour un exemple développé. Pour un diagramme d'accord, les propriétés de l'interface dépendent de `Voice.TextScript`.

Morceaux choisis

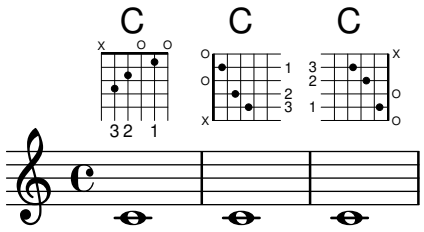
Orientation des diagrammes de fret

Les diagrammes de fret peuvent s'orienter de trois manières différentes. Ils s'aligneront par défaut sur la corde du haut ou le sommet du fret.

```
\include "predefined-guitar-fretboards.ly"

<<
  \chords {
    c1
    c1
    c1
  }
```

```
  \new FretBoards \chordmode {
    c1
    \override FretBoard.fret-diagram-details.orientation =
      #'landscape
    c1
    \override FretBoard.fret-diagram-details.orientation =
      #'opposing-landscape
    c1
  }
  \new Voice {
    c'1
    c'1
    c'
  }
>>
```



Personnalisation des diagrammes de fret

Les propriétés d'un diagramme de fret sont modifiables grâce à la propriété `fret-diagram-details`. Lorsqu'ils sont générés sous forme de `\markup`, rien n'empêche de modifier les diagrammes en jouant sur les réglages de l'objet `Voice.TextScript` ou bien directement sur le *markup*.

```
<<
  \chords { c1 | c | c | d }

  \new Voice = "mel" {
    \textLengthOn
    % Set global properties of fret diagram
    \override TextScript.size = 1.2
    \override TextScript.fret-diagram-details.finger-code = #'in-dot
    \override TextScript.fret-diagram-details.dot-color = #'white

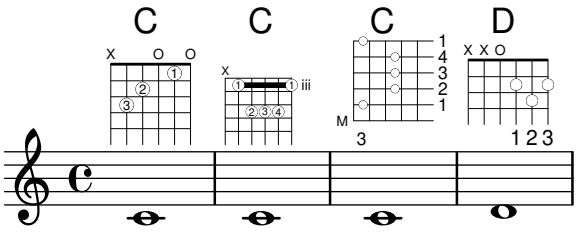
    %% C major for guitar, no barre, using defaults
       % terse style
    c'1^\markup { \fret-diagram-terse "x;3-3;2-2;o;1-1;o;" }

    %% C major for guitar, barred on third fret
       % verbose style
       % size 1.0
       % roman fret label, finger labels below string, straight barre
    c'1^\markup {
      % standard size
      \override #'(size . 1.0) {
        \override #'(fret-diagram-details . (
                    (number-type . roman-lower)
```

```
                    (finger-code . in-dot)
                    (barre-type . straight))) {
          \fret-diagram-verbose #'((mute 6)
                                   (place-fret 5 3 1)
                                   (place-fret 4 5 2)
                                   (place-fret 3 5 3)
                                   (place-fret 2 5 4)
                                   (place-fret 1 3 1)
                                   (barre 5 1 3))
        }
      }
    }

    %% C major for guitar, barred on third fret
       % verbose style
       % landscape orientation, arabic numbers, M for mute string
       % no barre, fret label down or left, small mute label font
    c'1^\markup {
      \override #'(fret-diagram-details . (
                  (finger-code . below-string)
                  (number-type . arabic)
                  (label-dir . -1)
                  (mute-string . "M")
                  (orientation . landscape)
                  (barre-type . none)
                  (xo-font-magnification . 0.4)
                  (xo-padding . 0.3))) {
        \fret-diagram-verbose #'((mute 6)
                                 (place-fret 5 3 1)
                                 (place-fret 4 5 2)
                                 (place-fret 3 5 3)
                                 (place-fret 2 5 4)
                                 (place-fret 1 3 1)
                                 (barre 5 1 3))
      }
    }

    %% simple D chord
       % terse style
       % larger dots, centered dots, fewer frets
       % label below string
    d'1^\markup {
      \override #'(fret-diagram-details . (
                  (finger-code . below-string)
                  (dot-radius . 0.35)
                  (dot-position . 0.5)
                  (fret-count . 3))) {
        \fret-diagram-terse "x;x;o;2-1;3-2;2-3;"
      }
    }
  }
>>
```



Voir aussi

Manuel de notation : Section A.1.6 [Instrument-specific markup], page 892.

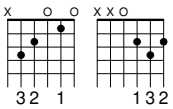
Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

Référence des propriétés internes : Section “fret-diagram-interface” dans *Référence des propriétés internes*.

12.1.6 Tablatures prédéfinies

Les diagrammes d’accord peuvent être affichés en utilisant le contexte `FretBoards`. Par défaut le contexte `FretBoards` affichera des diagrammes d’accord stockés dans une table de correspondance :

```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {
    c1 d
  }
}
```



Les diagrammes d’accord définis par défaut sont dans le fichier `predefined-guitar-fretboards.ly`. Les diagrammes d’accord sont stockés en fonction des notes de l’accord ainsi que de l’accordage (`stringTunings`) utilisé. Le fichier d’initialisation `predefined-guitar-fretboards.ly` contient les diagrammes d’accord prédéfinis uniquement pour l’accordage standard (`guitar-tuning`). Des diagrammes d’accords peuvent être définis pour d’autres instruments ou d’autres accordages en suivant les exemples du fichier `predefined-guitar-fretboards.ly`.

Les diagrammes de fret propres au ukulele se trouvent dans le fichier `predefined-ukulele-fretboards.ly`.

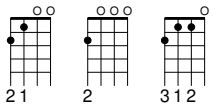
```
\include "predefined-ukulele-fretboards.ly"

myChords = \chordmode { a1 a:m a:aug }

\new ChordNames {
  \myChords
}

\new FretBoards {
  \set Staff.stringTunings = #ukulele-tuning
  \myChords
}
```





Les diagrammes de fret propres à la mandoline se trouvent dans le fichier `predefined-mandolin-fretboards.ly`.

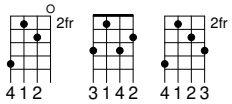
```
\include "predefined-mandolin-fretboards.ly"

myChords = \chordmode { c1 c:m7.5- c:aug }

\new ChordNames {
  \myChords
}

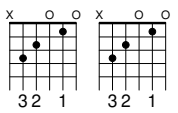
\new FretBoards {
  \set Staff.stringTunings = #mandolin-tuning
  \myChords
}
```

C Cø C+



Les notes des accords peuvent être entrées aussi bien comme musique simultanée qu'en utilisant le mode accord (voir Section 15.1.1 [Généralités sur le mode accords], page 522).

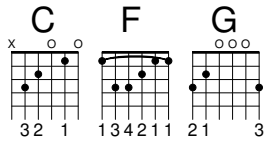
```
\include "predefined-guitar-fretboards.ly"
\new FretBoards {
  \chordmode {c1}
  <c' e' g'>1
}
```



Il est courant d'afficher simultanément le nom des accords et les diagrammes d'accord correspondants. Ceci s'obtient en mettant en parallèle un contexte `ChordNames` et un contexte `FretBoards`, tout en affectant aux deux la même musique.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 f g
}

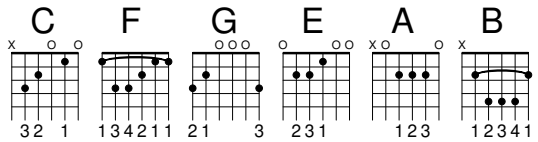
<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



Les diagrammes d'accord prédéfinis sont transposables tant qu'il y a un diagramme correspondant à l'accord transposé dans la base des diagrammes d'accord.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 f g
}

mychordlist = {
  \mychords
  \transpose c e { \mychords }
}
<<
  \new ChordNames {
    \mychordlist
  }
  \new FretBoards {
    \mychordlist
  }
>>
```



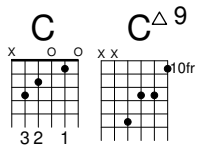
La table des diagrammes d'accord contient huit types d'accord (majeur, mineur, augmenté, diminué, septième de dominante, septième majeure, septième mineure, neuvième de dominante) pour chacune des 17 fondamentales possibles.

La table des diagrammes pour ukulele contient trois types d'accord supplémentaires : sixte majeure, seconde suspendue et quarte suspendue.

Une liste complète des diagrammes d'accords prédéfinis se trouve à l'annexe Section 12.1.6 [Tablatures prédéfinies], page 479. S'il n'y a pas d'entrée dans la table pour un accord donné, le graveur `Fretboard_engraver` calculera le diagramme d'accord en utilisant la fonctionnalité automatique décrite dans Section 12.1.7 [Tablatures automatiques], page 490.

```
\include "predefined-guitar-fretboards.ly"
mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



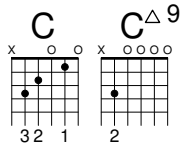
Des diagrammes d'accord peuvent être ajoutés à la table des diagrammes d'accord. Pour ajouter un diagramme d'accord, il faut spécifier l'accord correspondant au diagramme, l'accord utilisé et la définition du diagramme. Cette définition de diagramme peut être aussi bien de type *terse* que *verbose*.

```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c:maj9 }
                        #guitar-tuning
                        "x;3-2;o;o;o;o;"

mychords = \chordmode {
  c1 c:maj9
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



On peut enregistrer différents diagrammes pour un même accord en les définissant à des octaves différentes. Notez qu'il faut un intervalle de deux octaves, le premier servant à la transposition.

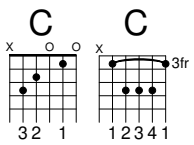
```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table
                        \chordmode { c'' }
                        #guitar-tuning
                        #(offset-fret 2
                           (chord-shape 'bes guitar-tuning))

mychords = \chordmode {
  c1 c''
}

<<
  \new ChordNames {
    \mychords
```

```
  }
  \new FretBoards {
    \mychords
  }
>>
```



En plus des diagrammes d'accord, LilyPond possède une liste interne de carrures d'accord. Les carrures d'accords sont des diagrammes d'accord qui peuvent être transposés le long du manche. Les carrures d'accords peuvent être ajoutées à la liste interne et être ensuite utilisées pour définir des accords prédéfinis. Puisqu'elles sont transposables le long du manche, les carrures d'accord ne contiennent généralement pas de corde à vide. Tout comme les diagrammes d'accord, les carrures d'accord sont définies grâce aux interfaces `fret-diagram-terse` ou `fret-diagram-verbose`.

```
\include "predefined-guitar-fretboards.ly"

% Add a new chord shape

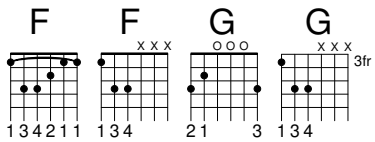
\addChordShape #'powerf #guitar-tuning "1-1;3-3;3-4;x;x;x;"

% add some new chords based on the power chord shape

\storePredefinedDiagram #default-fret-table
                        \chordmode { f'' }
                        #guitar-tuning
                        #(chord-shape 'powerf guitar-tuning)
\storePredefinedDiagram #default-fret-table
                        \chordmode { g'' }
                        #guitar-tuning
                        #(offset-fret 2
                           (chord-shape 'powerf guitar-tuning))

mychords = \chordmode{
  f1 f'' g g''
}

<<
  \new ChordNames {
    \mychords
  }
  \new FretBoards {
    \mychords
  }
>>
```



La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur grâce aux propriétés de l'interface `fret-diagram-interface`. Pour plus d'information, consultez Section "fret-diagram-interface" dans *Référence des propriétés internes*. Voir aussi Section "Construction et développement de diagrammes de fret" dans *Morceaux choisis* pour un exemple développé. Pour un diagramme d'accord donné, les propriétés de l'interface dépendent de `FretBoards.FretBoard`.

Morceaux choisis

Personnalisation de diagrammes de fret

Les propriétés d'un diagramme de fret peuvent se modifier à l'aide de la propriété `'fret-diagram-details`. En matière de diagramme de fret, les adaptations s'appliquent à l'objet `FretBoards.FretBoard`. Un `FretBoards` est comparable à un `Voice` : il s'agit d'un contexte du plus bas niveau, et il n'est donc pas primordial de l'instancier de manière explicite pour adapter ses propriétés.

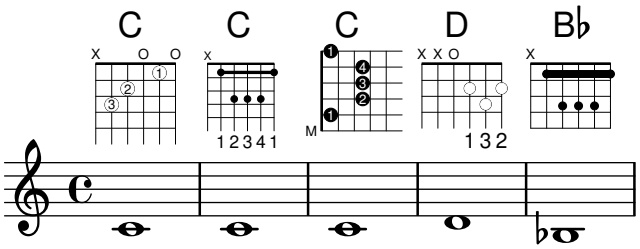
```
\include "predefined-guitar-fretboards.ly"

\storePredefinedDiagram #default-fret-table \chordmode { c' }
                        #guitar-tuning
                        "x;1-1-(;3-2;3-3;3-4;1-1-);"

% shorthand
oo = #(define-music-function
       (grob-path value)
       (list? scheme?)
       #{ \once \override $grob-path = #value #})

<<
  \new ChordNames {
    \chordmode { c1 | c | c | d | bes }
  }
  \new FretBoards {
    % Set global properties of fret diagram
    \override FretBoards.FretBoard.size = 1.2
    \override FretBoard.fret-diagram-details.finger-code = #'in-dot
    \override FretBoard.fret-diagram-details.dot-color = #'white
    \chordmode {
      c
      \oo FretBoard.size #1.0
      \oo FretBoard.fret-diagram-details.barre-type #'straight
      \oo FretBoard.fret-diagram-details.dot-color #'black
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
      c'
      \oo FretBoard.fret-diagram-details.barre-type #'none
      \oo FretBoard.fret-diagram-details.number-type #'arabic
      \oo FretBoard.fret-diagram-details.orientation #'landscape
      \oo FretBoard.fret-diagram-details.mute-string "M"
      \oo FretBoard.fret-diagram-details.label-dir #LEFT
      \oo FretBoard.fret-diagram-details.dot-color #'black
      c'
      \oo FretBoard.fret-diagram-details.finger-code #'below-string
```

```
      \oo FretBoard.fret-diagram-details.dot-radius #0.35
      \oo FretBoard.fret-diagram-details.dot-position #0.5
      \oo FretBoard.fret-diagram-details.fret-count #3
      d
      \oo FretBoard.fret-diagram-details.barre-type #'straight
      \oo FretBoard.fret-diagram-details.finger-code #'none
      \oo FretBoard.fret-diagram-details.dot-radius #0.25
      \oo FretBoard.fret-diagram-details.dot-color #'black
      \oo FretBoard.fret-diagram-details.string-overhang #0.
      \oo FretBoard.fret-diagram-details.barre-thickness #2.
      bes
    }
  }
  \new Voice {
    c'1 | c' | c' | d' | bes
  }
>>
```



Création de diagrammes de fret prédéfinis pour d'autres instruments

La liste des diagrammes standards prédéfinis pour la guitare peut être augmentée d'autres définitions spécifiques à d'autres instruments. Voici comment définir un nouvel accordage ainsi que quelques diagrammes prédéfinis pour le « cuatro vénézuélien ».

Cet exemple illustre aussi la manière d'ajouter des doigtés aux accords ; ils serviront de référence pour la boucle d'accord et seront indiqués dans les diagrammes et le `TabStaff`, mais pas dans la musique.

Ces diagrammes ne peuvent pas être transposés, dans la mesure où ils contiennent des informations sur les cordes. Ceci est amené à évoluer.

```
% Add fretboards for the cuatro.
%
%   Note: This section could be put into a separate file
%         `predefined-cuatro-fretboards.ly`
%         and be \included into each of your compositions.

cuatroTuning = #`(,(ly:make-pitch 0 6 0)
                  ,(ly:make-pitch 1 3 SHARP)
                  ,(ly:make-pitch 1 1 0)
                  ,(ly:make-pitch 0 5 0))

dSix = { <a\4 b\1 d\3 fis\2> }
dMajor = { <a\4 d\1 d\3 fis \2> }
aMajSeven = { <a\4 cis\1 e\3 g\2> }
dMajSeven = { <a\4 c\1 d\3 fis\2> }
```

```
gMajor = { <b\4 b\1 d\3 g\2> }

\storePredefinedDiagram #default-fret-table \dSix
                        #cuatroTuning
                        "o;o;o;o;"
\storePredefinedDiagram #default-fret-table \dMajor
                        #cuatroTuning
                        "o;o;o;3-3;"
\storePredefinedDiagram #default-fret-table \aMajSeven
                        #cuatroTuning
                        "o;2-2;1-1;2-3;"
\storePredefinedDiagram #default-fret-table \dMajSeven
                        #cuatroTuning
                        "o;o;o;1-1;"
\storePredefinedDiagram #default-fret-table \gMajor
                        #cuatroTuning
                        "2-2;o;1-1;o;"

% End of potential include file `predefined-cuatro-fretboards.ly`.


#(set-global-staff-size 16)

primerosNames = \chordmode {
  d:6 d a:maj7 d:maj7
  g
}
primeros = {
  \dSix \dMajor \aMajSeven \dMajSeven
  \gMajor
}

\score {
  <<
    \new ChordNames {
      \set chordChanges = ##t
      \primerosNames
    }

    \new Staff {
      \new Voice \with {
        \remove "New_fingering_engraver"
      }
      \relative c'' {
        \primeros
      }
    }

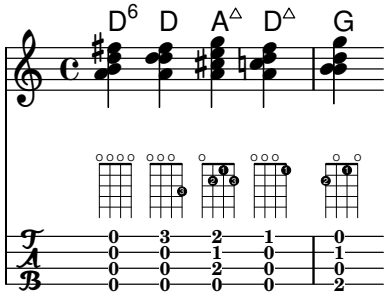
    \new FretBoards {
      \set Staff.stringTunings = #cuatroTuning
%      \override FretBoard
%        #'(fret-diagram-details string-count) = 4
```

```
      \override FretBoard.fret-diagram-details.finger-code = #'in-dot
      \primeros
    }

    \new TabStaff \relative c'' {
      \set TabStaff.stringTunings = #cuatroTuning
      \primeros
    }

  >>

  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        \musicLength 16
    }
  }
  \midi { }
}
```



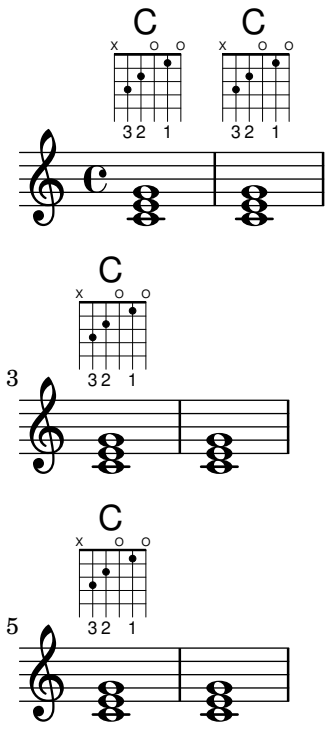
Changement d'accord et diagramme de fret

Vous pouvez opter pour n'imprimer les diagrammes de fret qu'à l'occasion d'un changement d'accord ou de saut de ligne.

```
\include "predefined-guitar-fretboards.ly"

myChords = \chordmode {
  c1 c1 \break
  \set chordChanges = ##t
  c1 c1 \break
  c1 c1
}

<<
  \new ChordNames { \myChords }
  \new FretBoards { \myChords }
  \new Staff { \myChords }
>>
```



Diagrammes de fret alternatifs

Vous pouvez tout à fait créer des tables de diagrammes de fret supplémentaires, notamment dans l'optique d'un fret alternatif pour un accord donné. Avant de pouvoir utiliser un diagramme alternatif, vous devrez avoir alimenté une table à cet effet. Les différents diagrammes seront ajoutés à cette table.

Il peut aussi bien s'agir d'une table vide, que de la recopie d'une table existante. La table servant de base pour les diagrammes prédéfinis est sélectionnée par la propriété \predefinedDiagramTable.

```
\include "predefined-guitar-fretboards.ly"

% Make a blank new fretboard table.
#(define custom-fretboard-table-one
   (make-fretboard-table))

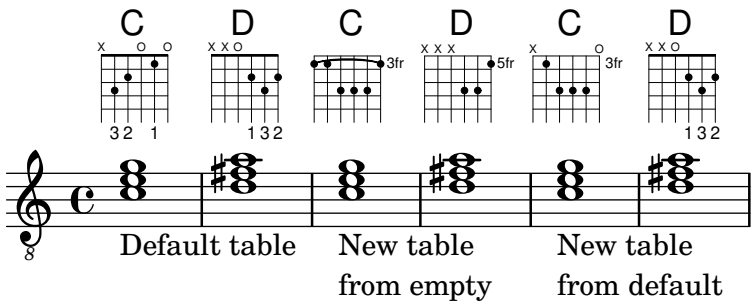
% Make a new fretboard table as a copy of `default-fret-table`.
```

```
#(define custom-fretboard-table-two
   (make-fretboard-table default-fret-table))

% Add a chord to `custom-fretboard-table-one`.
\storePredefinedDiagram #custom-fretboard-table-one
                        \chordmode {c}
                        #guitar-tuning
                        "3-(;3;5;5;5;3-);"

% Add a chord to `custom-fretboard-table-two`.
\storePredefinedDiagram #custom-fretboard-table-two
                        \chordmode {c}
                        #guitar-tuning
                        "x;3;5;5;5;o;"

<<
  \chords {
    c1 | d1 |
    c1 | d1 |
    c1 | d1 |
  }
  \new FretBoards {
    \chordmode {
      \set predefinedDiagramTable = #default-fret-table
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-one
      c1 | d1 |
      \set predefinedDiagramTable = #custom-fretboard-table-two
      c1 | d1 |
    }
  }
  \new Staff {
    \clef "treble_8"
    <<
      \chordmode {
        c1 | d1 |
        c1 | d1 |
        c1 | d1 |
      }
      {
        s1_\markup "Default table" | s1 |
        s1_\markup \column { "New table" "from empty" } | s1 |
        s1_\markup \column { "New table" "from default" } | s1 |
      }
    >>
  }
>>
```



Voir aussi

Manuel de notation : Section 15.1.1 [Généralités sur le mode accords], page 522, Section 12.1.7 [Tablatures automatiques], page 490, Section 12.1.4 [Tablatures personnalisées], page 465, Section 12.1.6 [Tablatures prédéfinies], page 479.

Fichiers d'initialisation : `ly/predefined-guitar-fretboards.ly`, `ly/predefined-guitar-ninth-fretboards.ly`, `ly/predefined-ukulele-fretboards.ly`, `ly/predefined-mandolin-fretboards.ly`.

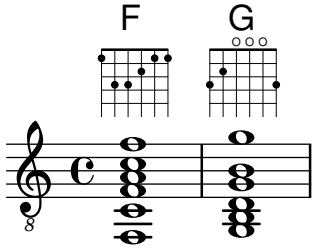
Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "fret-diagram-interface" dans *Référence des propriétés internes*.

12.1.7 Tablatures automatiques

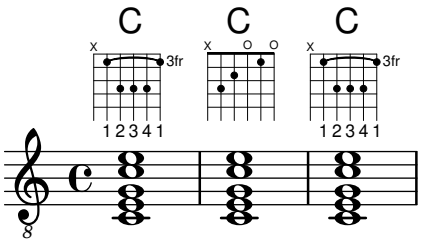
Les diagrammes d'accord peuvent être créés automatiquement ; il suffit d'affecter les notes à un contexte `FretBoards`. Si aucun diagramme prédéfini n'est disponible pour les notes entrées avec l'accordage actuel (`stringTunings`), les cordes et cases correspondant aux notes seront automatiquement calculées.

```
<<
  \new ChordNames {
    \chordmode {
      f1 g
    }
  }
  \new FretBoards {
    <f, c f a c' f'>1
    <g,\6 b, d g b g'>1
  }
  \new Staff {
    \clef "treble_8"
    <f, c f a c' f'>1
    <g, b, d g b g'>1
  }
>>
```



Dans la mesure où aucun diagramme prédéfini n'est chargé par défaut, le calcul automatique des diagrammes d'accord est le comportement par défaut. Dès que les diagrammes par défaut sont chargés, le calcul automatique peut être activé ou désactivé par des commandes prédéfinies :

```
\storePredefinedDiagram #default-fret-table
                        <c e g c' e'>
                        #guitar-tuning
                        "x;3-1-(;5-2;5-3;5-4;3-1-1-);"
<<
  \new ChordNames {
    \chordmode {
      c1 c c
    }
  }
  \new FretBoards {
    <c e g c' e'>1
    \predefinedFretboardsOff
    <c e g c' e'>1
    \predefinedFretboardsOn
    <c e g c' e'>1
  }
  \new Staff {
    \clef "treble_8"
    <c e g c' e'>1
    <c e g c' e'>1
    <c e g c' e'>1
  }
>>
```

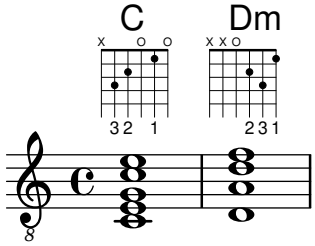


Le calculateur se trouvera parfois incapable de trouver un diagramme d'accord convenable. On peut souvent y remédier en assignant les notes aux cordes. Dans bien des cas, il suffit de placer manuellement une seule note pour que les autres soient alors placées de manière appropriée par le contexte `FretBoards`.

Il est possible d'ajouter des indications de doigté aux diagrammes de fret.

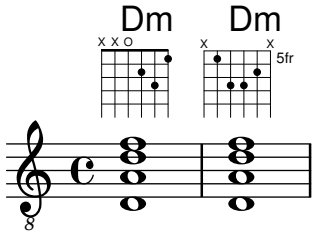
```
<<
  \new ChordNames {
    \chordmode {
      c1 d:m
    }
  }
  \new FretBoards {
    <c-3 e-2 g c'-1 e'>1
    <d a-2 d'-3 f'-1>1
  }
  \new Staff {
```

```
    \clef "treble_8"
    <c e g c' e'>1
    <d a d' f'>1
  }
>>
```



La propriété `minimumFret` permet de définir la case minimale qui servira à calculer les cordes et les cases du contexte `FretBoard`.

```
<<
  \new ChordNames {
    \chordmode {
      d1:m d:m
    }
  }
  \new FretBoards {
    <d a d' f'>1
    \set FretBoards.minimumFret = 5
    <d a d' f'>1
  }
  \new Staff {
    \clef "treble_8"
    <d a d' f'>1
    <d a d' f'>1
  }
>>
```



Les cordes et les cases du contexte `FretBoards` sont liées à la propriété `stringTunings`, qui a là même signification que dans le contexte `TabStaff`. Voir Section 12.1.4 [Tablatures personnalisées], page 465, pour plus d'information sur la propriété `stringTunings`.

La disposition graphique d'un diagramme d'accord peut être modifiée suivant les préférences de l'utilisateur au travers des propriétés de l'interface `fret-diagram-interface`. Voir aussi Section "Construction et développement de diagrammes de fret" dans *Morceaux choisis* pour un exemple. Pour un diagramme d'accord `FretBoards` donné, les propriétés de l'interface dépendent de `FretBoards.FretBoard`.

Commandes prédéfinies

`\predefinedFretboardsOff`, `\predefinedFretboardsOn`.

Voir aussi

Manuel de notation : Section 12.1.4 [Tablatures personnalisées], page 465.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

Référence des propriétés internes : Section “fret-diagram-interface” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

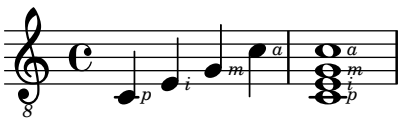
Le calcul automatique des diagrammes de fret se révèle inapproprié pour les instruments dont l'ordre des cordes ne correspond pas à l'ordre des hauteurs.

12.1.8 Doigtés pour la main droite

Les doigtés de main droite *p-i-m-a* doivent être entrés à l'aide de l'instruction `\rightHandFinger` suivie d'un nombre.

> **Note :** Lorsque vous utilisez l'instruction `\rightHandFinger` dans un accord, une espace **doit** précéder la fermeture de la construction par un >.

```
\clef "treble_8"
c4\rightHandFinger 1
e\rightHandFinger 2
g\rightHandFinger 3
c'\rightHandFinger 4
<c\rightHandFinger 1 e\rightHandFinger 2
 g\rightHandFinger 3 c'\rightHandFinger 4 >1
```



Pour plus de clarté, vous pouvez traduire ou abréger la commande `\rightHandFinger`, par exemple en `doigtMainDroite` ou même `MD` en tête du fichier source :

```
MD = #rightHandFinger \etc
```

La plupart des comportements en matière de doigtés pour la main droite, à savoir l'objet `StrokeFinger`, est comparable aux doigtés ordinaires – voir Section 7.1.2 [Doigtés], page 290.

Morceaux choisis

Positionnement des doigtés main droite

Le positionnement des doigtés main droite, grâce à une propriété spécifique, peut se contrôler finement, comme l'indique l'exemple suivant.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"

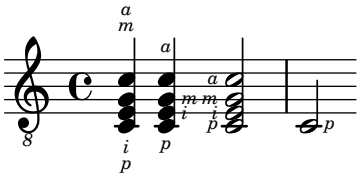
  \set strokeFingerOrientations = #'(up down)
  <c\RH 1 e\RH 2 g\RH 3 c\RH 4 >4

  \set strokeFingerOrientations = #'(up right down)
```

```
  <c\RH 1 e\RH 2 g\RH 3 c\RH 4 >4

  \set strokeFingerOrientations = #'(left)
  <c\RH 1 e\RH 2 g\RH 3 c\RH 4 >2

  \set strokeFingerOrientations = #'(right)
  c\RH 1
}
```

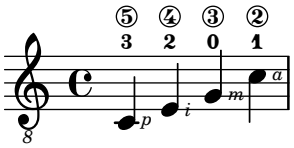


Doigtés, indications de corde, et doigtés main droite

L'exemple suivant illustre comment combiner des doigtés pour la main gauche, des indications de corde et des doigtés pour la main droite.

```
#(define RH rightHandFinger)

\relative c {
  \clef "treble_8"
  <c-3\5\RH 1 >4
  <e-2\4\RH 2 >4
  <g-0\3\RH 3 >4
  <c-1\2\RH 4 >4
}
```



Voir aussi

Morceaux choisis : Section "Cordes frettées" dans *Morceaux choisis*.

Référence des propriétés internes : Section "StrokeFinger" dans *Référence des propriétés internes*.

12.2 Guitare
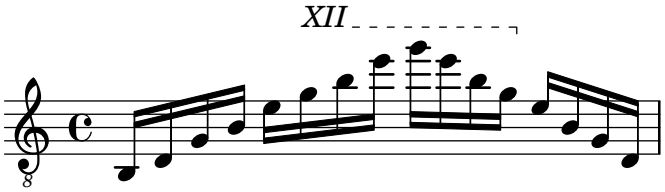
La plupart des aspects en matière de notation pour guitare sont traités dans la partie commune aux instruments frettés. Il subsiste cependant quelques particularités que nous allons maintenant examiner.

Parfois l'utilisateur aimerait créer des documents de type recueil de chansons, où l'on ne trouve que des accords au-dessus des paroles. Dans la mesure où LilyPond est un éditeur de partitions, il n'est pas l'outil optimal pour des documents sans partition. Une meilleure alternative serait de recourir à un traitement de texte, un éditeur de texte ou, pour les utilisateurs expérimentés, un logiciel typographique tel que GuitarTeX.

12.2.1 Indication de la position et du barré

Cet exemple montre comment indiquer les positions et les barrés :

```
\relative {
  \clef "treble_8"
  b,16 d g b e
  \once \override TextSpanner.bound-details.left.text =
    \markup {"XII" \hspace #0.4 }
  \once \override TextSpanner.bound-details.right.text =
    \markup \draw-line #'(0 . -.5)
  \once \override TextSpanner.bound-details.right.padding = -0.65
  \once \override TextSpanner.dash-fraction = 0.4
  \once \override TextSpanner.dash-period = 1.3
  g16\startTextSpan
  b16 e g e b g\stopTextSpan
  e16 b g d
}
```



Une solution plus sophistiquée est disponible ici : Barré de guitare (`https://wiki.lilypond.community/wiki/Guitar_barre`).

Voir aussi
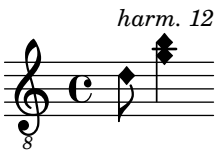
Manuel de notation : Section 8.1.3 [Indication textuelle avec extension], page 321.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*, Section “Signes d’interprétation” dans *Morceaux choisis*.

12.2.2 Indication des harmoniques et notes étouffées

Des têtes de note spéciales peuvent servir à indiquer les notes étouffées et les sons harmoniques. Les sons harmoniques sont souvent détaillés grâce à des indications textuelles.

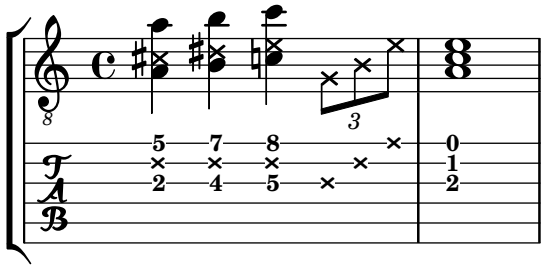
```
\relative {
  \clef "treble_8"
  \override NoteHead.style = #'harmonic-mixed
  d'8^\markup { \italic \fontsize #-2 "harm. 12" } <g b>4
}
```



Les notes étouffées, ou *notes fantômes*, se rencontrent aussi bien sur une portée normale que dans une tablature. Dans l’exemple suivant, ces notes sont affichées à leur hauteur dans une portée normale, avec indication du fret étouffé. Une alternative consiste à utiliser des cordes à vide. Ceci ne fait, en matière de tablature, aucune différence.

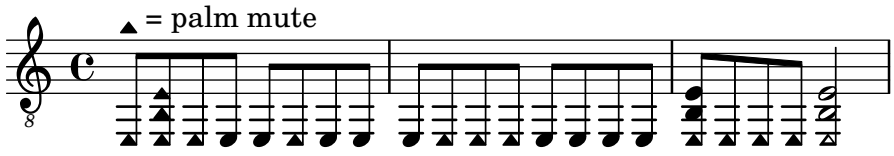
```
music = \relative c' {
  \omit StringNumber

  <a \deadNote cis a'>4
    <b\3 \deadNote dis b'>
    <c\3 \deadNote e\2 c'>
    \deadNotesOn
    \tuplet 3/2 { g8 b e }
    \deadNotesOff
  <a, c e>1
}
\new StaffGroup <<
  \new Staff {
    \clef "treble_8"
    \music
  }
  \new TabStaff {
    \music
  }
>>
```



Le *palm mute*, appelé aussi parfois *chop*, est une technique de jeu pour la guitare électrique ; elle est connue sous le nom de pizzicato par les joueurs de guitare classique. Elle consiste à poser la main droite sur les cordes juste au-dessus du chevalet, de façon à étouffer plus ou moins légèrement les notes. LilyPond permet d'indiquer ce style de jeu en affectant un profil spécifique aux têtes de note.

```
\relative c, {
  \clef "G_8"
    \palmMuteOn
    e8^\markup { \musicglyph "noteheads.s2do"  = palm mute }
    < e b' e > e
    \palmMuteOff
    e e \palmMute e e e |
  e8 \palmMute { e e e } e e e e |
  < \palmMute e b' e >8 \palmMute { e e e } < \palmMute e b' e >2
}
```



Voir aussi

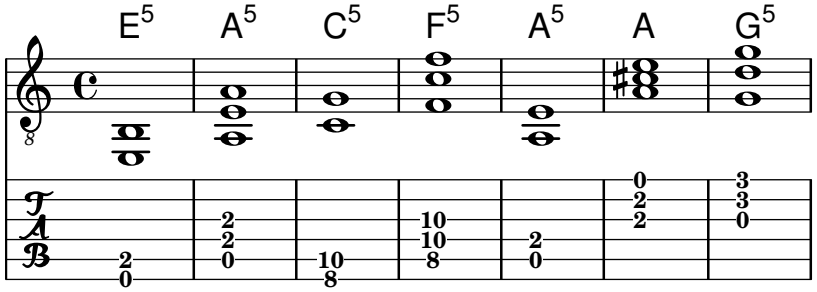
Manuel de notation : Section 1.4.1 [Têtes de note spécifiques], page 45, Section 1.4 [Têtes de note], page 45.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

12.2.3 Indication de *power chord*

Les *power chords* – terme anglais signifiant littéralement « accords de puissance » – s'indiquent aussi bien en mode accord que dans une construction en accord. La quinte est exceptionnellement indiquée dans le chiffrage de ces accords, contrairement aux accords habituels (parfaits ou non) :

```
ChordsAndSymbols = {
  \chordmode {
    e,,1:5
    a,,:5.8
    \set TabStaff.restrainOpenStrings = ##t
    \set minimumFret = 8
    c,:5
    f,:5.8
  }
  \set minimumFret = 2
  \set restrainOpenStrings = ##f
  <a, e> <a cis' e'>
  <g d' g'>
}
\score {
  <<
    \new ChordNames {
    \ChordsAndSymbols
    }
    \new Staff {
      \clef "treble_8"
      \ChordsAndSymbols
    }
    \new TabStaff {
      \ChordsAndSymbols
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section “power chord” dans *Glossaire*.

Manuel de notation : Section 15.1.3 [Extension et altération d'accords], page 524, Section 15.2.1 [Impression des noms d'accord], page 527.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

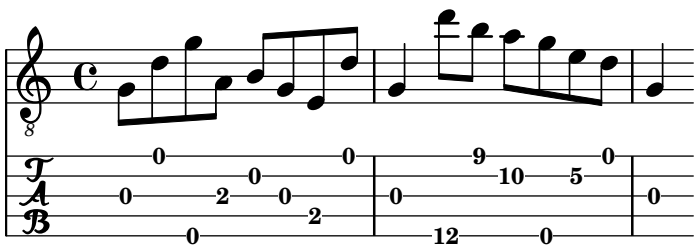
12.3 Banjo

12.3.1 Tablatures pour banjo

LilyPond permet d’écrire des tablatures de base pour le banjo à cinq cordes. Pour ce faire, pensez à utiliser le format de tablature pour banjo, afin d’avoir le bon nombre de cordes et le bon accordage :

```
music = {
  g8 d' g'\5 a b g e d' |
  g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
  g4
}

<<
  \new Staff \with { \omit StringNumber }
  { \clef "treble_8"  \music }
  \new TabStaff \with {
    tablatureFormat = #fret-number-tablature-format-banjo
    stringTunings = #banjo-open-g-tuning
  }
  { \music }
>>
```



LilyPond prend en charge un certain nombre d’accordages courants pour banjo : `banjo-c-tuning` sol do sol si ré (gCGBD), `banjo-modal-tuning` sol ré sol do ré (gDGCD), `banjo-open-d-tuning` la ré fa# la ré (aDF#AD), `banjo-open-dm-tuning` la ré fa la ré (aDFAD), `banjo-double-c-tuning` sol do sol do ré (gCGCD) et `banjo-double-d-tuning` la ré sol ré mi (aDGDE).

Ces accordages peuvent être convertis pour banjo à quatre cordes au moyen de la fonction `four-string-banjo` :

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Voir aussi

Fichiers d’initialisation : `ly/string-tunings-init.ly`.

Morceaux choisis : Section “Cordes frettées” dans *Morceaux choisis*.

12.4 Luth

12.4.1 Tablatures pour luth

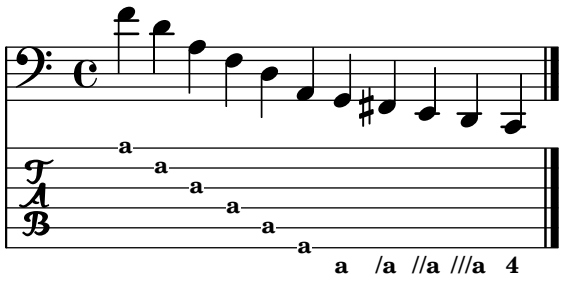
LilyPond prend en charge les tablatures pour le luth.

Les cordes supplémentaires s'ajoutent à l'aide de la commande `additionalBassStrings` qui permet de définir les hauteurs respectives de ces cordes. Elles viendront s'imprimer ainsi au-dessous de la ligne inférieure : a, /a, //a, ///a, 4, 5, etc.

`fret-letter-tablature-format`, et probablement `fretLabels`, fournis en argument à `tablatureFormat`, autoriseront des personnalisations plus avancées.

```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  <<
    \new Staff { \clef bass \cadenzaOn  \m }
    \new TabStaff \m
  >>
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



Problèmes connus et avertissements

L'utilisation de `FretBoards` avec des `additionalBassStrings` n'est pas prise en charge et conduira à un résultat insatisfaisant.

13 Percussions

13.1 Vue d'ensemble des percussions

La notation rythmique sert avant tout aux parties de percussions ou de batterie, mais on peut aussi s'en servir à des fins pédagogiques, pour montrer le rythme d'une mélodie.

13.1.1 Références en matière de notation pour percussions

- Certains instruments à percussion se notent sur une portée rythmique. Vous trouverez des informations détaillées à ce sujet aux rubriques Section 2.3.8 [Gravure de lignes rythmiques], page 101, et Section 6.1.1 [Initialisation de nouvelles portées], page 248.
- Le rendu MIDI des percussions fait l'objet d'une rubrique dédiée : Chapitre 24 [Génération de fichiers MIDI], page 672.

Voir aussi

Manuel de notation : Chapitre 24 [Génération de fichiers MIDI], page 672, Section 2.3.8 [Gravure de lignes rythmiques], page 101, Section 6.1.1 [Initialisation de nouvelles portées], page 248.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

13.1.2 Notation de base pour percussions

Les parties de percussions peuvent être saisies avec le mode `\drummode`, qui est l'équivalent du mode standard utilisé pour les notes à hauteur déterminée. Le moyen plus simple pour saisir une partie de percussion est d'utiliser la commande `\drums`, qui crée un contexte spécifique :

```
\drums {
  hihat4 hh bassdrum bd
}
```



Il s'agit en fait d'un raccourci pour
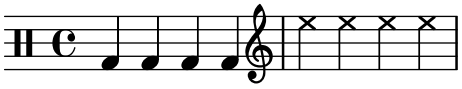
```
\new DrumStaff \drummode {
  hihat4 hh bassdrum bd
  }
```



Chaque instrument de percussion peut avoir, dans le fichier LilyPond, un nom complet et un nom raccourci. Ces noms sont inventoriés à l'annexe Chapitre 24 [Génération de fichiers MIDI], page 672.

Notez bien que l'utilisation de hauteurs (comme un `cis4`) dans un contexte `DrumStaff` déclenchera immanquablement une erreur. Un contexte `DrumStaff` fait automatiquement appel à une clef spécifique ; vous pouvez la spécifier explicitement ou utiliser une autre clef.

```
\drums {
  \clef percussion
  bd4 4 4 4
  \clef treble
  hh4 4 4 4
}
```



Certains instruments à percussion connaissent quelques problèmes quant à leur prise en charge au niveau de la génération de fichiers MIDI ; de plus amples informations sont disponibles à la rubrique Chapitre 24 [Génération de fichiers MIDI], page 672.

Voir aussi

Manuel de notation : Chapitre 24 [Génération de fichiers MIDI], page 672, Section B.15 [Notes utilisées en percussion], page 946.
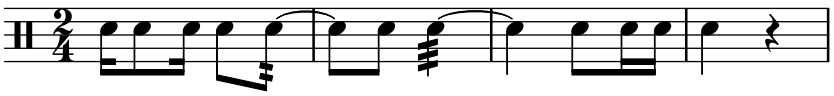
Fichiers d'initialisation : `ly/drumpitch-init.ly`

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

13.1.3 Roulements de tambour
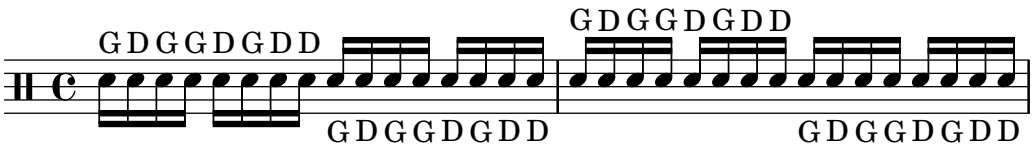
Les roulements de tambour s'indiquent par une triple barre en travers des hampes. Qu'il s'agisse d'une noire ou d'une durée plus longue, cette triple barre s'affiche explicitement. Dans le cas de croches, seules deux barres traversent les hampes (la troisième faisant office de ligature). Si ce roulement s'applique à des notes plus courtes que la croche, LilyPond fait apparaître une seule barre en travers des hampes en supplément du nombre de barres de ligature. Ces différents graphismes s'obtiennent à l'aide d'une notation de trémolo, en suivant les préceptes mentionnés à la rubrique Section 4.2.2 [Répétitions en trémolo], page 217.

```
\drums {
  \time 2/4
  sn16 8 16 8 8:32 ~
  8 8 4:32 ~
  4 8 16 16
  4 r4
}
```



Les coups de baguette peuvent s'indiquer à l'aide de *markups* `"D"` et `"G"` au-dessus ou en dessous des notes comme indiqué à la rubrique Section 36.1 [Direction et positionnement], page 801. Vous devrez peut-être jouer sur la propriété `staff-padding` pour obtenir une ligne de base satisfaisante.

```
\drums {
  \repeat unfold 2 {
    sn16^"G" 16^"D" 16^"G" 16^"G" 16^"D" 16^"G" 16^"D" 16^"D"
    \stemUp
    sn16_"G" 16_"D" 16_"G" 16_"G" 16_"D" 16_"G" 16_"D" 16_"D"
  }
}
```



Voir aussi

Manuel de notation : Section 36.1 [Direction et positionnement], page 801, Section 4.2.2 [Répétitions en trémolo], page 217.

Morceaux choisis : Section “Percussion” dans *Morceaux choisis*.

13.1.4 Hauteurs en percussions

Certains instruments à percussion émettent des hauteurs, comme le xylophone, le vibraphone ou les timbales ; ils utilisent donc des portées classiques. Cette possibilité est abordée dans d’autres parties du manuel.

Voir aussi

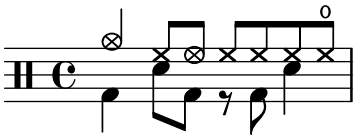
Manuel de notation : Section 24.5 [Gestion des instruments MIDI], page 678.

Morceaux choisis : Section “Percussion” dans *Morceaux choisis*.

13.1.5 Portées de percussion

Une partie de percussions utilisant plusieurs instruments requiert en général une portée de plusieurs lignes, où chaque hauteur sur la portée représente un instrument à percussion. La gravure d’une telle musique requiert que les notes soient situées dans des contextes `DrumStaff` et `DrumVoice`.

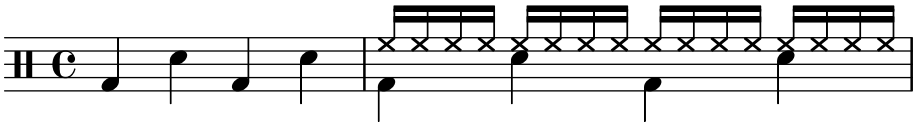
```
up = \drummode {
  crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat
}
down = \drummode {
  bassdrum4 snare8 bd r bd sn4
}
\new DrumStaff <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```



L’exemple ci-dessus montre une notation polyphonique détaillée. La notation polyphonique abrégée peut être employée – voir la rubrique Section “J’entends des voix” dans *Manuel d’initiation* – comme ici :

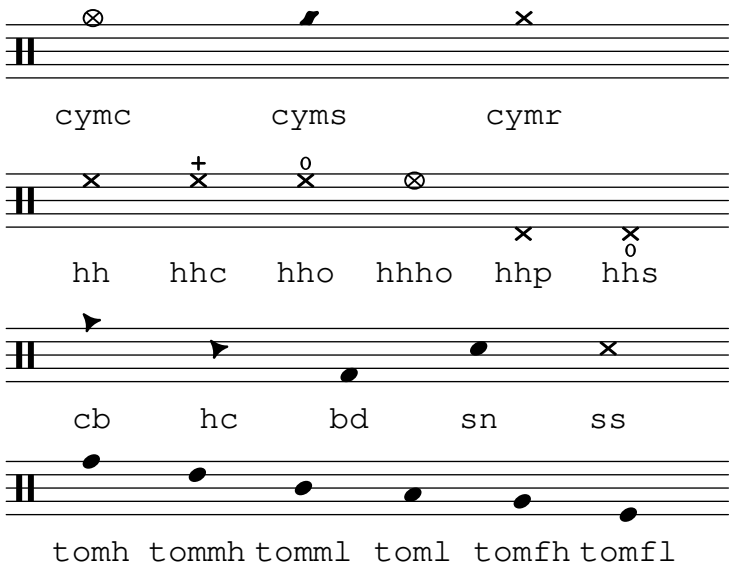
```
\new DrumStaff <<
  \drummode {
    bd4 sn4 bd4 sn4
    << {
      \*16 hh16
    } \\ {
      bd4 sn4 bd4 sn4
    } >>
```

```
  }
>>
```



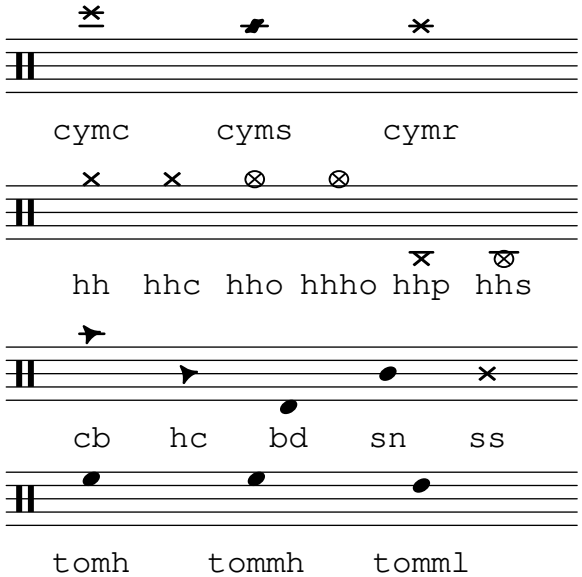
On peut choisir d'autres mises en forme si l'on définit la propriété `drumStyleTable` dans le contexte `DrumVoice`. Quelques variables prédéfinies sont disponibles :

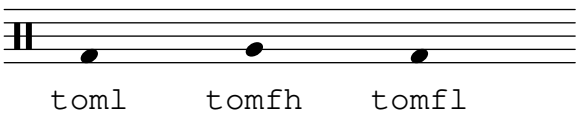
`drums-style`
: La notation par défaut : une batterie standard, sur une portée de cinq lignes.



Le plan de la batterie peut inclure jusqu'à six toms différents. Bien sûr, vous n'êtes pas obligé de tous les utiliser si la musique en prévoit moins ; par exemple, les trois toms des lignes du milieu sont `tommh`, `tomml`, et `tomfh`.

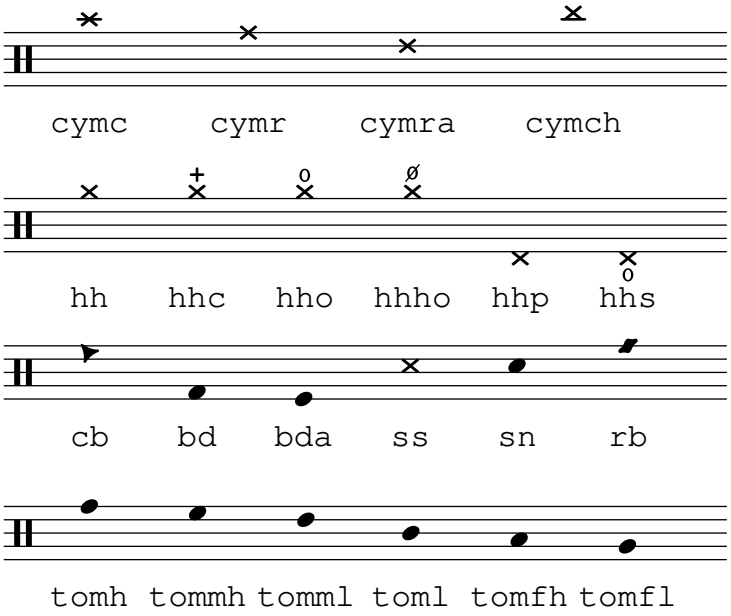
`agostini-drums-style`
: Inventée par le percussionniste français Dante Agostini en 1965, cette notation s'est répandue au-delà de l'hexagone.





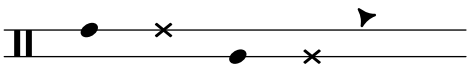
`weinberg-drums-style`

Cette notation est basée sur les travaux de Norman Weinberg, publiés dans son *Guidelines for Drumset Notation.*



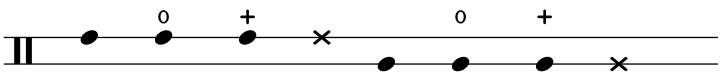
`timbales-style`

Ce style permet de saisir une partie de timbales, sur une portée à deux lignes.



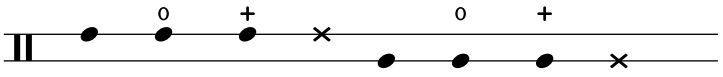
`congas-style`

Ce style produit une portée à deux lignes pour une partie de congas.



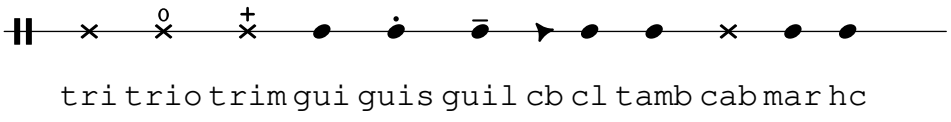
`bongos-style`

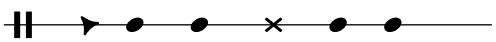
Ce style produit une portée à deux lignes pour une partie de bongos.



`percussion-style`

Ce style permet de saisir toute sorte de percussions sur des portées d'une ligne.





cb cl tamb cab mar hc

Il est par ailleurs possible de définir son propre style, comme indiqué dans Section 13.1.6 [Personnalisation de portées de percussion], page 505.

Voir aussi

Manuel d'initiation : Section "J'entends des voix" dans *Manuel d'initiation*.

Manuel de notation : Section 13.1.6 [Personnalisation de portées de percussion], page 505.

Fichiers d'initialisation : `ly/drumpitch-init.ly`.

Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

13.1.6 Personnalisation de portées de percussion

LilyPond offre la possibilité de construire son propre style de percussion grâce à une définition de la propriété `drumStyleTable`. Les notations existantes peuvent se redéfinir au travers d'une liste associative dans laquelle chaque entrée doit comporter quatre éléments : un nom, le style de tête de note (ou '`()`' pour adopter la valeur par défaut), un éventuel signe d'articulation (ou `#f` dans le cas contraire) et, enfin, le positionnement de la tête de note sur la portée. Cette liste devra alors être convertie en table de hachage à l'aide de la fonction `alist->hash-table`.

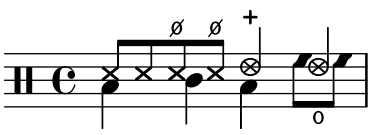
L"entrée pour une articulation peut être soit un nom symbolique de script, soit une paire, auquel cas le premier élément est le nom symbolique du script et le second un indicateur de positionnement de ce script.

[Dans l'exemple ci-dessous, notez bien l'utilisation de l'apostrophe inversée ('`') au début de l'expression Scheme au lieu de l'apostrophe normale ('), ce qui permet de traiter correctement la virgule (',') pour l'évaluation de l'élément qui précède.]

```
#(define mydrums `(
         (bassdrum        ()        #f              -1)
         (snare           ()        #f              0)
         (hihat           cross     #f              1)
         (halfopenhihat   cross     halfopen        1)
         (pedalhihat      xcircle   stopped         2)
         (splashhihat     xcircle   (open . ,DOWN)   2)
         (lowtom          diamond   #f              3)))

up = \drummode { hh8 hh hhho hhho hhp4 hhs }
down = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff \with { drumStyleTable = #(alist->hash-table mydrums) }
  <<
  \new DrumVoice { \voiceOne \up }
  \new DrumVoice { \voiceTwo \down }
>>
```

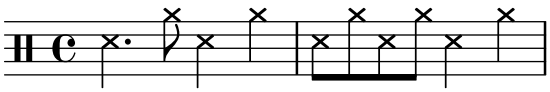


De nouveaux noms peuvent venir s'ajouter à ces notations personnalisées grâce à la variable `drumPitchNames` qui peut être redéfinie en tant que liste associative (ou augmentée par ajout d'une nouvelle liste aux valeurs existantes comme indiqué ci-dessous). Ceci permet par ailleurs de définir des alias : raccourcis pour la saisie de certaines notations.

```
drumPitchNames =
  #(append
    '((leftsnap . sidestick)
      (rightsnap . ridecymbal))
    drumPitchNames)

drumPitchNames.ls = #'sidestick
drumPitchNames.rs = #'ridecymbal

\drums {
  leftsnap4. rightsnap8 leftsnap4 rightsnap
  ls8 rs ls rs ls4 rs
}
```



De la même manière, la propriété `drumPitchTable` associe une hauteur spécifique (autrement dit un son instrumental différent tel que fourni par les fontes sonores MIDI disponibles) à chaque notation. Cette propriété requiert d'être définie sous forme de table de hachage, convertie elle aussi à partir d'une liste associative (enregistrée par défaut en tant que variable `midiDrumPitches`). La redéfinition de ces associations s'effectue comme indiqué ci-dessus, soit en créant l'intégralité d'une liste associative, soit certains de ses membres. L'exemple ci-dessous démontre la manière de créer un jeu entier de notation, avec sa propre syntaxe de saisie, sa notation personnalisée et les instruments MIDI correspondants.
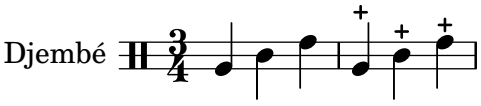
```
drumPitchNames.dbass     = #'dbass
drumPitchNames.dba       = #'dbass  % 'db is in use already
drumPitchNames.dbassmute = #'dbassmute
drumPitchNames.dbm       = #'dbassmute
drumPitchNames.do        = #'dopen
drumPitchNames.dopenmute = #'dopenmute
drumPitchNames.dom       = #'dopenmute
drumPitchNames.dslap     = #'dslap
drumPitchNames.ds        = #'dslap
drumPitchNames.dslapmute = #'dslapmute
drumPitchNames.dsm       = #'dslapmute

#(define djembe-style
  '((dbass       ()   #f        -2)
    (dbassmute   ()   stopped   -2)
    (dopen       ()   #f        0)
    (dopenmute   ()   stopped   0)
    (dslap       ()   #f        2)
    (dslapmute   ()   stopped   2)))

midiDrumPitches.dbass     = g
midiDrumPitches.dbassmute = fis
midiDrumPitches.dopen     = a
midiDrumPitches.dopenmute = gis
midiDrumPitches.dslap     = b
midiDrumPitches.dslapmute = ais
```

```
test = \drummode { dba4 do ds dbm dom dsm }

\score {
  \new DrumStaff \with {
    \override StaffSymbol.line-count = 3
    instrumentName = "Djembé "
    drumStyleTable = #(alist->hash-table djembe-style)
    drumPitchTable = #(alist->hash-table midiDrumPitches)
  } {
    \time 3/4
    \test
  }
  \layout {}
  \midi {}
}
```



Voir aussi

Fichiers d'initialisation : `ly/drumpitch-init.ly`.

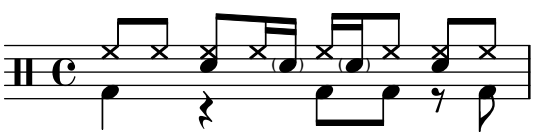
Morceaux choisis : Section "Percussion" dans *Morceaux choisis*.

Référence des propriétés internes : Section "DrumStaff" dans *Référence des propriétés internes*, Section "DrumVoice" dans *Référence des propriétés internes*.

13.1.7 Notes fantômes

Des notes fantômes, qu'on appelle aussi en anglais *dead, muted, silenced ou false* notes, peuvent être créées pour les parties de percussion, grâce à la commande `\parenthesize` décrite à la rubrique Section 7.1.8 [Parenthèses], page 304.

```
\new DrumStaff
<<
  \context DrumVoice  = "1" { s1 }
  \context DrumVoice  = "2" { s1 }
  \drummode {
    <<
      {
        hh8[ 8] <hh sn> hh16
        \parenthesize sn hh
        \parenthesize sn hh8 <hh sn> hh
      } \\
      {
        bd4 r4 bd8 8 r8 bd
      }
    >>
  }
>>
```

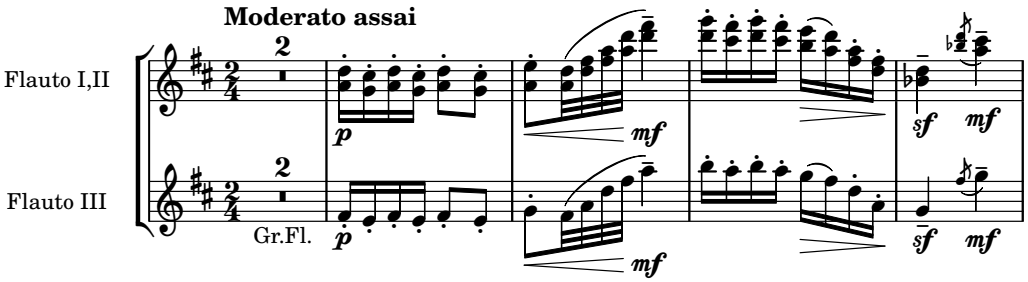


Voir aussi

Manuel de notation : Section 7.1.8 [Parenthèses], page 304.

Morceaux choisis : Section “Percussion” dans *Morceaux choisis*.

14 Instruments à vent



Ce chapitre traite de certains aspects particuliers en matière de notation pour instruments à vent.

14.1 Vue d'ensemble des instruments à vent

Nous allons aborder ici quelques aspects communs à la plupart des instruments à vent.

14.1.1 Références en matière d'instruments à vent

Ce qui caractérise les partitions pour instruments à vent a trait principalement à la respiration et à l'attaque :

- Les respirations s'indiquent par des silences ou des Section 3.2.3 [Signes de respiration], page 179.
- Un jeu *legato* s'indique par des Section 3.2.1 [Liaisons d'articulation], page 174.
- Les différents types d'attaque – *legato*, détaché ou piqué – s'indiquent en principe par des signes d'articulation, parfois agrémentés de liaisons. Voir à ce sujet Section 3.1.1 [Articulations et ornements], page 158, et Section B.13 [Liste des signes d'articulation], page 943.
- Un *Flatterzunge* (trémolo dental ou trémolo en roulant les r) s'indique par une marque de trémolo et une étiquette textuelle attachée à la note concernée. Voir à ce sujet Section 4.2.2 [Répétitions en trémolo], page 217.

D'autres aspects de la notation s'appliquent aussi aux instruments à vent :

- De nombreux instruments à vent sont transpositeurs ; voir Section 1.3.4 [Instruments transpositeurs], page 30.
- Les glissandos sont l'une des caractéristiques du trombone à coulisse, bien que d'autres instruments puissent y parvenir en jouant sur les pistons ou des clés ; consulter Section 3.3.1 [Glissando], page 181.
- Des glissandos harmoniques sont réalisables par les cuivres. Ils sont traditionnellement indiqués par des Section 2.6.1 [Notes d'ornement], page 150.
- Les inflexions en fin de note sont abordées au chapitre Section 3.2.4 [Chutes et sauts], page 180.
- Les « bruitageS » de clé ou de piston s'indiquent souvent par le style `cross` ou des Section 1.4.1 [Têtes de note spécifiques], page 45.
- Les bois peuvent émettre des harmoniques dans le bas de leur registre. On les indique avec un `flageolet` – voir Section B.13 [Liste des signes d'articulation], page 943.
- En ce qui concerne les cuivres, la sourdine s'indique en principe par une étiquette textuelle. Cependant, lorsque les changements sont nombreux et rapides, il est d'usage de recourir aux articulations `stopped` et `open`. Pour de plus amples détails, voir Section 3.1.1 [Articulations et ornements], page 158, et Section B.13 [Liste des signes d'articulation], page 943.
- La sourdine du cor d'harmonie s'indique par un `stopped`. Voir le chapitre Section 3.1.1 [Articulations et ornements], page 158.

Voir aussi

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158, Section 3.2.4 [Chutes et sauts], page 180, Section 3.3.1 [Glissando], page 181, Section 1.3.4 [Instruments transpositeurs], page 30, Section 3.2.1 [Liaisons d'articulation], page 174, Section B.13 [Liste des signes d'articulation], page 943, Section 2.6.1 [Notes d'ornement], page 150, Section 4.2.2 [Répétitions en trémolo], page 217, Section 3.2.3 [Signes de respiration], page 179, Section 1.4.1 [Têtes de note spécifiques], page 45.

Morceaux choisis : Section "Vents" dans *Morceaux choisis*.

14.1.2 Doigtés pour vents

Tout instrument à vent, hormis le trombone à coulisse, fait appel à plusieurs doigts pour produire un son. Les exemples ci-dessous vous donnent un aperçu de différentes manières d'indiquer des doigtés.

La gestion des diagrammes de doigté spécifiques aux bois est abordée plus en profondeur au chapitre Section 14.3.1 [Diagrammes pour bois], page 514.

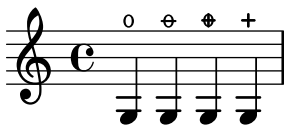
Morceaux choisis

Symboles de doigtés pour instruments à vent

Des symboles spécifiques peuvent être obtenus en combinant les glyphes disponibles, ce qui est tout à fait indiqué en matière d'instrument à vent.

```
lineup =
  \tweak outside-staff-padding #0
  \tweak staff-padding #0
  \tweak padding #0.2
  \tweak parent-alignment-X #CENTER
  \tweak self-alignment-X #CENTER
  \etc

\relative c' {
  g\open
  g\lineup ^\markup \combine
              \musicglyph "scripts.open"
              \musicglyph "scripts.tenuto"
  g\lineup ^\markup \combine
              \musicglyph "scripts.open"
              \musicglyph "scripts.stopped"
  g\stopped
}
```



Doigtés pour flûte à bec

Cet exemple illustre la manière de créer et afficher des indications de doigté pour instrument à vent.

```
% range chart for paetzold contrabass recorder

centermarkup = {
  \once \override TextScript.self-alignment-X = #CENTER
  \once \override TextScript.X-offset = #(lambda (g)
    (+ (ly:self-alignment-interface::centered-on-x-parent g)
       (ly:self-alignment-interface::x-aligned-on-self g)))
}

\new Staff \with {
  \remove "Time_signature_engraver"
  \omit Stem
  \omit Flag
  \consists "Horizontal_bracket_engraver"
} {
  \clef bass
  \set Score.timing = ##f

  f,1*1/4 \glissando

  \clef violin
  gis'1*1/4

  a'4^\markup "1)"

  \centermarkup
    \once \override TextScript.padding = 2
    bes'1*1/4_\markup \override #'(baseline-skip . 1.7) \column {
      \fontsize #-5
      \slashed-digit #0 \finger 1 \finger 2
      \finger 3 \finger 4 \finger 5 \finger 6 \finger 7 }

  b'1*1/4

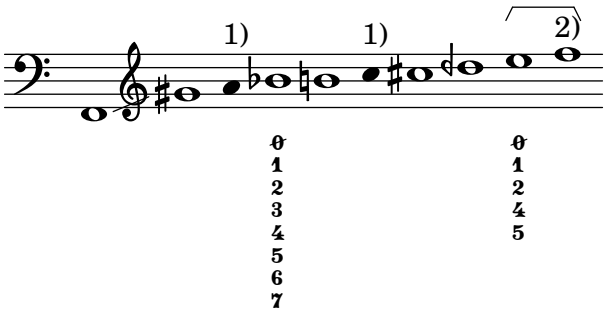
  c''4^\markup "1)"

  cis''1*1/4

  deh''1*1/4

  \centermarkup
    \once \override TextScript.padding = 2
    \once \override Staff.HorizontalBracket.direction = #UP
    e''1*1/4_\markup \override #'(baseline-skip . 1.7) \column {
      \fontsize #-5
      \slashed-digit #0 \finger 1 \finger 2
      \finger 4 \finger 5 } \startGroup

  f''1*1/4^\markup "2)" \stopGroup
}
```



Voir aussi

Manuel de notation : Section 14.3.1 [Diagrammes pour bois], page 514.

Morceaux choisis : Section “Vents” dans *Morceaux choisis*.

14.2 Cornemuse

Voici quelques informations spécifiques à la cornemuse.

14.2.1 Définitions pour la cornemuse

LilyPond inclut des définitions spécifiques destinées à la notation pour cornemuse écossaise ; pour les utiliser, il suffit d’ajouter
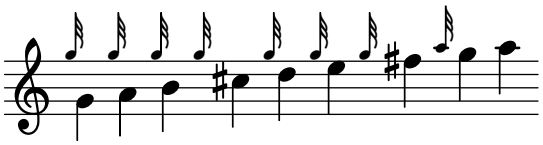
```
\include "bagpipe.ly"
```

en début de fichier. Vous bénéficierez ainsi de commandes courtes pour les appoggiatures spéciales et idiomatiques de la cornemuse. Par exemple, `\taor` est un raccourci pour

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` prend également en charge les définitions de hauteurs pour la cornemuse ; vous n’avez donc pas à vous soucier d’employer `\relative` ou `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



La musique pour cornemuse est traditionnellement écrite en ré majeur. Dans la mesure où c’est la seule tonalité possible, l’usage est de ne pas indiquer l’armure explicitement. À cet effet, pensez à commencer votre partition par `\hideKeySignature` – ou bien `\showKeySignature` si, pour une raison ou pour une autre, vous tenez à afficher l’armure.

Des compositions actuelles peuvent ne pas respecter cette tonalité traditionnelle, auquel cas les do et fa devraient être abaissés en utilisant `c-flat` ou `f-flat` ; ils seront représentés par une note en forme de croix.

Lorsqu’on joue des œuvres moins cérémonieuses que pour une parade ou un défilé, peut survenir un sol aigu, au doux nom de « Piobaireachd », et que l’on indiquera par `g-flat`.

Voir aussi

Morceaux choisis : Section “Vents” dans *Morceaux choisis*.

14.2.2 Exemple pour la cornemuse
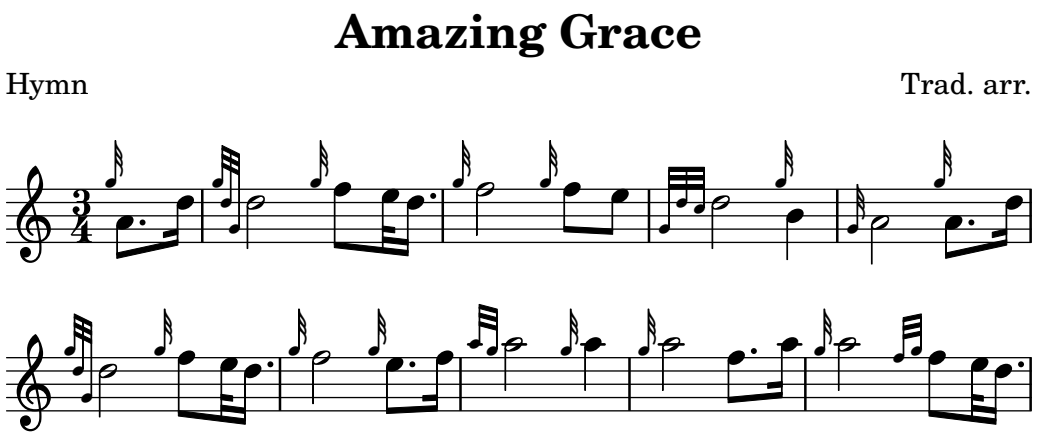
Amazing Amazing Grace : exemple pour cornemuse
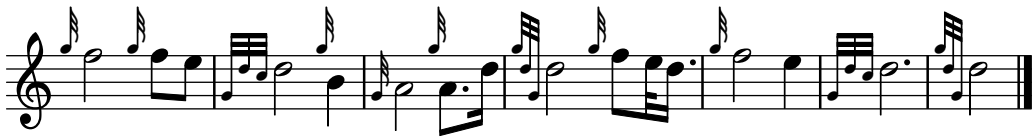
Et voici en guise d'exemple, à quoi ressemble le chant populaire *Amazing Grace*, noté dans l'idiome de la cornemuse.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
  \context { \Score \remove Bar_number_engraver }
}

\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{
  \hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
  \thrwd d2.
  \slurd d2
  \bar "|."
}
```





Voir aussi

Morceaux choisis : Section “Vents” dans *Morceaux choisis*.

14.3 Bois

Nous allons ici nous intéresser aux spécificités de la section des bois.

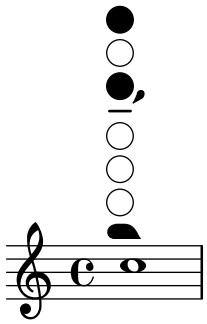
14.3.1 Diagrammes pour bois

Les doigtés pour obtenir une note particulière peuvent s’afficher sous forme graphique. LilyPond dispose de diagrammes pour la plupart des bois, et tout particulièrement les instruments suivants :

- piccolo
- flûte
- hautbois
- clarinette
- clarinette basse
- saxophone
- basson
- contrebasson

Les diagrammes sont générés en tant qu’objet de type *markup* :

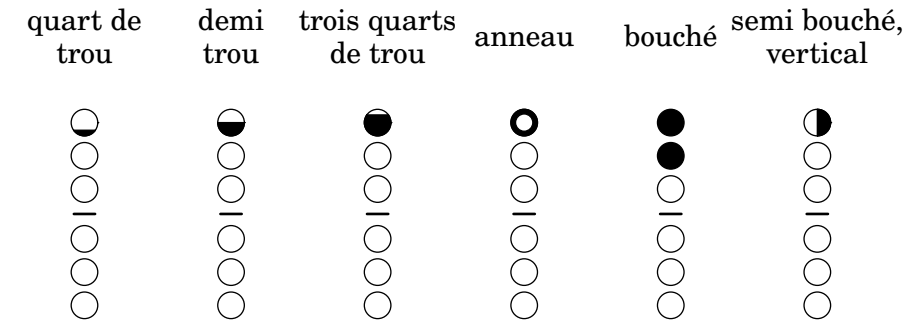
```
c''1^\markup
  \woodwind-diagram #'piccolo #'((lh . (gis))
                                 (cc . (one three))
                                 (rh . (ees)))
```



Les clés ou trous peuvent être partiellement enfoncés ou bouchés, et il est possible de spécifier l’angle de bouchage :

```
\markup \override #'(baseline-skip . 22) \column {
  \override #'(baseline-skip . 2.5) \fill-line {
    ""
    \raise #1.2 \center-column { "quart de" "trou" }
    \raise #1.2 \center-column { "demi" "trou" }
    \raise #1.2 \center-column { "trois quarts" "de trou" }
    "anneau"
    "bouché"
```

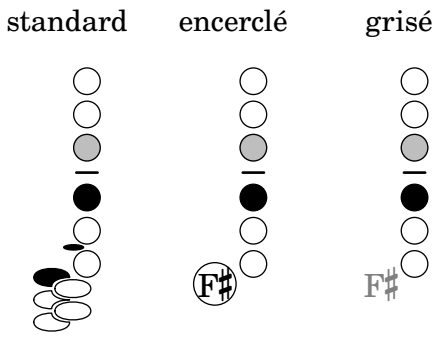
```
    \raise #1.2 \center-column { "semi bouché," "vertical" }
    ""
  }
  \fill-line {
    ""
    \woodwind-diagram #'flute #'((cc . (one1q))
                                 (lh . ())
                                 (rh . ()))
    \woodwind-diagram #'flute #'((cc . (one1h))
                                 (lh . ())
                                 (rh . ()))
    \woodwind-diagram #'flute #'((cc . (one3q))
                                 (lh . ())
                                 (rh . ()))
    \woodwind-diagram #'flute #'((cc . (oneR))
                                 (lh . ())
                                 (rh . ()))
    \woodwind-diagram #'flute #'((cc . (oneF two))
                                 (lh . ())
                                 (rh . ()))
    \override #'(woodwind-diagram-details . ((fill-angle . 90)))
    \woodwind-diagram #'flute #'((cc . (one1h))
                                 (lh . ())
                                 (rh . ()))
    ""
  }
}
```



L'indication du doigté permettant de triller s'obtient en grisant une position ou, de manière non graphique, par un texte encerclé (par défaut) ou grisé :

```
\markup {
  \override #'(baseline-skip . 20)
  \override #'(line-width . 50)
  \column {
    \fill-line {
      ""
      "standard"
      "encerclé"
      "grisé"
      ""
    }
    \fill-line {
```

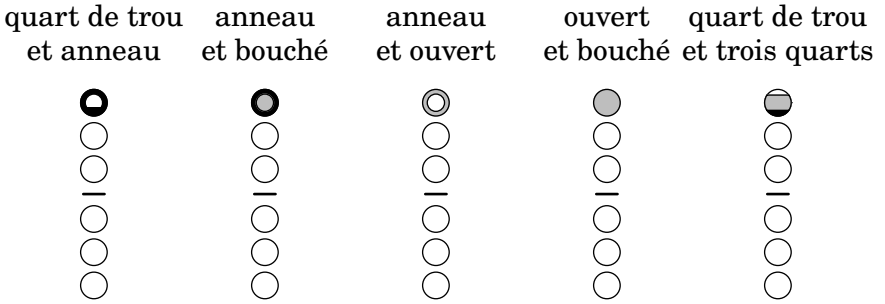
```
      ""
      \woodwind-diagram #'bass-clarinet
                        #'((cc . (threeT four))
                           (lh . ())
                           (rh . (b fis)))
      \override #'(graphical . #f)
      \woodwind-diagram #'bass-clarinet
                        #'((cc . (threeT four))
                           (lh . ())
                           (rh . (b fisT)))
      \override #'(graphical . #f)
      \override #'(woodwind-diagram-details
                   . ((text-trill-circled . #f)))
      \woodwind-diagram #'bass-clarinet
                        #'((cc . (threeT four))
                           (lh . ())
                           (rh . (b fisT)))
      ""
    }
  }
}
```



Certaines combinaisons particulières en matière de trille sont possibles :

```
\markup {
  \override #'(baseline-skip . 22)
  \column {
    \override #'(baseline-skip . 2.5) \fill-line {
      ""
      \center-column { "quart de trou" "et anneau" }
      \center-column { "anneau" "et bouché" }
      \center-column { "anneau" "et ouvert" }
      \center-column { "ouvert" "et bouché" }
      \center-column { "quart de trou" "et trois quarts" }
      ""
    }
    \fill-line {
      ""
      \woodwind-diagram #'flute #'((cc . (one1qTR))
                                   (lh . ())
                                   (rh . ()))
      \woodwind-diagram #'flute #'((cc . (oneTR))
                                   (lh . ())
```

```
                                (rh . ()))
      \woodwind-diagram #'flute #'((cc . (oneRT))
                                (lh . ())
                                (rh . ()))
      \woodwind-diagram #'flute #'((cc . (oneT))
                                (lh . ())
                                (rh . ()))
      \woodwind-diagram #'flute #'((cc . (one1qT3q))
                                (lh . ())
                                (rh . ()))
      ""
    }
  }
}
```



Bien que cela ne produise pas de notation, vous pouvez obtenir la liste de toutes les possibilités pour un instrument donné, en utilisant dans un fichier les instructions `#(print-keys-verbose 'vent)` – affichage à l'écran – ou `#(print-keys-verbose 'vent (current-error-port))` – génération d'un fichier de journalisation.

De nouveaux diagrammes sont réalisables, bien que ceci requiert de maîtriser le langage Scheme et n'est pas à la portée de tous les utilisateurs. Des gabarits sont contenus dans les fichiers `scm/define-woodwind-diagrams.scm` et `scm/display-woodwind-diagrams.scm`.

Morceaux choisis

Liste des diagrammes de doigtés pour bois

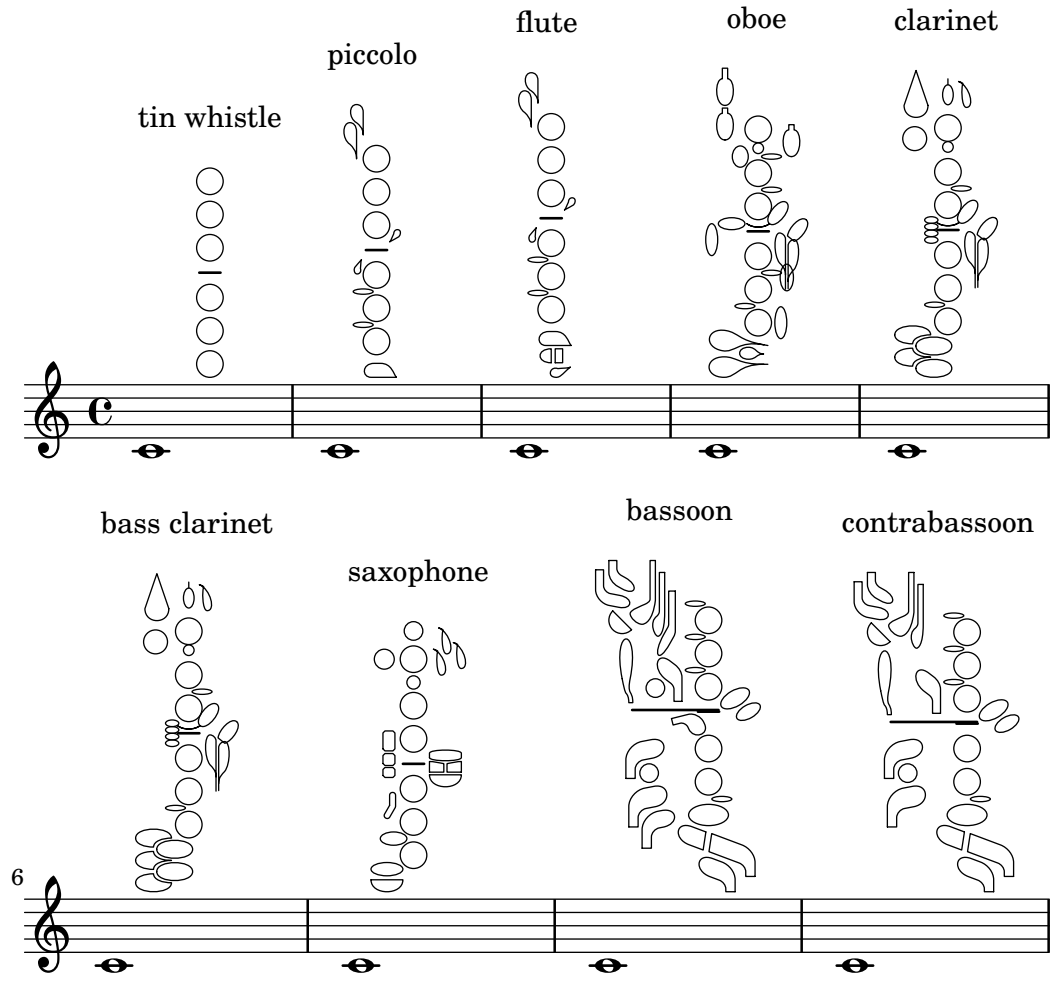
Voici les différents instruments à vent de la section des bois pour lesquels LilyPond peut, à ce jour, afficher des doigtés.

```
\relative c' {
  \textLengthOn
  c1^\markup \center-column { "tin whistle"
                              " "
                              \woodwind-diagram #'tin-whistle #'() }
  c1^\markup \center-column { "piccolo"
                              " "
                              \woodwind-diagram #'piccolo #'() }
  c1^\markup \center-column { "flute"
                              " "
                              \woodwind-diagram #'flute #'() }
  c1^\markup \center-column { "oboe"
                              " "
```

```
                                \woodwind-diagram #'oboe #'() }
  c1^\markup \center-column { "clarinet"
                                " "
                                \woodwind-diagram #'clarinet #'() }
  \break

  c1^\markup \center-column { "bass clarinet"
                                " "
                                \woodwind-diagram #'bass-clarinet #'() }
  c1^\markup \center-column { "saxophone"
                                " "
                                \woodwind-diagram #'saxophone #'() }
  c1^\markup \center-column { "bassoon"
                                " "
                                \woodwind-diagram #'bassoon #'() }
  c1^\markup \center-column { "contrabassoon"
                                " "
                                \woodwind-diagram #'contrabassoon #'() }
}

\paper {
  system-system-spacing.padding = 5
}
```

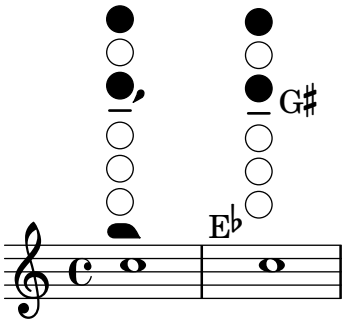


Ajout de texte à un diagramme de doigté

Dans certains cas, vous pouvez opter pour l'affichage textuel d'une clé située à côté d'un trou plutôt que sa représentation graphique.

```
\relative c'' {
  \textLengthOn
  c1^\markup
    \woodwind-diagram #'piccolo
                      #'((cc . (one three))
                         (lh . (gis))
                         (rh . (ees)))

  c^\markup
    \override #'(graphical . #f)
      \woodwind-diagram #'piccolo
                        #'((cc . (one three))
                           (lh . (gis))
                           (rh . (ees)))
}
```

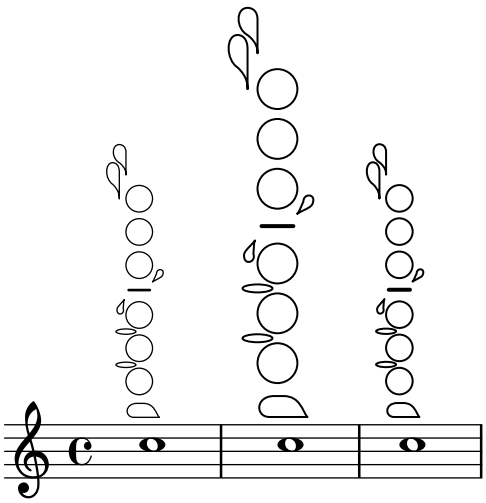


Modification de la taille d'un diagramme pour bois

La taille et l'épaisseur des diagrammes de doigté pour bois est modifiable à souhait.

```
\relative c'' {
  \textLengthOn
  c1^\markup
       \woodwind-diagram #'piccolo #'()

  c^\markup \override #'(size . 1.5)
      \woodwind-diagram #'piccolo #'()
  c^\markup \override #'(thickness . 0.15)
      \woodwind-diagram #'piccolo #'()
}
```
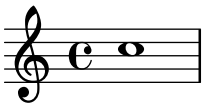


Liste des différents diagrammes de doigtés pour bois

Le code suivant permet d'obtenir une liste de toutes les possibilités en matière de doigtés pour bois, tels qu'ils sont définis dans le fichier `scm/define-woodwind-diagrams.scm`. Cette liste sera produite dans le fichier de journalisation, mais pas sous forme de musique. Pour un affichage en console, supprimez la partie `(current-error-port)` des commandes.

```
#(print-keys-verbose 'piccolo (current-error-port))
#(print-keys-verbose 'flute (current-error-port))
#(print-keys-verbose 'flute-b-extension (current-error-port))
#(print-keys-verbose 'tin-whistle (current-error-port))
#(print-keys-verbose 'oboe (current-error-port))
#(print-keys-verbose 'clarinet (current-error-port))
#(print-keys-verbose 'bass-clarinet (current-error-port))
#(print-keys-verbose 'low-bass-clarinet (current-error-port))
#(print-keys-verbose 'saxophone (current-error-port))
#(print-keys-verbose 'soprano-saxophone (current-error-port))
#(print-keys-verbose 'alto-saxophone (current-error-port))
#(print-keys-verbose 'tenor-saxophone (current-error-port))
#(print-keys-verbose 'baritone-saxophone (current-error-port))
#(print-keys-verbose 'bassoon (current-error-port))
#(print-keys-verbose 'contrabassoon (current-error-port))

\score {c''1}
```



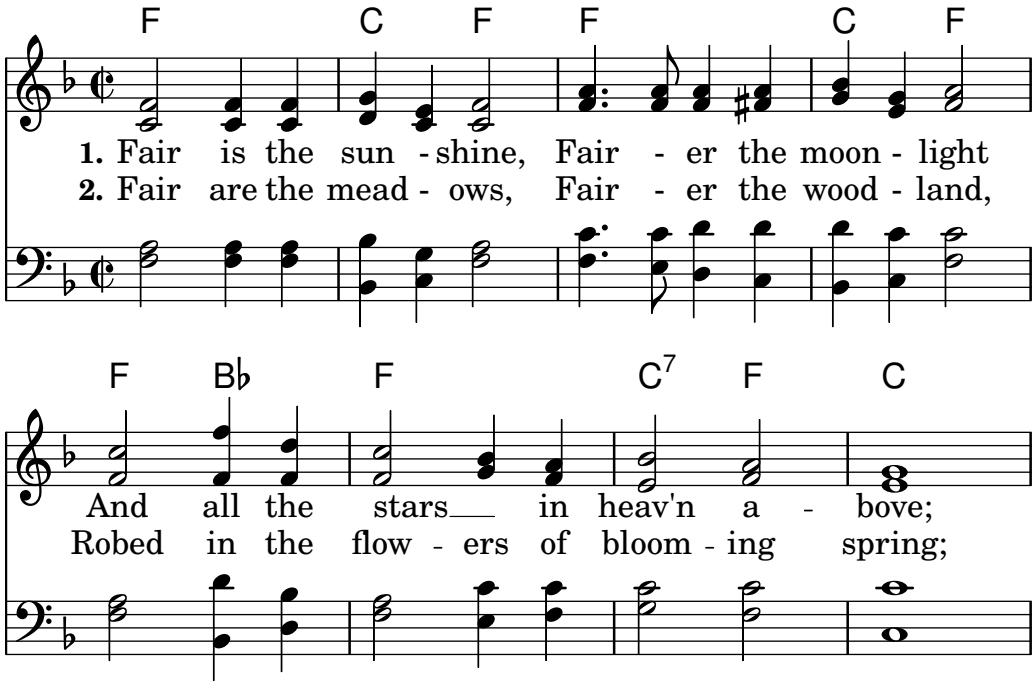
Voir aussi

Fichiers d'initialisation : `scm/define-woodwind-diagrams.scm`, `scm/display-woodwind-diagrams.scm`.

Morceaux choisis : Section "Vents" dans *Morceaux choisis*.

Référence des propriétés internes : Section "TextScript" dans *Référence des propriétés internes*, Section "instrument-specific-markup-interface" dans *Référence des propriétés internes*.

15 Notation des accords



Les accords se saisissent soit comme des notes normales, soit à l'aide d'un mode dédié ; ils seront rendus selon l'une des diverses nomenclatures occidentales. Les accords peuvent aussi se présenter sous forme nominale, ou bien en basse figurée.

15.1 Mode accords

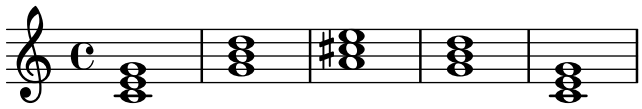
Le mode accords permet de saisir des accords en indiquant leur structure plutôt que les notes qui les composent.

15.1.1 Généralités sur le mode accords

Un accord peut se saisir en tant que musique simultanée, comme nous l'avons vu à la rubrique Section 5.1.1 [Notes en accords], page 221.

LilyPond dispose aussi d'un « mode accords » au sein duquel sera considérée la structure des accords, tels qu'ils se présentent dans les traditions occidentales, plutôt que les différentes hauteurs qui les composent. Ce mode est tout à fait adapté pour ceux qui sont plus habitués aux accords nommés. Pour plus d'information quant aux différentes façons de libeller votre code, voir Chapitre 19 [Modes de saisie], page 605.

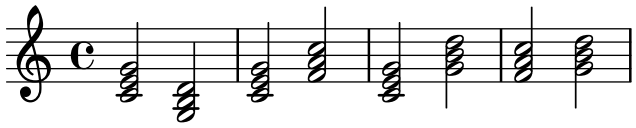
```
\chordmode { c1 g a g c }
```



Tout accord saisi dans ce mode dédié constitue un élément musical à part entière ; il pourra donc par exemple être transposé comme n'importe quel ensemble de hauteurs simultanées. `\chordmode` travaille en absolu ; une instruction `\relative` restera sans effet au sein d'un bloc `chordmode`. Notez toutefois que les hauteurs absolues sont une octave plus haut en `\chordmode` qu'en mode notes traditionnel.

Mode notes et mode accords peuvent tout à fait cohabiter dans une séquence musicale :

```
\relative {
  <c' e g>2 <g b d>
  \chordmode { c2 f }
  <c e g>2 <g' b d>
  \chordmode { f2 g }
}
```



Voir aussi

Glossaire musicologique : Section “Accord” dans *Glossaire*.

Manuel de notation : Chapitre 19 [Modes de saisie], page 605, Section 5.1.1 [Notes en accords], page 221.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

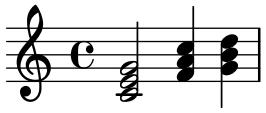
Problèmes connus et avertissements

Les raccourcis d’articulation ou d’ornementation ne sont pas disponibles en mode accord – voir Section 3.1.1 [Articulations et ornements], page 158.

15.1.2 Accords courants
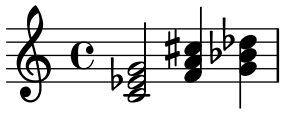
Dans le mode accords, introduit par la commande `\chordmode`, les accords ne sont indiqués que par leur note fondamentale, à laquelle on peut adjoindre une durée.

```
\chordmode { c2 f4 g }
```



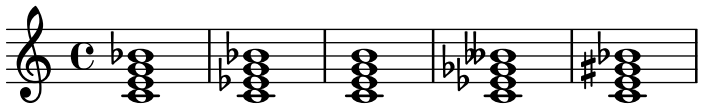
On peut cependant obtenir des accords différents – mineur, augmenté ou diminué – en utilisant le caractère deux points (`:`).

```
\chordmode { c2:m f4:aug g:dim }
```



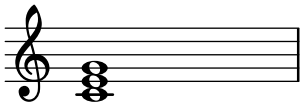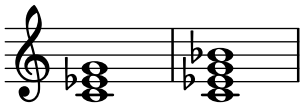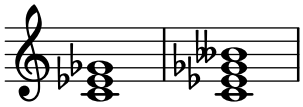
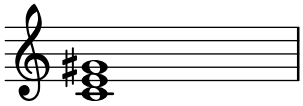
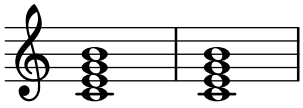
Les accords de septième sont aisément stipulables :

```
\chordmode { c1:7 c:m7 c:maj7 c:dim7 c:aug7 }
```



Voici les différents modificateurs d’accord de trois sons ou de septième. Par défaut, la septième ajoutée sera mineure ; la septième de dominante est donc l’accord de septième de base. Toutes les altérations sont relatives à la septième de dominante. Une table étendue des modificateurs et de leur utilisation est à l’annexe Section B.2 [Modificateurs d’accord usuels], page 918.

Modificateur	**Action**	**Exemple**

Néant	Action par défaut ; produit une triade majeure.	
m, m7	Accord mineur. Ce modificateur abaisse la tierce, et la septième s'il y en a une.	
dim, dim7	Accord diminué. Ce modificateur minorise la tierce, diminue la quinte et la septième s'il y en a.	
aug	Accord augmenté. Ce modificateur augmente la quinte.	
maj, maj7	Accord de septième majeure. Ce modificateur majorise la septième. Le 7 à la suite du `maj` est facultatif. Ce modificateur ne sert pas à créer une triade majeure.	

Voir aussi

Manuel de notation : Section 15.1.3 [Extension et altération d'accords], page 524, Section B.2 [Modificateurs d'accord usuels], page 918.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

Problèmes connus et avertissements

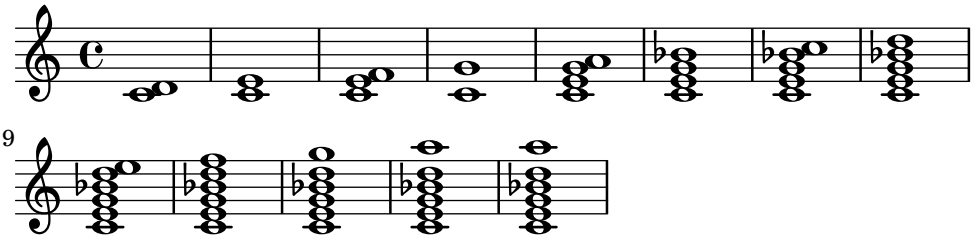
Un accord ne devrait comporter qu'un seul modificateur de qualité. La présence de plusieurs modificateurs ne déclenchera ni avertissement, ni erreur, mais le résultat pourrait être surprenant. Un accord qui n'est pas constructible à l'aide d'un unique modificateur devra faire l'objet d'une altération de ses composantes, comme indiqué à la rubrique Section 15.1.3 [Extension et altération d'accords], page 524.

15.1.3 Extension et altération d'accords

Le mode accords permet d'élaborer des accords complexes. Ainsi, on peut enrichir l'accord par des notes ajoutées, enlever certaines notes, augmenter ou diminuer certains intervalles, ajouter la note de basse ou créer un renversement.

Le premier nombre qui suit le caractère deux-points (`:`) permet de déterminer l'étendue d'un accord. L'accord sera construit par ajout à la fondamentale, d'autant de tierces consécutives que nécessaire pour atteindre le nombre spécifié. N'oubliez pas que la septième ajoutée à un accord est minorée par défaut. Lorsque le dernier degré ne correspond pas à une tierce – la sixte par exemple – les tierces seront empilées jusqu'à celle directement inférieure au degré mentionné, qui sera alors ajouté pour conclure l'accord. L'étendue maximale autorisée est la treizième ; toute étendue plus élevée sera interprétée comme un accord de treizième.

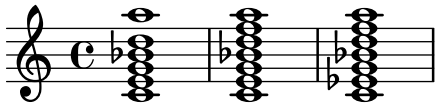
```
\chordmode {
  c1:2 c:3 c:4 c:5
  c1:6 c:7 c:8 c:9
  c1:10 c:11 c:12 c:13
  c1:14
}
```



De manière exceptionnelle, `c:5` produit un *power chord*, accord formé de la fondamentale et la quinte.
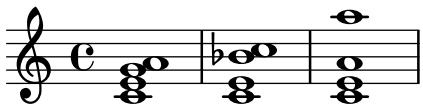
Dans la mesure où un accord de treizième majeure ne sonne pas très bien avec la onzième, la onzième est automatiquement enlevée, sauf à l'avoir explicitement spécifiée.

```
\chordmode {
  c1:13 c:13.11 c:m13
}
```



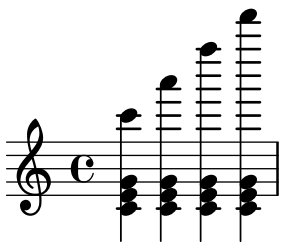
On peut enrichir l'accord par des notes ajoutées, que l'on indique après le chiffrage principal et que l'on sépare par des points, sans oublier que si l'on y ajoute une septième, celle-ci sera minorée et non majeure.

```
\chordmode {
  c1:3.5.6 c:3.7.8 c:3.6.13
}
```
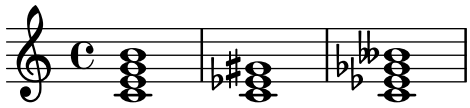


Les notes ajoutées peuvent monter aussi haut que de besoin.

```
\chordmode {
  c4:3.5.15 c:3.5.20 c:3.5.25 c:3.5.30
}
```



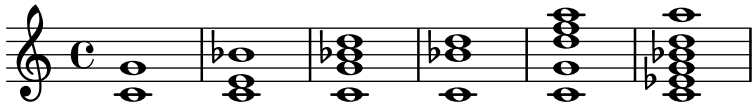
On peut augmenter ou diminuer certains intervalles au moyen des signes - ou + au degré considéré. L'altération de l'un des degrés automatiquement inclus dans la structure de base d'un accord s'effectue de la même manière.

```
\chordmode {
  c1:7+ c:5+.3- c:3-.5-.7-
}
```



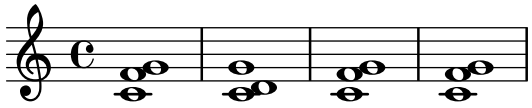
Après avoir ajouté des notes à un accord, on peut aussi en enlever certaines, en les spécifiant derrière un signe ^ – les séparer par un point lorsqu'il y en a plus d'une.

```
\chordmode {
  c1^3 c:7^5 c:9^3 c:9^3.5 c:13.11^3.7 c:m13^11
}
```



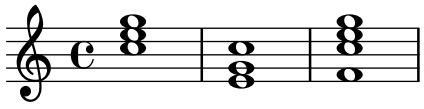
L'ajout du modificateur sus permet de créer un accord de suspension. Ceci aura pour effet de supprimer la tierce et d'y ajouter, suivant que vous spécifiez 2 ou 4, la seconde ou la quarte ; sus est alors équivalent à ^3. sus4 est équivalent à 5.4.

```
\chordmode {
  c1:sus c:sus2 c:sus4 c:5.4
}
```



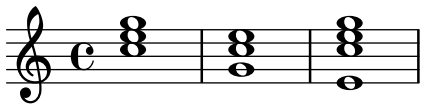
Les accords peuvent être combinés avec une note étrangère à la basse, au moyen de *accord/hauteur*.

```
\chordmode {
  c'1 c'/e c'/f
}
```



Si la note ajoutée appartient déjà à l'accord, la doublure supérieure sera supprimée, ce qui permet d'imprimer un renversement. Pour l'éviter, utilisez la syntaxe */+hauteur*.

```
\chordmode {
  c'1 c'/g c'/+e
}
```



L'automatisation des renversements et la vocification sont abordées dans Section 15.1.4 [Renversement d'accord et vocification spécifique], page 527.

Une table étendue des modificateurs et de leur utilisation est à l'annexe Section B.2 [Modificateurs d'accord usuels], page 918.

Voir aussi
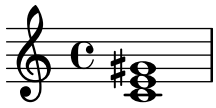
Manuel de notation : Section B.2 [Modificateurs d'accord usuels], page 918, Section 15.1.4 [Renversement d'accord et vocification spécifique], page 527.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

Problèmes connus et avertissements

Aucun nom de note ne peut être indiqué deux fois dans un accord. Ainsi, dans l'accord suivant, seule la quinte augmentée – notée 5+ – est prise en compte, parce qu'elle est indiquée en dernier :

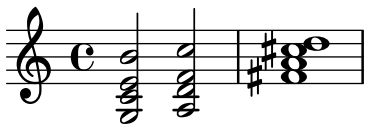
```
\chordmode { c1:3.5.5-.5+ }
```



15.1.4 Renversement d'accord et vocification spécifique

Les modifications d'accord ou l'ajout d'une basse sont complétées par diverses fonctions permettant d'imprimer automatiquement des renversements ou des vocifications particulières, à l'instar du *drop 2* couramment utilisé en jazz.

```
\chordmode {
  \dropNote 2 {
    c2:maj7 d:m7
  }
  \invertChords 1 d1:maj7
}
```



Contrairement à l'ajout de basses comme indiqué dans Section 15.1.3 [Extension et altération d'accords], page 524, ceci n'interfère que sur la façon dont les accords seront imprimés sur une portée, et aucunement sur leur représentation lettrée. Par ailleurs, l'utilisation de ces fonctions ne se limite pas seulement au mode accords ; elles sont aussi disponibles dans une construction d'accord avec <. . .> comme indiqué dans Section 5.1.1 [Notes en accords], page 221.

Voir aussi

Manuel de notation : Section 15.1.3 [Extension et altération d'accords], page 524, Section 5.1.1 [Notes en accords], page 221.

Morceaux choisis : Section "Accords" dans *Morceaux choisis*.

15.2 Gravure des accords

Les accords peuvent se présenter aussi bien sous forme nominative que comme un empilement de notes sur une portée.

15.2.1 Impression des noms d'accord

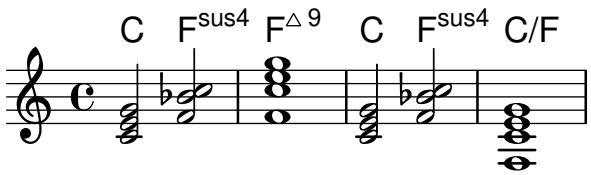
Les chiffrages d'accords sont liés au contexte `ChordNames` :

```
\new ChordNames {
  \chordmode {
    c2 f4. g8
  }
}
```

C F G

Les accords peuvent être saisis soit en tant que hauteurs simultanées, soit au moyen du mode accords. Le chiffrage affiché sera identique quel que soit le mode de saisie, à moins qu'il n'y ait inversion ou ajout de la basse.

```
chordmusic = \relative {
  <c' e g>2 <f bes c>
  <f c' e g>1
  \chordmode {
    c2 f:sus4 c1:/f
  }
}
<<
  \new ChordNames {
    \chordmusic
  }
  {
    \chordmusic
  }
>>
```

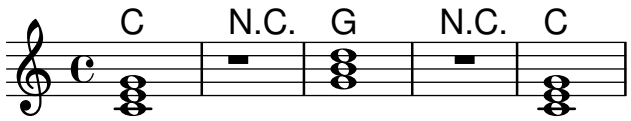


No Chord No Chord, symbole

L'apparition de silences, y compris multimesures, dans un contexte `ChordNames` déclenchera l'impression du texte « N.C. » (*No Chord* pour sans accord).

```
myChords = \chordmode {
  c1
  r1
  g1
  R1
  c1
}

<<
  \new ChordNames \myChords
  \new Staff \myChords
>>
```



`\chords { ... }` est un raccourci de `\new ChordNames \chordmode { ... }`.

```
\chords {
  c2 f4.:m g8:maj7
}
```




```
\new ChordNames {
  \chordmode {
    c2 f4.:m g8:maj7
  }
}
```



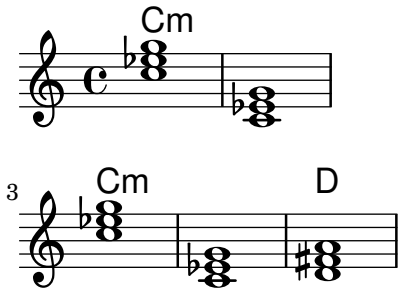
Morceaux choisis

Impression des accords si changement

Tout accord saisi est par défaut imprimé. Ce comportement est amendable : vous pouvez faire ressortir les chiffrages d'accords s'ils ne sont imprimés qu'aux changements d'accord ou en début de ligne.

```
harmonies = \chordmode {
  c'1:m c:m \break
  c'1:m c:m d
}

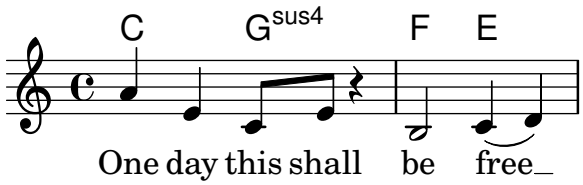
<<
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies
  }
  \new Staff {
    \harmonies
  }
>>
```



Chanson simple

Assembler des noms d'accords, une mélodie et des paroles permet d'obtenir la partition d'une chanson.

```
<<
  \chords { c2 g:sus4 f e }
  \new Staff \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```

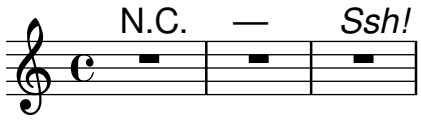


No Chord No Chord, personnalisation du symbole

Personnalisation du symbole no-chord

Par défaut, les silences apparaissant dans un contexte `ChordNames` déclenchent l'impression d'un texte « N.C. ». Ce *markup* peut être personnalisé grâce à la propriété de contexte `noChordSymbol`.

```
<<
  \chords {
    R1
    \set noChordSymbol = "---"
    R1
    \set noChordSymbol = \markup \italic "Ssh!"
    R1
  }
  {
    R1*3
  }
>>
```



Voir aussi

Glossaire musicologique : Section “Accord” dans *Glossaire*.

Manuel de notation : Section 5.2.6 [Saisie de musique en parallèle], page 244.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Bar_engraver” dans *Référence des propriétés internes*, Section “Chord_name_engraver” dans *Référence des propriétés internes*, Section “ChordNames” dans *Référence des propriétés internes*, Section “ChordName” dans *Référence des propriétés internes*, Section “Volta_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Un accord contenant un renversement ou dont la basse est changée ne sera pas chiffré correctement s'il est saisi en tant que musique simultanée.

15.2.2 Personnalisation des noms d'accord

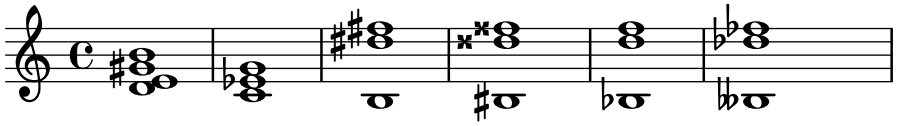
Il existe plus d'un système de chiffrage d'accords. Le nom des accords varie selon les traditions musicales, et plusieurs symboles représentent un même accord. LilyPond vous permet de créer votre propre nomenclature d'accords, tant au niveau des noms que des symboles qui les représenteront.

Le modèle par défaut des chiffrages d'accord est celui de Klaus Ignatzek pour le jazz (cf. Section "Références bibliographiques" dans *Essai*). Il est possible de créer son propre modèle de chiffrages à l'aide des différentes propriétés mentionnées ci-dessous. LilyPond dispose d'un système alternatif de chiffrage jazz qui a été élaboré grâce à ces même propriétés. Les deux notations, Ignatzek et alternative, sont illustrés dans le morceau choisi Section "Noms d'accord alternatifs" dans *Morceaux choisis*. Différentes notations courrantes de jazz sont prrésentées dans Section B.1 [Table des noms d'accord], page 918.

En plus des différents systèmes de nommage, le nom de la fondamentale varie selon la langue utilisée. Les instructions `\germanChords`, `\italianChords` et `\frenchChords` permettent de définir la langue.

Un système spécifique, `\semiGermanChords`, originellement dédié à l'enseignment musical dans les pays germanophones, a acquis une audience internationale. Il représente un si germanique par un B affublé d'un bémol ; tout le reste est identique à `\germanChords`.

default	E/D	Cm	B/B	B♯/B♯	B♭/B♭	B𝄫/B𝄫
German	E/d	Cm	H/h	H♯/his	B/b	H𝄫/heses
semi-German	E/d	Cm	H/h	H♯/his	B♭/b	H𝄫/heses
Italian	Mi/Re	Do m	Si/Si	Si♯/Si♯	Si♭/Si♭	Si𝄫/Si𝄫
French	Mi/Ré	Do m	Si/Si	Si♯/Si♯	Si♭/Si♭	Si𝄫/Si𝄫
English	E/D	Cm	B/B	B♯/B♯	B♭/B♭	B𝄫/B𝄫



Nombre de carnets de chant allemands indiquent un accord mineur par l'emploi de caractères en minuscule, sans le suffixe « m ». Cette fonctionnalité est gérée par la propriété `chordNameLowercaseMinor` :

```
\chords {
  \set chordNameLowercaseMinor = ##t
  c2 d:m e:m f
}
```

C d e F

La représentation d'un accord peut s'adapter à l'aide des propriétés suivantes :

`chordRootNamer`

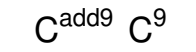
Le réglage par défaut pour le chiffrage d'accords jazz consiste à afficher la note fondamentale de chaque accord, exprimée par une lettre capitale, parfois suivie d'une altération, correspondant à la notation anglo-saxonne de la musique. Cette propriété a pour valeur la fonction qui transforme la hauteur de la note fondamentale en nom

de note ; c'est donc en assignant une nouvelle fonction à cette propriété que l'on peut produire des noms de note spéciaux adaptés par exemple aux systèmes de notation d'autres pays comme le « H » pour un accord de si en allemand.

`majorSevenSymbol`
: Cette propriété définit l'objet ajouté au `chordRootNamer` pour indiquer une septième majeure. Les options prédéfinies sont `whiteTriangleMarkup` (triangle blanc) – par défaut – et `blackTriangleMarkup` (triangle noir).

`additionalPitchPrefix`
: Lorsqu'un chiffrage contient des notes ajoutées, vous pouvez le préfixer d'une annotation. LilyPond n'en ajoute pas par défaut, dans le but de ne pas trop surcharger la partition ; vous pouvez cependant les faire apparaître si elles sont visuellement efficaces.

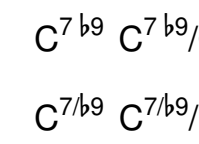
```
\new ChordNames {
  <c e g d'>    % add9
  \set additionalPitchPrefix = ""
  <c e g d'>    % add9
}
```



`chordNoteNamer`
: Lorsqu'un chiffrage mentionne une note ajoutée (par exemple la basse), les règles utilisées sont par défaut celles définies par la propriété `chordRootNamer` ci-dessus. Cependant, la propriété `chordNoteNamer` permet de régler cet élément indépendamment, par exemple pour imprimer la basse en caractères minuscules – c'est le cas de la fonction `chord-name:german-lowercase-name-markup` utilisée pour `\germanChords`.

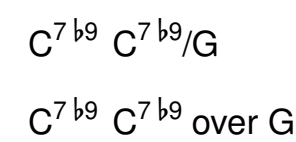
`chordNameSeparator`
: Les différents termes d'un chiffrage jazz (par exemple les notes de l'accord et la basse) sont habituellement légèrement espacés. La propriété `chordNameSeparator` permet d'indiquer un autre séparateur. Le séparateur entre un chiffrage et sa basse est géré par la propriété `slashChordSeparator`.

```
\chords {
  c4:7.9- c:7.9-/g
  \set chordNameSeparator = \markup { "/" }
  \break
  c4:7.9- c:7.9-/g
}
```



`slashChordSeparator`
: La note basse d'un accord n'est pas forcément la fondamentale. L'accord est alors « renversé » – certains diront « barré » parce que son chiffrage est habituellement flanqué d'une barre oblique entre l'accord de base et sa basse. La propriété `slashChordSeparator` permet de modifier ce séparateur – par défaut la barre de fraction.

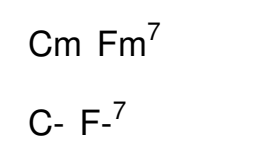
```
\chords {
  c4:7.9- c:7.9-/g
  \set slashChordSeparator = \markup { " over " }
  \break
  c4:7.9- c:7.9-/g
}
```



chordNameExceptions
: Cette propriété recense, sous forme de paire, les accords mis en forme de manière particulière. Le premier élément de chacune des paires répertorie les différentes hauteurs qui constituent l'accord. Le second élément est un *markup* qui sera ajouté au `chordRootNamer` lors de l'impression du chiffrage.

minorChordModifier
: Les accords mineurs sont habituellement identifiés par un « m » après leur fondamentale, ce qui est le comportement par défaut de LilyPond. Certaines nomenclatures ont cependant adopté un autre suffixe, comme le signe moins.

```
\chords {
  c4:min f:min7
  \set minorChordModifier = \markup { "-" }
  \break
  c4:min f:min7
}
```



chordPrefixSpacer
: Le modificateur pour accord mineur, géré par la propriété `minorChordModifier`, est par défaut accolé à la fondamentale. Vous pouvez cependant l'espacer de la fondamentale à l'aide de la propriété `chordPrefixSpacer` comme avec `\italianChords`. Notez bien que cet espacement sera réduit à néant si la fondamentale est altérée.

Commandes prédéfinies

`\whiteTriangleMarkup`, `\blackTriangleMarkup`, `\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`, `\englishChords`.

Morceaux choisis

Modèles de chiffrage d'accords

La propriété `chordNameExceptions` enregistre une liste de chiffrages personnalisés afin de traiter des cas non prévus ou incorrectement gérés.

Par défaut, LilyPond utilise le jeu de chiffrages issus de l'ouvrage de Klaus Ignatzek « Die Jazzmethode für Klavier 1 » ; l'algorithme de conversion des accords en chiffrages est inclus dans le fichier `scm/chord-ignatzek-names.scm`. Des règles additionnelles pour gérer des chiffrages exceptionnels sont enregistrées en tant que variables `ignatzekExceptions` dans le fichier `ly/chord-modifiers-init.ly`.

Le code ci-dessous modifie ces exceptions en trois étapes.

1. Saisie de musique en accords avec *markup* associé. Par convention, la base – autrement dit la note la plus basse – de chaque accord aura une hauteur de do (`c` pour les anglophones).
2. Appel de la fonction Scheme `sequential-music-to-chord-exceptions` pour créer une liste de nouvelles exceptions, puis l'ajouter à celles déjà répertoriées. Dans la mesure où `ignatzekExceptions` est défini avec le second paramètre de cette fonction fixé à `#t`, afin d'ignorer la base de chaque accord, il faut faire de même.
3. Enregistrement de la nouvelle liste d'exceptions.

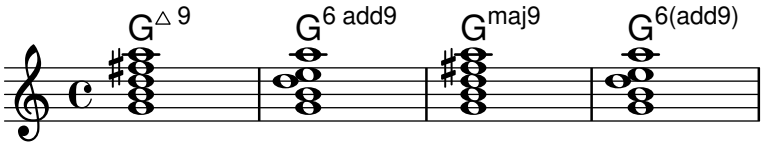
```
% Step 1: Define music with chords and markup for maj9 and 6(add9).
chExceptionMusic = {
  <c e g b d'>-\markup { \super "maj9" }
  <c e g a d'>-\markup { \super "6(add9)" }
}

% Step 2: Create extended exception list.
chExceptions =
#(append (sequential-music-to-chord-exceptions chExceptionMusic #t)
         ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  % Step 3: Register extended exception list.
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

<<
  \new ChordNames \theMusic
  \new Voice \theMusic
>>

\layout {
  line-width = 10\cm
  ragged-right = ##f
}
```



Chiffrage d'un maj7

La représentation d'un accord de septième majeure se gère par la propriété de contexte `majorSevenSymbol`.

```
\chords {
  c:7+
  \set majorSevenSymbol = \markup { j7 }
  c:7+
}
```

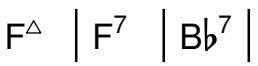


Chiffrages et barres de mesure

L'ajout du graveur `Bar_engraver` à un contexte `ChordNames` permet d'imprimer les barres de mesure entre les chiffrages.

```
\new ChordNames \with {
  \override BarLine.bar-extent = #'(-1 . 3)
  \consists "Bar_engraver"
}

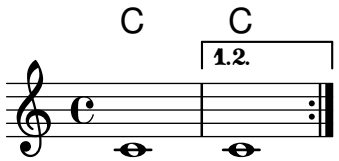
\chordmode {
  f1:maj7 f:7 bes:7
}
```



Crochet de reprise sous les chiffrages d'accord

L'ajout du `Volta_engraver` à la bonne portée permet d'imprimer les crochets de reprise entre les chiffrages et la portée.

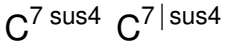
```
\score {
  <<
    \chords { c1 c1 }
    \new Staff \with { \consists "Volta_engraver" }
    {
      \repeat volta 2 { c'1 \alternative { c' } }
    }
  >>
  \layout {
    \context {
      \Score
      \remove "Volta_engraver"
    }
  }
}
```



Personnalisation du séparateur d'accords

Le séparateur de termes d'un chiffrage peut adopter n'importe quelle forme à l'aide d'un *markup*.

```
\chords {
  c:7sus4
  \set chordNameSeparator = \markup { \typewriter | }
  c:7sus4
}
```



Voir aussi

Manuel de notation : Section B.2 [Modificateurs d'accord usuels], page 918, Section B.1 [Table des noms d'accord], page 918.

Essai sur la gravure musicale automatisée : Section “Références bibliographiques” dans *Essai*.

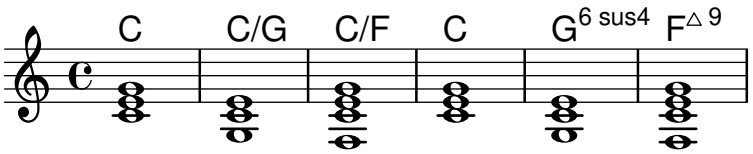
Fichiers d’initialisation : `scm/chords-ignatzek-names.scm`, `scm/chord-entry.scm`, `ly/chord-modifiers-init.ly`.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

Problèmes connus et avertissements

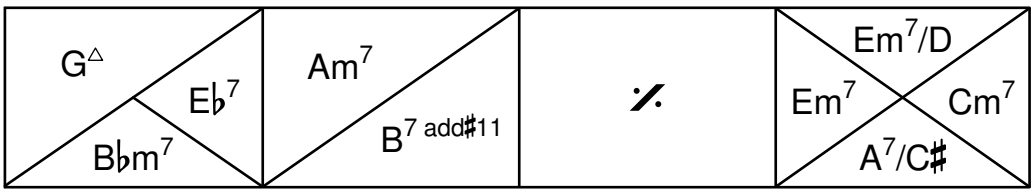
Les chiffrages d’accords ne sont déterminés que par la succession des hauteurs de notes. En d’autres termes, les accords inversés ne sont pas reconnus, ni les notes ajoutées à la basse. C’est pourquoi les accords saisis au moyen de la syntaxe `<...>` peuvent produire des chiffrages étranges.

```
myChords = \relative c' {
  \chordmode { c1 c/g c/f }
  <c e g>1 <g c e> <f c' e g>
}
<<
  \new ChordNames { \myChords }
  \new Staff { \myChords }
>>
```



15.2.3 Grilles harmoniques

Dans certaines contrées européennes, notamment en France, les musiciens de jazz utilisent ce qu’ils appellent des « grilles harmoniques », dans lesquelles les accords sont imprimés dans des cases.



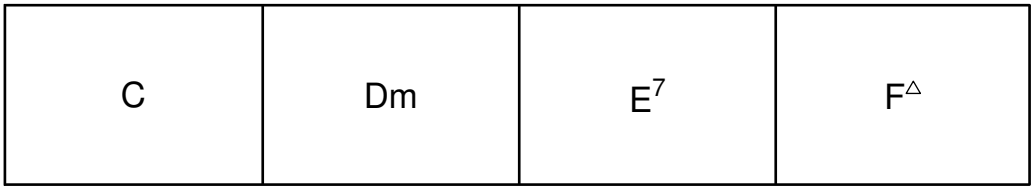
Bien qu’ils soient omis dans le reste de cette section pour des raisons de concision, les réglages suivants du bloc `\paper` sont recommandés pour l’impression de grilles harmoniques :

```
\paper {
  indent = 0
  ragged-right = ##f
}
```

`indent = 0` donne l’assurance que la première ligne ne sera pas indentée – voir Section 26.5.3 [Variables d’indentation et de décalage], page 699. La clause `ragged-right = ##f` permet d’étaler en pleine largeur les grilles sur une ligne – voir Section 26.5.1 [Variables de marge et de largeur], page 697.

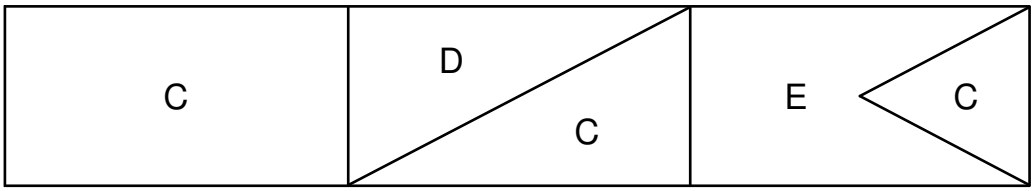
La création d’une grille harmonique demande l’instanciation d’un contexte `ChordGrid`.

```
\new ChordGrid \chordmode { c1 d1:m e1:7 f1:7+ }
```

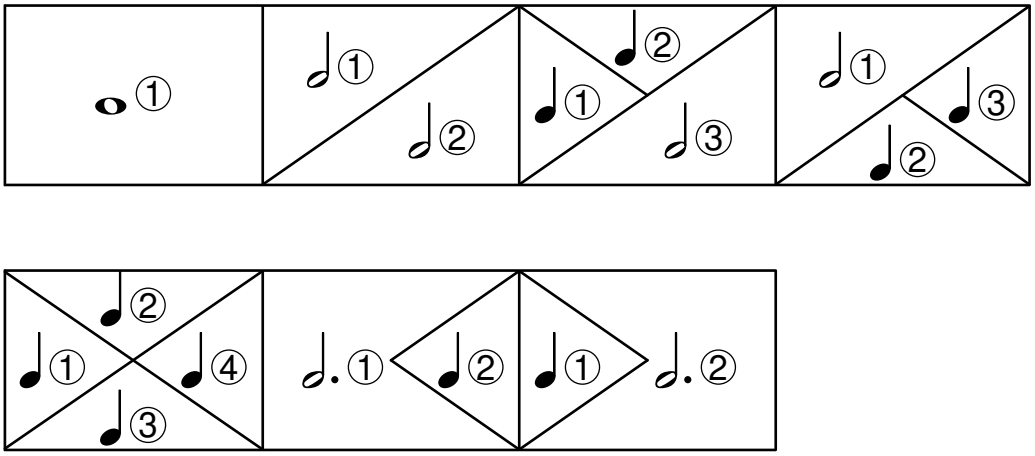


Chaque case est automatiquement subdivisée.

```
\new ChordGrid \chordmode {
  c1
  d2 c2
  e2. c4
}
```



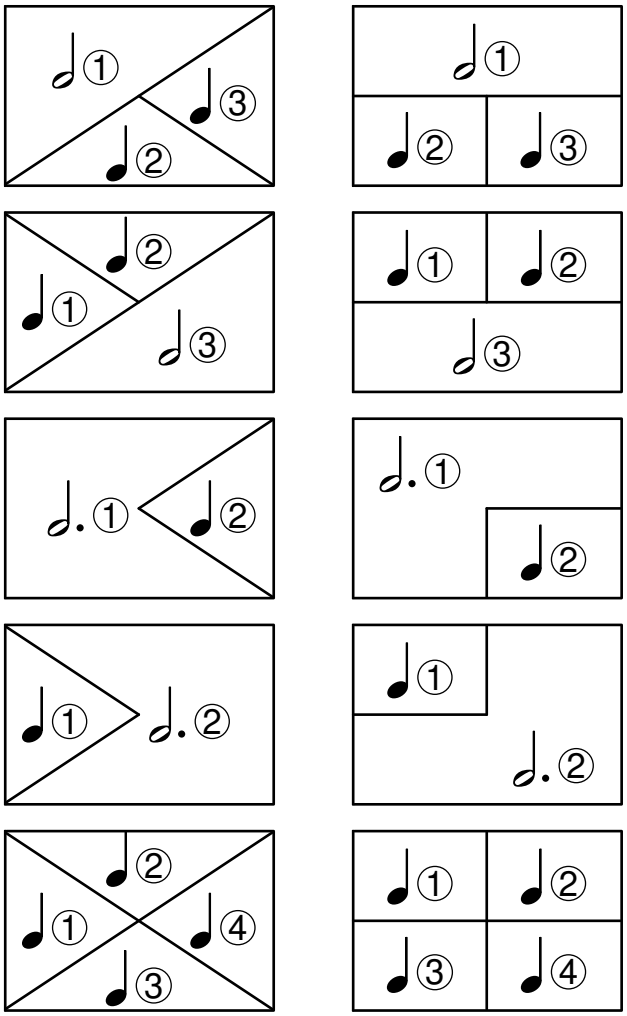
Les accords qui s'étendent sur la mesure sont centrés dans leur case. Ceux durant la moitié de la mesure occupent la moitié de la case, et ceux n'en prenant que le quart occupent le quart de la case. Voici un résumé des règles par défaut en matière de subdivision des cases :



L'instruction `\medianChordGridStyle` modifie la présentation par défaut des cases marquant les divisions de la mesure pour adopter le style préconisé par Philippe Baudoin dans son ouvrage *Jazz, mode d'emploi*.

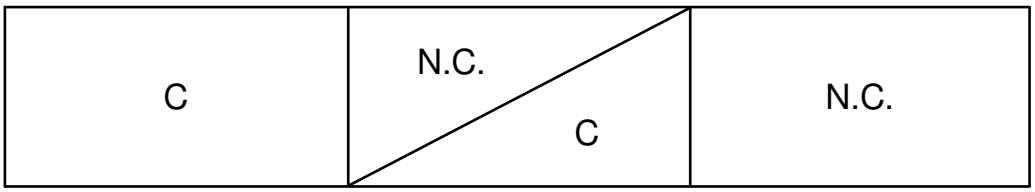
```
\layout {
  \context {
    \ChordGrid
    \medianChordGridStyle
  }
}
```

Default style `\medianChordGridStyle`



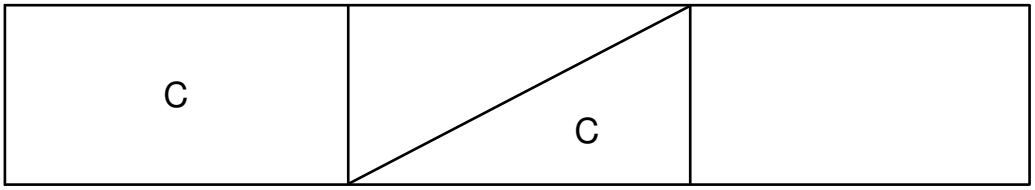
Dans une grille harmonique, les silences déclenchent l'impression du symbole `noChordSymbol` comme dans un contexte `ChordNames` – voir Section 15.2.1 [Impression des noms d'accord], page 527.

```
\new ChordGrid \chordmode { c1 r2 c2 R1 }
```



Les sauts créent de l'espace blanc, qui peut occuper seule une partie de la case.

```
\new ChordGrid \chordmode { c1 s2 c2 s1 }
```



Morceaux choisis

Personnalisation du style de grille harmonique

Il est possible de personnaliser la division des cases à l'aide des propiétés de `ChordSquare` `measure-division-lines-alist` et `measure-division-chord-placement-alist`, qui sont toutes deux des listes associatives. Leurs clés sont des divisions de la mesure, autrement dit des listes de fractions de la mesure que chaque accord, silence ou saut représente. Plus précisément, cette liste de divisions de la mesure est constituée de nombres positifs exacts dont l'addition fait 1, comme par exemple `'(1/2 1/4 1/4)`. L'exigence d'exactitude signifie par exemple que `1/2` est valide, contrairement à `0.5`.

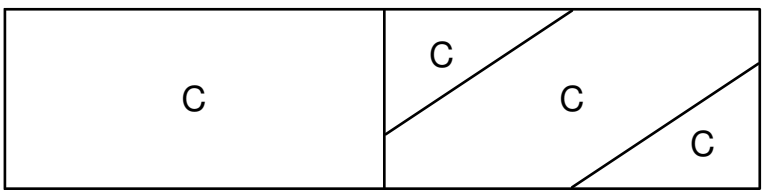
Les valeurs de `measure-division-lines-alist` sont des listes de lignes, représentes sous la forme `(`*`x1 y1 x2 y2`*`)`. La ligne débute au point `(`*`x1 . y1`*`)` et se temine en `(`*`x2 . y2`*`)`. Les coordonnées sont comprises dans l'intervalle $[-1, 1]$ relativement à l'étendue de la case.

Les valeurs de `measure-division-chord-placement-alist` sont des listes de paires `(`*`x . y`*`)` indiquant le positionnement des accords respectifs.

L'exemple ci-dessous illustre le cas particulier d'une grille dont le style spécifie le découpage des mesures en trois parts égales.

```
\paper {
  line-width = 10\cm
  ragged-right = ##f
}

\new ChordGrid \with {
  \override ChordSquare.measure-division-lines-alist =
    #'(((1) . ())
       ((1/3 1/3 1/3) . ((-1 -0.4 0 1) (0 -1 1 0.4))))
  \override ChordSquare.measure-division-chord-placement-alist =
    #'(((1) . ((0 . 0)))
       ((1/3 1/3 1/3) . ((-0.7 . 0.5) (0 . 0) (0.7 . -0.5))))
}
\chordmode {
  \time 3/4
  c2.
  c4 c4 c4
}
```

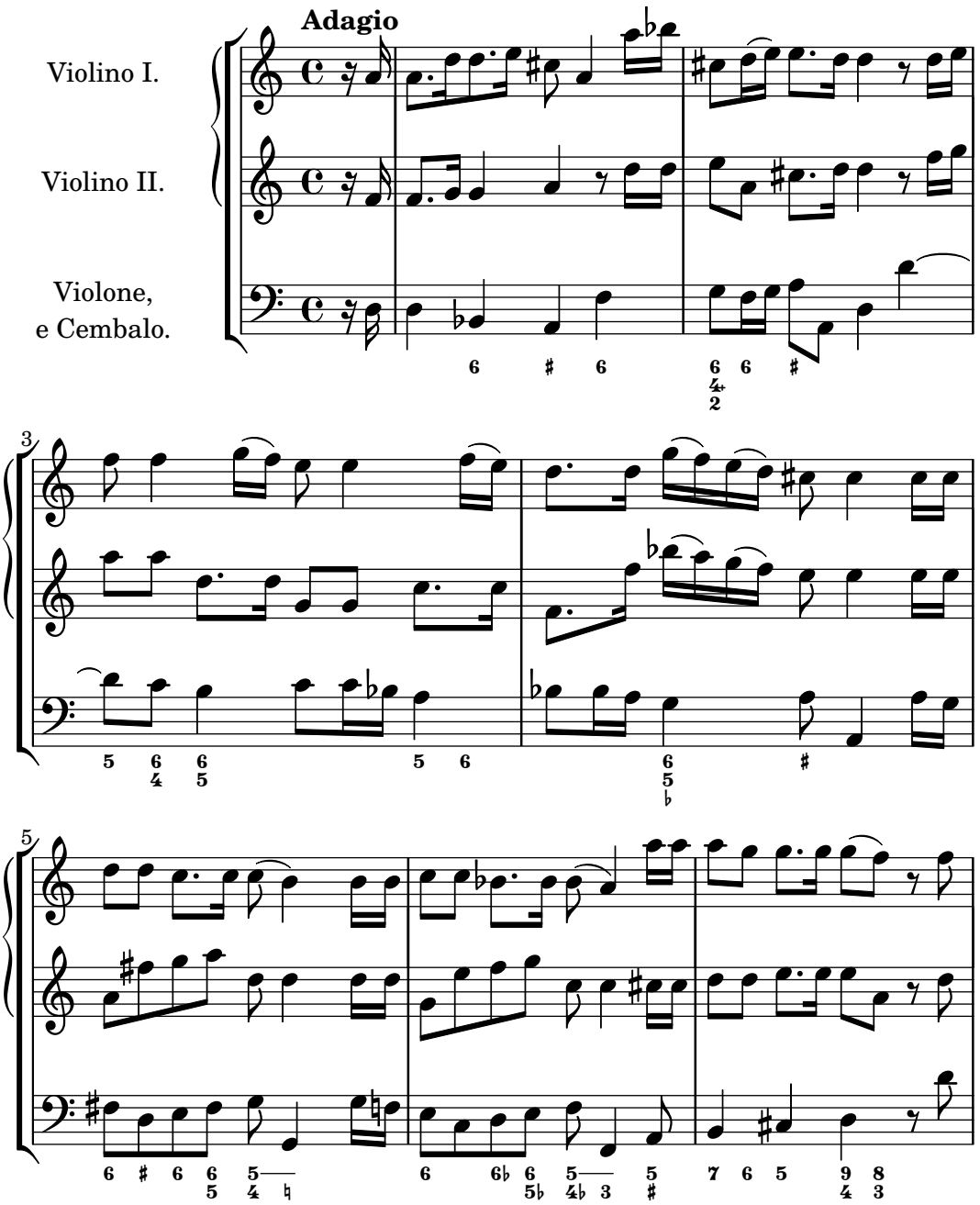


Voir aussi

Glossaire musicologique : Section “chord grid” dans *Glossaire*.

Référence des propriétés internes : Section “ChordGrid” dans *Référence des propriétés internes*, Section “ChordGridScore” dans *Référence des propriétés internes*, Section “ChordSquare” dans *Référence des propriétés internes*, Section “Chord_square_engraver” dans *Référence des propriétés internes*, Section “GridChordName” dans *Référence des propriétés internes*, Section “Grid_chord_name_engraver” dans *Référence des propriétés internes*.

15.3 Basse chiffrée



LilyPond permet de générer des parties de continuo.

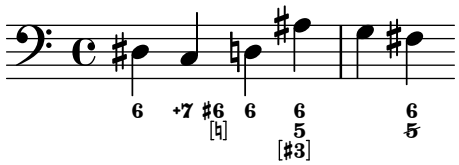
15.3.1 Introduction à la basse chiffrée

Les parties de basse continue étaient très répandues dans la musique baroque et jusqu’à la fin du XVIII[e] siècle. Comme son nom l’indique, le *continuo* constitue une partie à lui seul, qui se déroule tout au long de l’œuvre pour en donner la structure harmonique.

Les musiciens du *continuo* jouent des claviers (clavecin, orgue) ou de tout autre instrument pouvant réaliser des accords. Leur partie est constituée d'une portée de basse aux notes agrémentées de combinaisons de chiffres et signes indiquant le développement des accords à jouer, ainsi que leur éventuel renversement. Cette notation était avant tout un guide, invitant le musicien à improviser de lui-même l'accompagnement.

LilyPond gère la basse chiffrée, appelée aussi *continuo* ou *basse continue*.

```
<<
  \new Voice { \clef bass dis4 c d ais g fis}
  \new FiguredBass {
    \figuremode {
      <6>4 <7\+>8 <6+ [_!]> <6>4 <6 5 [3+]> |
      <_>4 <6 5/>4
    }
  }
>>
```



La gestion de la basse chiffrée se décompose en deux parties. Dans un premier temps, le mode `\figuremode` permet de saisir les accords sous forme chiffrée. Le contexte `FiguredBass` s'occupera ensuite de gérer les objets `BassFigure`. La basse chiffrée pourra être attachée à un contexte `Staff`.

L'expression `\figures { ... }` constitue une raccourci à `\new FiguredBass \figuremode { ... }`.

Bien que la gestion de la basse chiffrée ressemble beaucoup à celle des accords, elle est beaucoup plus simpliste. Le mode `\figuremode` ne fait que stocker des chiffres que le contexte `FiguredBass` se chargera d'imprimer tels quels. En aucune manière ils ne sont transformés en son, et ils ne sont pas rendus dans un fichier MIDI.

Voir aussi

Glossaire musicologique : Section “basse chifrée” dans *Glossaire*.

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

15.3.2 Saisie de la basse chiffrée

`\figuremode` permet de faire la relation entre ce qui est saisi et le mode de chiffrage. De plus amples informations quant aux différents modes sont regroupées à la rubrique Chapitre 19 [Modes de saisie], page 605.

En mode de saisie, un chiffrage est délimité par ‘<’ et ‘>’. La durée est indiquée après le ‘>’ :

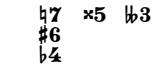
```
\new FiguredBass {
  \figuremode {
    <6 4>2
  }
}
```



Une altération – y compris un bécarre – permet de modifier l'un des degrés, en adjoignant un ‘+’ (dièse), un ‘-’ (bémol) ou un ‘!’ (bécarre) au chiffre considéré. Une altération double

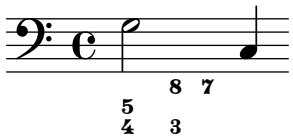
s'obtient en doublant le modificateur. Le chiffre est souvent omis lorsque la tierce est modifiée, ce qui s'obtient en utilisant un '_' en lieu et place du chiffre.

```
\figures {
  <7! 6+ 4-> <5++> <3--> <_+> <7 _!>
}
```



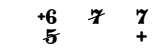
En l'absence d'altération, un '_' crée un chiffrage vide qui néanmoins occupe de l'espace. Ceci peut permettre de contrôler l'empilement des chiffres.

```
<<
  {
    \clef bass
    g2 c4
  }
  \figures {
    <_ 5 4>4 <8 _ 3>8 <7>
  }
>>
```



Vous pouvez stipuler un intervalle augmenté ou diminué :

```
\figures {
  <6\+ 5/> <7/> <7 _\+>
}
```



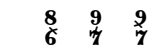
Vous pouvez barrer un chiffre d'une oblique inversée :

```
\figures {
  <5> <5\\>
}
```



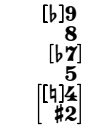
Certains chiffrages disposent d'un glyphe spécifique :

```
\figures {
  <8 6\\> <9 7\\> <9\\ 7>
}
```



Vous pouvez insérer des crochets aux altérations, chiffrages et groupes de chiffrages consécutifs :

```
\figures {
  <9[-] 8 [7-] 5 [4[!] 2+]>
}
```



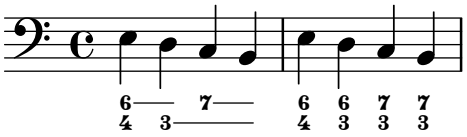
Vous pouvez aussi ajouter des chaînes de caractères ou des étiquettes – cf. Section A.1 [Commandes pour *markup*], page 837.

```
\figures {
  <\markup { \fontsize #-5 \number 6 \tiny \super (1) } 5>
}
```



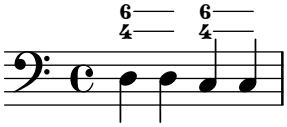
Lorsque des chiffrages se répètent, vous pouvez utiliser des lignes de prolongation.

```
<<
  {
    \clef bass
    e4 d c b,
    e4 d c b,
  }
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 3> <7 3> <7 3>
    \bassFigureExtendersOff
    <6 4>4 <6 3> <7 3> <7 3>
  }
>>
```

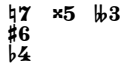
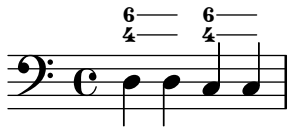


En pareil cas, la ligne de prolongation masquera toujours le chiffre qu'elle rappelle dans le chiffrage suivant à moins d'avoir été explicitement interrompue par un `\!`.

```
<<
  \figures {
    \bassFigureExtendersOn
    <6 4>4 <6 4> <6\! 4\!> <6 4>
  }
  {
    \clef bass
    d4 d c c
  }
>>
```



Voici, de manière synthétique, les différents modificateurs disponibles :

Modificateur	Utilisation	Exemple
+, -, !	altérations	
\+, /	augmentation ou diminution d'un degré	
\\	augmentation d'un demi-ton	
\!	terminaison d'une prolongation	

Commandes prédéfinies

`\bassFigureExtendersOn`, `\bassFigureExtendersOff`.

Morceaux choisis

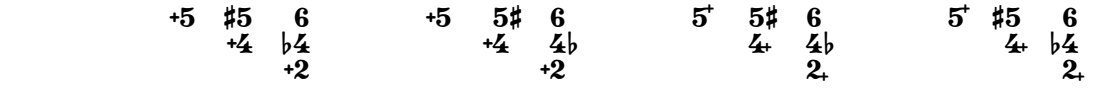
Emplacement des altération en basse continue

On peut choisir d'imprimer les altérations et signes plus aussi bien avant qu'après les chiffres, en réglant les propriétés `figuredBassAlterationDirection` et `figuredBassPlusDirection`.

Lorsqu'un chiffre est suivi d'un signe plus, certains chiffrages utiliseront des glyphes spécifiques.

```
#(set-global-staff-size 26)

\figures {
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassAlterationDirection = #RIGHT
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassPlusDirection = #RIGHT
  <5\+> <5+ 4\+> <6 4- 2\+> r
  \set figuredBassAlterationDirection = #LEFT
  <5\+> <5+ 4\+> <6 4- 2\+> r
}
```



Ajustement des glyphes d'altération en basse chifrée

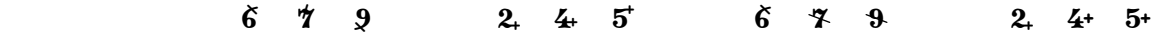
En basse chiffrée, des glyphes spécifiques sont utilisés par défaut pour les `6\\`, `7\\` et `9\\`. D'autres glyphes ont été conçus pour les `2\+`, `4\+` et `5\+` ; ils sont utilisés par défaut dès lors qu'un signe plus est ajouté au chiffre.

Pour modifier ce comportement, il faut passer une liste associative à `figuredBassPlusStrokedAlist` et basculer le glyphe en question sur `#f` (ou ne pas le mentionner).

```
#(set-global-staff-size 26)

\figures {
  \set figuredBassPlusDirection = #RIGHT
  <6\\> <7\\> <9\\> r
  <2\+> <4\+> <5\+> r

  \set figuredBassPlusStrokedAlist =
    #'((2 . "figbass.twoplus")
       ;; (4 . "figbass.fourplus")
       ;; (5 . "figbass.fiveplus")
       (6 . "figbass.sixstroked")
       ;; (7 . "figbass.sevenstroked")
       ;; (9 . "figbass.ninestroked")
      )
  <6\\> <7\\> <9\\> r
  <2\+> <4\+> <5\+> r
}
```



Voir aussi

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

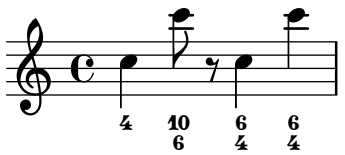
Référence des propriétés internes : Section “BassFigure” dans *Référence des propriétés internes*, Section “BassFigureAlignment” dans *Référence des propriétés internes*, Section “BassFigureBracket” dans *Référence des propriétés internes*, Section “BassFigureContinuation” dans *Référence des propriétés internes*, Section “BassFigureLine” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*.

15.3.3 Gravure de la basse chiffrée

Une ligne de basse chiffrée s’imprime soit dans un contexte `FiguredBass`, soit dans la plupart des autres contextes du niveau de la portée.

Le contexte `FiguredBass` ne tient aucun compte des notes qui apparaissent sur la portée.
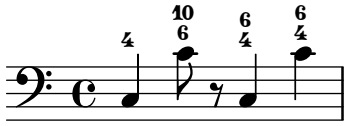
```
<<
  \relative {
    c''4 c'8 r8 c,4 c'
  }
  \new FiguredBass {
    \figuremode {
      <4>4 <10 6>8 s8
      <6 4>4 <6 4>
    }
  }
>>
```



Il est impératif, dans cet exemple, d’instancier explicitement le contexte `FiguredBass` pour éviter l’apparition d’une portée supplémentaire vide.

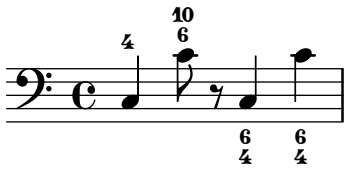
On peut ajouter une basse chiffrée directement à un contexte `Staff`. L'alignement vertical est alors automatiquement ajusté.

```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff" {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



La basse chiffrée attachée à un contexte `Staff` peut se positionner au-dessus ou en dessous de la portée.

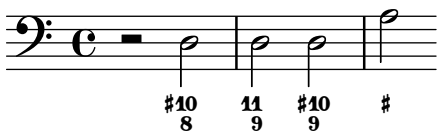
```
<<
  \new Staff = "myStaff"
  \figuremode {
    <4>4 <10 6>8 s8
    \bassFigureStaffAlignmentDown
    <6 4>4 <6 4>
  }
  %% Put notes on same Staff as figures
  \context Staff = "myStaff" {
    \clef bass
    c4 c'8 r8 c4 c'
  }
>>
```



Dès lors que l'empilement d'une basse chiffrée comporte des nombres de plus d'un caractère, il est possible de gérer leur alignement horizontal à l'aide de la propriété de contexte `figuredBassLargeNumberAlignment`.

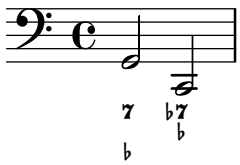
```
<<
  \new Voice {
    \clef bass
    r2 d | d d | a2
  }
  \new FiguredBass \figuremode {
    s2 <10+ 8> |
    \set figuredBassLargeNumberAlignment = #RIGHT
```

```
    <11 9>2
    \set figuredBassLargeNumberAlignment = #LEFT
    <10+ 9>2 |
    <_+>2
  }
>>
```



L'espacement vertical des éléments d'une basse chiffrée se gère à l'aide des sous-propriétés `minimum-distance` et `padding` de `staff-staff-spacing`.

```
<<
  { \clef bass g,2 c, }
  \figures {
    \once \override BassFigureLine
                    .staff-staff-spacing.minimum-distance = 3
    <7 _-> <7- _->
  }
>>
```



Commandes prédéfinies

`\bassFigureStaffAlignmentDown`, `\bassFigureStaffAlignmentUp`, `\bassFigureStaffAlignmentNeutral`.

Voir aussi

Morceaux choisis : Section “Accords” dans *Morceaux choisis*.

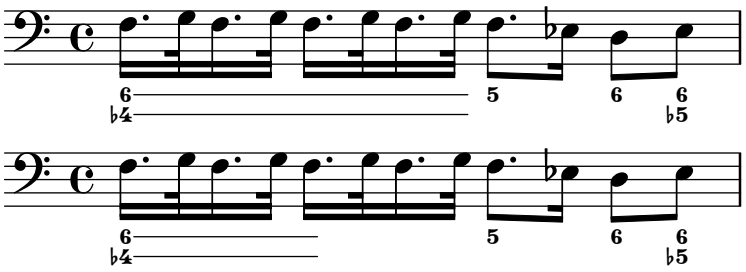
Référence des propriétés internes : Section “BassFigure” dans *Référence des propriétés internes*, Section “BassFigureAlignment” dans *Référence des propriétés internes*, Section “BassFigureBracket” dans *Référence des propriétés internes*, Section “BassFigureContinuation” dans *Référence des propriétés internes*, Section “BassFigureLine” dans *Référence des propriétés internes*, Section “FiguredBass” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les lignes de prolongation seront correctes dès lors que notes et chiffrages adoptent des durées identiques.

```
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are correct here,
    % with the same rhythm as the bass.
```

```
    \repeat unfold 4 { <6 4->16. <6 4->32 }
    <5>8. r16 <6>8 <6\! 5->
  }
>>
<<
  {
    \clef bass
    \repeat unfold 4 { f16. g32 } f8. es16 d8 es
  }
  \figures {
    \bassFigureExtendersOn
    % The extenders are incorrect here,
    % even though the timing is the same.
    <6 4->4 <6 4->4
    <5>8. r16 <6>8 <6\! 5->
  }
>>
```



16 Musique contemporaine

L'aube du XXe siècle a vu bourgeonner nombre de techniques et de styles de composition. Qu'il s'agisse des nouveaux développements autour de l'harmonie et du rythme, de l'expansion du spectre des hauteurs et de l'évolution de nombreuses techniques instrumentales, tous ces différents phénomènes ont participé à l'évolution de la notation musicale. Les paragraphes qui suivent sont là pour vous proposer des références et informations quant à ces nouvelles techniques de notation.

16.1 Hauteur et harmonie en musique contemporaine

Intéressons-nous tout d'abord à ce qui relève de la notation des hauteurs et à l'harmonie en musique contemporaine.

16.1.1 Généralités en matière de hauteur et d'harmonie

- La notation habituelle des quarts de tons est abordée à la rubrique Section 1.1.5 [Nom des notes dans d'autres langues], page 11.
- Les tonalités inhabituelles sont abordées à la rubrique Section 1.3.2 [Armure], page 25.
- Les pratiques contemporaines en matière d'altération sont abordées à la rubrique Section 1.3.5 [Altérations accidentelles automatiques], page 32.

16.1.2 Notation microtonale

16.1.3 Armures contemporaines et harmonie

16.2 Approches du rythme en musique contemporaine

Abordons à présent certaines particularités de la notation du rythme en musique contemporaine.

16.2.1 Généralités sur le rythme en musique contemporaine

- Les métrique complexes sont abordées à la rubrique Section 2.3.2 [Métrique], page 81.
- Les bases de la polymétrie sont abordées à la rubrique Section 2.3.6 [Notation polymétrique], page 96.
- Certaines particularités en matière de ligature sont abordées à la rubrique Section 2.4.4 [Liens de croches en soufflet], page 121.
- Les lignes de mensuration (barres de mesures uniquement entre les portées) sont abordées à la rubrique Section 6.1.2 [Regroupement de portées], page 250.

16.2.2 N-olets et musique contemporaine

16.2.3 Métriques contemporaines

16.2.4 Notation polymétrique étendue

16.2.5 Ligatures et musique contemporaine

16.2.6 Barres de mesure et musique contemporaine
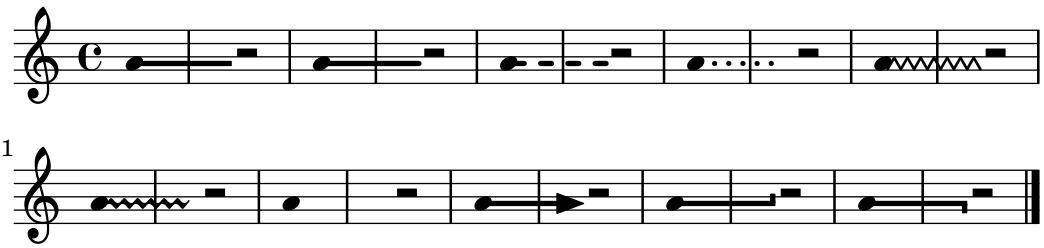
16.3 Notation graphique

Les éléments rythmiques peuvent se prolonger par une ligne de durée représentée par un objet graphique `DurationLine`. Cette ligne peut adopter différents styles : `'beam`, `'line`, `'dashed-line`,

'dotted-line, 'zigzag, 'trill ou 'none. La ligne peut se terminer par un crochet (uniquement pour le style 'beam) ou une flèche.

```
\layout {
  \context {
    \Voice
    \consists Duration_line_engraver
    \omit Stem
    \omit Flag
    \omit Beam
    \override NoteHead.duration-log = 2
  }
}

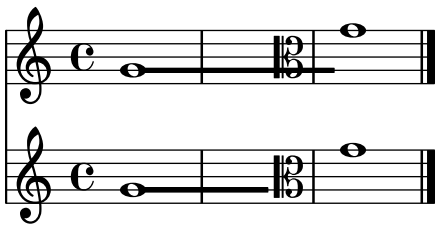
{
  a'1\- s2 r
  \once \override DurationLine.style = #'line
  a'1\- s2 r
  \once \override DurationLine.style = #'dashed-line
  \once \override DurationLine.dash-period = 2
  a'1\- s2 r
  \once \override DurationLine.style = #'dotted-line
  \once \override DurationLine.dash-period = 1
  \once \override DurationLine.bound-details.right.padding = 1
  a'1\- s2 r
  \once \override DurationLine.thickness = 2
  \once \override DurationLine.style = #'zigzag
  a'1\- s2 r
  \once \override DurationLine.style = #'trill
  a'1\- s2 r
  \once \override DurationLine.style = #'none
  a'1\- s2 r
  \once \override DurationLine.bound-details.right.end-style = #'arrow
  a'1\- s2 r
  \override DurationLine.bound-details.right.end-style = #'hook
  a'1\- s2 r
  \override DurationLine.details.hook-direction = #DOWN
  a'1\- s2 r
  \bar "|."
}
```



DurationLine peut éviter les éléments de BreakAlignGroup en cours de ligne.

```
\layout {
  \context {
    \Voice
    \consists "Duration_line_engraver"
  }
}

<<
  \new Staff {
    g'1\- s \clef "alto" g'
  }
  \new Staff {
    \override DurationLine.bound-details
                          .right.end-on-break-align-group = ##t
    g'1\- s \clef "alto" g' \bar "|."
  }
>>
```



16.4 Techniques de partition contemporaine

16.5 Nouvelles techniques instrumentales

16.6 Informations complémentaires et exemples pertinents

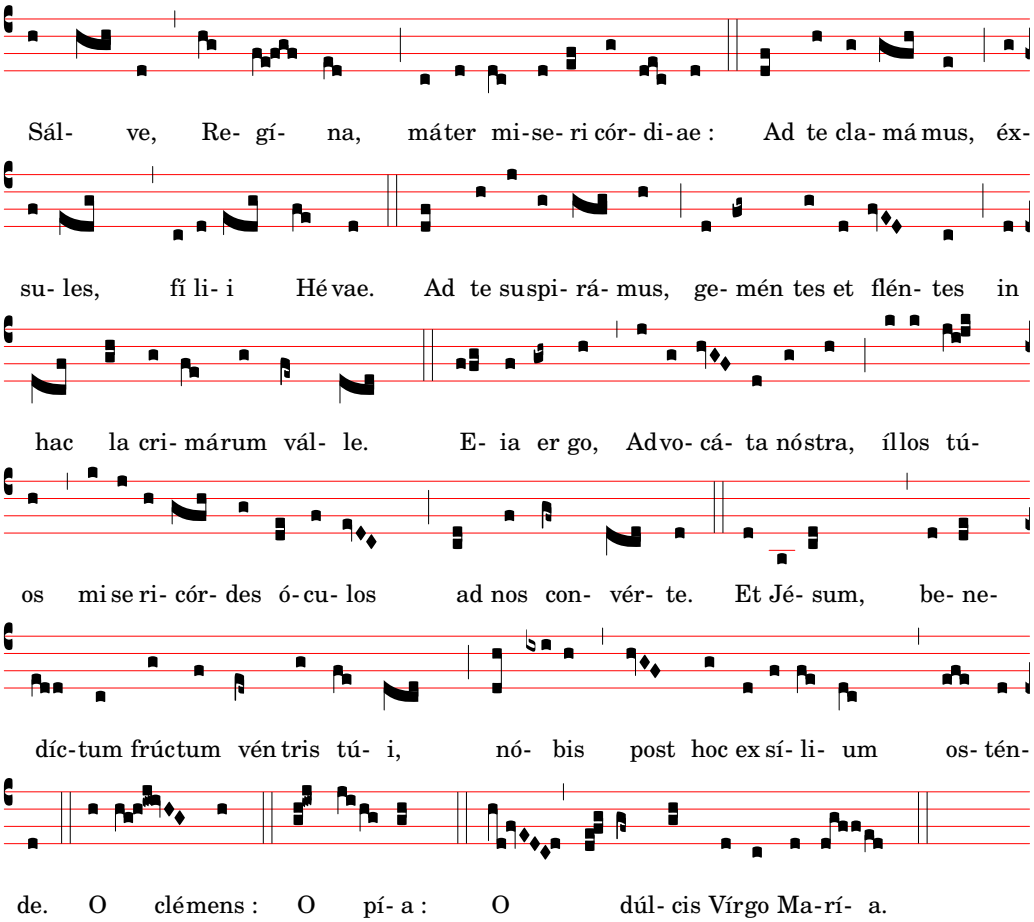
Vous trouverez ici une sélection d'ouvrages de référence, d'exemples et autres ressources qui vous permettront d'étudier plus avant la notation contemporaine.

16.6.1 Ouvrages et articles sur la notation en musique contemporaine

- *Music Notation in the Twentieth Century: A Practical Guidebook* par Kurt Stone [W. W. Norton, 1980]
- *Music Notation: A Manual of Modern Practice* par Gardner Read [Taplinger, 1979]
- *Instrumentation and Orchestration* par Alfred Blatter [Schirmer, 2de ed. 1997]

16.6.2 Partitions et exemples

17 Notations anciennes



La gestion par LilyPond des formes de notation ancienne inclut des fonctionnalités spécifiques à la notation mensurale, au chant grégorien et à la notation de style kiévien. Ces fonctionnalités sont accessibles en modifiant les propriétés de style des objets graphiques tels que tête de note ou silence, ou bien grâce aux contextes prédéfinis à cet effet.

De nombreux objets graphiques – « grobs » dans le jargon de LilyPond – disposent d'une propriété `style`. Manipuler cette propriété permet d'adapter l'aspect typographique des *grobs* à une forme de notation particulière, ce qui évite la création de nouveaux concepts de notation. Voir à ce sujet

- Section 17.3.4 [Têtes de note anciennes], page 558,
- Section 17.3.7 [Altérations et armures anciennes], page 561,
- Section 17.3.6 [Silences anciens], page 560,
- Section 17.3.2 [Clefs anciennes], page 557,
- Section 17.4.2 [Clefs grégoriennes], page 570,
- Section 17.3.5 [Crochets anciens], page 559,
- Section 17.3.3 [Métriques anciennes], page 558.

D'autres aspects de la notation ancienne ne peuvent pas être gérés aussi simplement qu'en jouant sur les propriétés d'un style appliqué à un objet graphique ou en lui ajoutant des articulations. Certains concepts sont spécifiques à la notation ancienne :

- Section 17.2.3 [Guidons], page 555,
- Section 17.4.4 [Divisions], page 571,
- Section 17.2.2 [Ligatures], page 554.

Voir aussi

Glossaire musicologique : Section “custos” dans *Glossaire*, Section “ligature” dans *Glossaire*, Section “mensural notation” dans *Glossaire*.

Manuel de notation : Section 17.3.7 [Altérations et armures anciennes], page 561, Section 17.4.2 [Clefs grégoriennes], page 570, Section 17.3.5 [Crochets anciens], page 559, Section 17.4.4 [Divisions], page 571, Section 17.2.3 [Guidons], page 555, Section 17.2.2 [Ligatures], page 554, Section 17.3.3 [Métriques anciennes], page 558, Section 17.3.6 [Silences anciens], page 560, Section 17.3.4 [Têtes de note anciennes], page 558.

17.1 Formes de notation ancienne prises en charge

En matière de chant grégorien, LilyPond dispose de trois différents styles :

- *Editio Vaticana* constitue un style à part entière dédié au chant grégorien, le plus approchant possible des éditions de Solesmes, éditeur officiel du Vatican depuis 1904. LilyPond dispose de tous les signes de notation propres à ce style, y compris les ligatures, custodes et certaines spécificités comme le quilisma et l’oriscus.
- *Editio Medicaea* dispose d’un certain nombre de spécificités des éditions qui faisaient autorité avant Solesmes. On le connaît aussi sous le nom de Ratisbone. Ce qui le distingue le plus du style *Vaticana* réside dans les clefs, en forme de barres obliques, et les têtes de note, plus carrées et régulières.
- Le style *Hufnagel* (« clou de fer à cheval ») ou *gothique* imite le style des manuscrits médiévaux d’Allemagne et d’Europe centrale. Il tire son nom de l’allure des notes (en virgule ou *virga*) qui ressemblent à des têtes de clou.

LilyPond dispose de trois styles imitant les manuscrits du Bas Moyen Âge et de la Renaissance ainsi que les premières impressions de musique mesurée.

- Le style *Mensural* est celui qui se rapproche le plus des manuscrits de la période allant de la fin du Moyen Âge au début de la Renaissance, avec ses petites têtes de note en forme de losange étroit et ses silences comme dessinés à main levée.
- Le style *Neomensural* est une version moderne et stylisée du style mensural : les têtes de note sont un peu plus galbées et les silences plus rectilignes. Ce style est tout à fait approprié à l’incipit d’une transcription de musique ancienne.
- Le style *Petrucci* tire son nom du fameux graveur vénitien Ottaviano Petrucci (1466-1539), premier imprimeur à utiliser des caractères amovibles pour la musique dans son édition du *Harmonice musices odhecaton* en 1501. Les têtes de notes de ce style sont plus larges que pour les autres styles mensuraux.

Bien qu’il ne soient pas complets, les styles *Baroque* et *classical* diffèrent du style par défaut par quelques détails – certaines têtes de note pour le *Baroque* et le soupir pour le *Classical*.

Seul le style mensural dispose de signes alternatifs couvrant tous les aspects de la notation. Ainsi, les silences et les crochets sont absents du style grégorien puisqu’ils ne sont pas utilisés dans la notation du plain-chant ; le style Petrucci ne dispose en propre d’aucun crochet ni d’altération.

Chacun des éléments de notation peut donc être modifié de manière indépendante jusqu’à, pourquoi pas, utiliser dans une même partition des crochet en *Mensural*, des têtes de note de *Petrucci*, des silences du *Classical* et des clefs du style *Vaticana*.

Voir aussi

Glossaire musicologique : Section “flag” dans *Glossaire*, Section “mensural notation” dans *Glossaire*.

17.2 Considérations communes aux musiques anciennes

17.2.1 Contextes prédéfinis

LilyPond dispose, en matière de musique ancienne et de grégorien, de contextes prédéfinis. Ceux-ci contiennent tout ce qui est nécessaire à la gestion d'une voix, d'une portée ou d'une partition selon le style adopté. Si tout cela vous dépasse et que vous désirez plonger dans le vif du sujet sans trop vous préoccuper d'ajuster des contextes, consultez les pages dédiées aux contextes prédéfinis. Ils vous permettront d'adapter vos contextes de voix et de portée, et vous n'aurez plus qu'à saisir les notes dans un contexte `VaticanaScore`, `VaticanaVoice`, `VaticanaStaff`, `MensuralVoice`, `MensuralStaff`, `PetrucciStaff`, `+PetrucciVoice`, `KievanVoice` ou `KievanStaff`.

Voir aussi

Glossaire musicologique : Section "mensural notation" dans *Glossaire*.
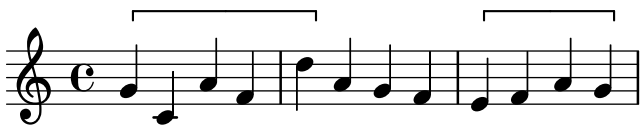
Manuel de notation : Section 17.4.1 [Contextes du chant grégorien], page 569, Section 17.3.1 [Contextes de musique mensurale], page 556, Section 17.5.1 [Contextes de notation kiévienne], page 579.

17.2.2 Ligatures

Une ligature est un symbole graphique qui représente un groupe d'au moins deux notes distinctes. Les ligatures ont commencé à apparaître dans les manuscrits de chant grégorien, pour indiquer des suites ascendantes ou descendantes de notes.

Les ligatures s'indiquent par un bornage entre `\[` et `\]`. Certains styles de ligature peuvent demander un complément de syntaxe spécifique. Par défaut, le graveur `LigatureBracket` place un simple crochet au dessus de la ligature :

```
\relative {
  \[ g' c, a' f d' \]
  a g f
  \[ e f a g \]
}
```



Trois autres styles de ligature sont accessibles : vatican pour le grégorien, mensural pour la musique ancienne (seules sont disponibles les ligatures mensurales blanches, avec quelques limitations) et kiévienne. Selon le style de ligature désiré, il faut remplacer, dans le contexte `Voice` approprié, le graveur `Ligature_bracket_engraver` par le graveur de ligature qui convient – voir les rubriques Section 17.3.10 [Ligatures mensurales], page 563, Section 17.4.7 [Neumes et ligatures grégoriennes], page 573, et Section 17.5.6 [Mélismes kiéviens], page 581, à ce sujet.

Voir aussi

Glossaire musicologique : Section "ligature" dans *Glossaire*.

Manuel de notation : Section 17.3.10 [Ligatures mensurales], page 563, Section 17.4.7 [Neumes et ligatures grégoriennes], page 573.

Problèmes connus et avertissements

La gestion de l'espacement spécifique aux ligatures n'est à ce jour pas implémentée. En conséquence, les ligatures sont trop espacées les unes des autres et les sauts de ligne mal ajustés.

Les paroles ne s'alignent pas de manière satisfaisante en présence de ligatures.

Les altérations ne pouvant être imprimées à l'intérieur d'une ligature, il faut les rassembler et les imprimer juste avant.

La syntaxe utilisée correspond à l'ancienne convention de préfixage `\[ expression_musicale \]`. Pour des raisons d'uniformité, nous opterons probablement pour le style en suffixe (postfix) `note\[ ... note\]`.

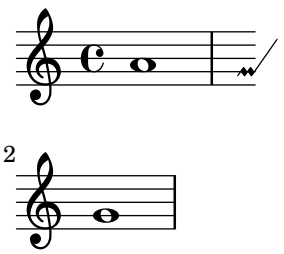
17.2.3 Guidons

Un guidon — *custos*, pluriel *custodes* en latin — est un symbole qui apparaît à la fin d'une portée. Il montre la hauteur de la ou des premières notes de la portée suivante, donnant une indication judicieuse à l'exécutant.

Les guidons étaient couramment utilisés jusqu'au XVIIe siècle. De nos jours, on les retrouve uniquement dans quelques formes particulières de notation telles que les éditions contemporaines de chant grégorien comme les *editio vaticana*. Différents glyphes existent selon le style de notation.

L'impression de guidons s'obtient en affectant, dans un bloc `\layout`, le Section "Custos_engraver" dans *Référence des propriétés internes* au contexte `Staff` et, selon les besoins, d'en spécifier le style comme le montre l'exemple suivant.

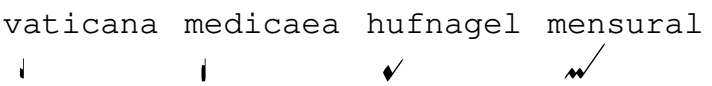
```
\score {
  \relative {
    a'1
    \break
    g
  }
  \layout {
    \context {
       \Staff
       \consists Custos_engraver
       \override Custos.style = #'mensural
    }
  }
}
```



Le glyphe du guidon est déterminé par la propriété `style`. Les styles disponibles sont `vaticana`, `medicaea`, `hufnagel` et `mensural`.

```
\new Lyrics \lyricmode {
  \markup { \column {
    \typewriter "vaticana "
    \line { " " \musicglyph "custodes.vaticana.u0" }
  } }
  \markup { \column {
    \typewriter "medicaea "
    \line { " " \musicglyph "custodes.medicaea.u0" }
  }}
```

```
  \markup { \column {
    \typewriter "hufnagel "
    \line { " " \musicglyph "custodes.hufnagel.u0" }
  }}
  \markup { \column {
    \typewriter "mensural "
    \line { " " \musicglyph "custodes.mensural.u0" }
  }}
}
```



Voir aussi

Glossaire musicologique : Section “custos” dans *Glossaire*.

Référence des propriétés internes : Section “Custos” dans *Référence des propriétés internes*.

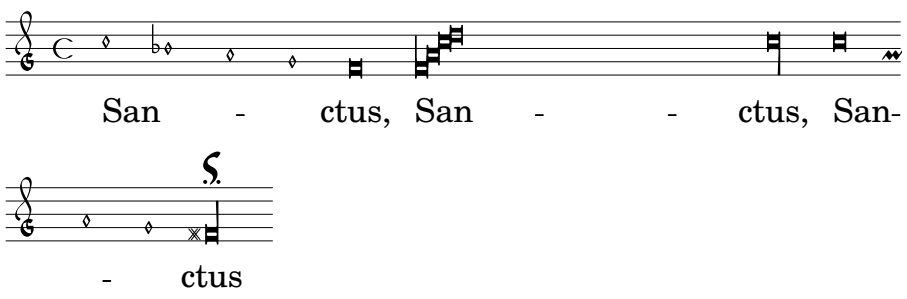
Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.

17.3 Typographie de musique ancienne

17.3.1 Contextes de musique mensurale

Les contextes `MensuralVoice` et `MensuralStaff` permettent de graver des chants dans le style mesuré. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant comme ci-après :

```
\score {
  <<
    \new MensuralVoice = "discantus" \relative {
      \hide Score.BarNumber {
        c''1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c\breve d\melismaEnd \]
        c\longa
        c\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```



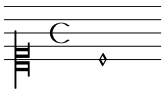
Voir aussi

Glossaire musicologique : Section “mensural notation” dans *Glossaire*.
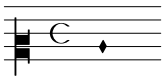
17.3.2 Clefs anciennes

Les clefs dédiées à la musique ancienne sont disponibles à l’aide de la commande `\clef`. Certaines de ces clés utilisent le même glyphe, attaché à l’une ou l’autre des lignes de la portée. Le chiffre porté en suffixe permet alors de les différencier, en partant de la ligne inférieure.
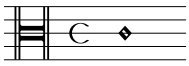
```
\new MensuralStaff {
\clef "mensural-c1" c'1
}
```
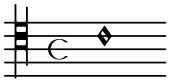


```
\new MensuralStaff {
  \override NoteHead.style = #'blackmensural
  \clef "blackmensural-c2" c'1
}
```



```
\new MensuralStaff {
  \override NoteHead.style = #'neomensural
  \clef "neomensural-c3" c'1
}
```



```
\new PetrucciStaff {
  \clef "petrucci-c4" c'1
}
```



Un glyphe de clef peut tout à fait arbitrairement être positionné sur une ligne particulière, comme indiqué à la rubrique Section 1.3.1 [Clefs], page 20. Une liste exhaustive des différentes clefs est disponible à l’annexe Section B.11 [Styles de clef], page 939.

Voir aussi

Glossaire musicologique : Section “clef” dans *Glossaire*, Section “mensural notation” dans *Glossaire*.

Manuel de notation : Section 1.3.1 [Clefs], page 20, Section 17.4.2 [Clefs grégoriennes], page 570.

Fichiers d’initialisation : `scm/parser-clef.scm`.

Morceaux choisis : Section “Hauteurs” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Clef” dans *Référence des propriétés internes*, Section “Clef_engraver” dans *Référence des propriétés internes*, Section “clef-interface” dans *Référence des propriétés internes*, Section “ClefModifier” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La clef de sol mensurale est calquée sur celle de Petrucci.

17.3.3 Métriques anciennes

Les chiffrages de métrique mensurale sont partiellement pris en charge. Les glyphes ne font que représenter des métriques particulières. En d'autres termes, pour obtenir le glyphe correspondant à une métrique mensurale particulière à l'aide de la commande `\time` *n*`/`*m*, vous devez choisir *n* et *m* parmi les valeurs suivantes :

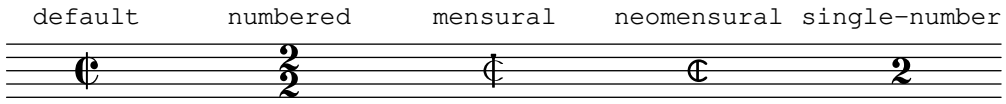
`\time 4/4` `\time 2/2` `\time 6/4` `\time 6/8`

`\time 3/2` `\time 3/4` `\time 9/4` `\time 9/8`

`\time 4/8` `\time 2/4`

La propriété `style` de l'objet `TimeSignature` permet d'accéder aux indicateurs de métrique anciens. Les styles `neomensural` et `mensural` sont disponibles. Vous avez vu ci-dessus le style `neomensural`, particulièrement utilisé pour l'incipit des transcriptions. Le style `mensural` imite l'aspect de certaines éditions du XVIe siècle.

Voici les différences entre les styles :



La rubrique Section 2.3.2 [Métrique], page 81, expose les principes généraux sur l'utilisation des indications de métrique.

Voir aussi

Glossaire musicologique : Section "mensural notation" dans *Glossaire*.

Manuel de notation : Section 2.3.2 [Métrique], page 81.

Problèmes connus et avertissements

Les équivalences de durées de note ne sont pas modifiées par un changement de métrique. Par exemple, l'équivalence une brève pour trois semi-brèves (*tempus perfectum*) doit s'effectuer à la main en entrant :

```
breveTP = #(ly:make-duration -1 0 3/2)
...
{ c\breveTP f1 }
```

Ce qui définira `breveTP` à 3/2 fois 2 = 3 fois une ronde.

Les symboles `mensural68alt` et `neomensural68alt` – alternatives à la métrique 6/8 – ne sont pas accessibles par la commande `\time`. Utilisez alors un `\markup {\musicglyph "timesig.mensural68alt" }`.

17.3.4 Têtes de note anciennes

Pour de la musique ancienne, vous disposez de plusieurs styles de tête de note, en plus du style par défaut `default`. Vous pouvez affecter à la propriété `style` de l'objet `NoteHead` les valeurs `baroque`, `neomensural`, `mensural`, `petrucci`, `blackpetrucci` ou `semipetrucci`.

Le style baroque diffère du style default par

- la disponibilité de la maxima, et
- la \breve qui sera carrée et non pas ovoïde.

Les styles neomensural, mensural et petrucci diffèrent du baroque par ceci :
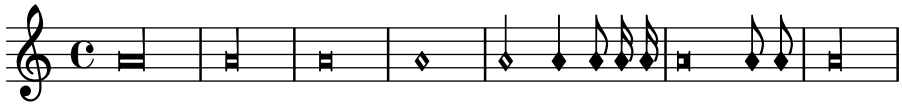
- les notes de durée inférieure ou égale à une ronde sont en forme de losange, et
- les hampes sont centrées sur la tête.

Le style blackpetrucci permet d'obtenir, en notation mensurale blanche, des têtes noircies. Cependant, et dans la mesure où le style de tête n'influence en rien le nombre des crochets, une *semiminima* devrait alors se noter a8*2 plutôt que a4, de telle sorte qu'elle ne se confonde pas avec une *minima*. Le multiplicateur peut varier, pour indiquer par exemple un triolet.

Le style semipetrucci permet de partiellement noircir certaines têtes, comme la brève, la longue et la maxime.

L'exemple suivant illustre le style petrucci.

```
\compressEmptyMeasures
\autoBeamOff
\override NoteHead.style = #'petrucci
a'\maxima a'\longa a'\breve a'1 a'2 a'4 a'8 a'16 a'
\override NoteHead.style = #'semipetrucci
a'\breve*5/6
\override NoteHead.style = #'blackpetrucci
a'8*4/3 a'
\override NoteHead.style = #'petrucci
a'\longa
```



La rubrique Section 1.4 [Têtes de note], page 45, présente tous les styles de notes disponibles.

Voir aussi
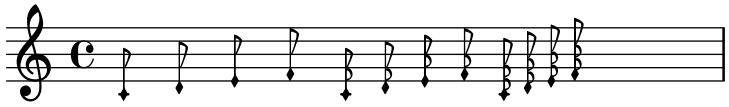
Glossaire musicologique : Section "mensural notation" dans *Glossaire*, Section "note head" dans *Glossaire*.

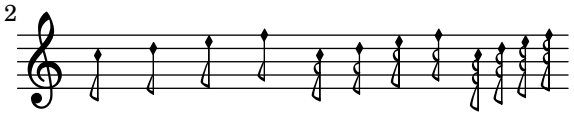
Manuel de notation : Section 1.4 [Têtes de note], page 45.

17.3.5 Crochets anciens

Le réglage de la propriété style de l'objet Flag donne accès aux crochets de style ancien. Les seuls styles actuellement pris en charge sont default et mensural.

```
\relative c' {
  \override Flag.style = #'mensural
  \override Stem.thickness = 1.0
  \override NoteHead.style = #'mensural
  \autoBeamOff
  c8 d e f c16 d e f c32 d e f s8
  c'8 d e f c16 d e f c32 d e f
}
```





Notez que, pour chaque crochet mensural, l'extrémité la plus proche de la tête de note sera attachée à une ligne de la portée.

Il n'existe pas de crochet spécifique au style néomensural.

Les crochets n'existent pas en notation grégorienne.

Voir aussi

Glossaire musicologique : Section "flag" dans *Glossaire*, Section "mensural notation" dans *Glossaire*.

Problèmes connus et avertissements

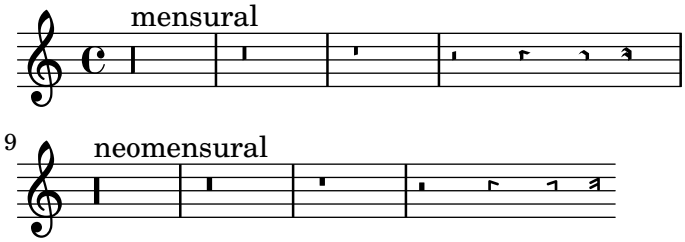
L'alignement vertical des crochets par rapport aux lignes de la portée sous-entend que les hampes se terminent toujours soit sur une ligne, soit à l'exact milieu d'un interligne. Ceci n'est pas toujours réalisable, surtout si vous faites appel à des fonctionnalités avancées de présentation de la notation classique qui, par définition, ne sont pas prévues pour être appliquées à la notation mensurale.

17.3.6 Silences anciens

La propriété `style` de l'objet `Rest` permet d'obtenir des silences de type ancien. Vous disposez des styles `mensural` et `neomensural`.

En voici une illustration.

```
\compressEmptyMeasures
\override Rest.style = #'mensural
r\longa^"mensural" r\breve r1 r2 r4 r8 r16 s \break
\override Rest.style = #'neomensural
r\longa^"neomensural" r\breve r1 r2 r4 r8 r16
```



Les styles `mensural` et `neomensural` ne disposent pas des huitième et seizième de soupir ; LilyPond utilise dans de tels cas le style par défaut.

Voir aussi

Glossaire musicologique : Section "mensural notation" dans *Glossaire*.

Manuel de notation : Section 2.2.1 [Silences], page 69.

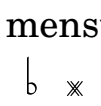
Morceaux choisis : Section "Notations anciennes" dans *Morceaux choisis*.

Problèmes connus et avertissements

En style mensural, LilyPond utilise pour un silence correspondant à une *maxima*, le même glyphe que pour une *longa* ; il faut donc les multiplier pour obtenir la durée *ad hoc*. Des silences correspondant à une *longa* ne sont pas groupés automatiquement ; utilisez en pareil cas des « notes silencieuses ».

17.3.7 Altérations et armures anciennes

Le style `mensural` dispose d'un dièse et d'un bémol différents du style par défaut. La notation mensurale n'utilise que très rarement le bécarre ; sont utilisés plutôt le dièse ou le bémol. Par exemple, un si bécarre en fa majeur sera indiqué par la présence d'un dièse. Si toutefois il était requis, le bécarre sera emprunté au style `vaticana`.



La manière d'utiliser ce style est abordée dans Section 1.3.6 [Glyphes d'altération alternatifs], page 40. Il est activé par défaut dans un contexte `MensuralStaff`.

Voir aussi

Glossaire musicologique : Section "accidental" dans *Glossaire*, Section "key signature" dans *Glossaire*, Section "mensural notation" dans *Glossaire*, Section "Pitch names" dans *Glossaire*.

Manuel de notation : Section 1.1.3 [Altérations], page 8, Section 1.3.5 [Altérations accidentelles automatiques], page 32, Section 1.3.2 [Armure], page 25, Section 1.3.6 [Glyphes d'altération alternatifs], page 40, Chapitre 1 [Hauteurs], page 3, Section B.10 [Jeux de glyphes d'altération], page 938.
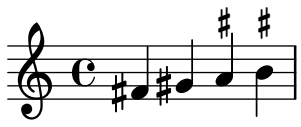
Référence des propriétés internes : Section "KeySignature" dans *Référence des propriétés internes*.

17.3.8 Altérations suggérées (*musica ficta*)

Dans la pratique ancienne, avant le XVIIe siècle, les altérations accidentelles de l'échelle modale n'étaient pas systématiquement notées et il incombait aux chanteurs, en fonction de certaines règles, de décider s'ils devaient chanter tel degré bémol, bécarre ou dièse. Cette technique est appelée *musica ficta*. Les transcriptions modernes de telles œuvres font apparaître ces altérations en surplomb de la note.

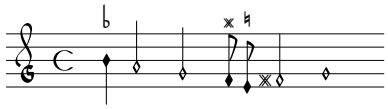
La reproduction de ces altérations suggérées est assurée par l'activation de la fonction `suggestAccidentals`.

```
\relative {
  fis' gis
  \set suggestAccidentals = ##t
  ais bis
}
```



Cette fonction considérera **toute** altération comme étant de la *musica ficta*, ce tant qu'elle n'aura pas été désactivée par un `\set suggestAccidentals = ##f`. Il est de ce fait plus pratique de recourir à une clause `\once \set suggestAccidentals = ##t`, qui peut tout à fait faire l'objet d'un raccourci :

```
ficta = { \once \set suggestAccidentals = ##t }
\score { \relative
  \new MensuralVoice  {
    \once \set suggestAccidentals = ##t
    bes'4 a2 g2 \ficta fis8 \ficta e! fis2 g1
  }
}
```



Voir aussi

Référence des propriétés internes : Section “Accidental_engraver” dans *Référence des propriétés internes*, Section “AccidentalSuggestion” dans *Référence des propriétés internes*.

17.3.9 Aperçu historique des ligatures

La polyphonie ancienne utilisait un rythme modal et indiquait la métrique par des ligatures (comme la notation grégorienne en neumes – voir Section 17.4.7 [Neumes et ligatures grégoriennes], page 573), rompant le flux de notes **non au pied** mais après chaque accentuation métrique. Autrement dit, la dernière note d'une ligature était accentuée, contrairement aux notes précédentes. Les ligatures les plus usitées comportaient deux ou trois notes (*ligatura binaria/ternaria* : ligature binaire/ternaire), correspondant à la distribution des accents dans les métriques communes *iambe* (court-long), *trochée* (long-court), *dactyle* (long-court-court) et *anapaest* (court-court-long). Il est à noter que long-court est le concept quantitatif ; l'accent est la contrepartie qualitative de long.

Lorsque le rythme modal a été abandonné et que sont apparues des notes galbées pour représenter leur longueur, les ligatures n'ont pas disparu mais, en interprétant la métrique de façon strictement quantitative, ont été lues comme des *brevis* suivies d'une *longa* finale. Le *pes* () indique une ligature ascendante, et le *climacus* () une descendante.

De nouveaux galbes ont été créés pour codifier d'autres séquences de longueurs, de préférence indépendamment du contexte, de sorte à pouvoir obtenir des ligatures de longueur arbitraire. Des galbes pour des ligatures binaires *brevis-brevis* auraient pu être tirées à partir des deux premières notes de ligatures ternaires existantes (signifiant *brevis-brevis-longa*) : le *torculus* () et le *scandici* modifié () s'entament de la même manière ; le galbe est ainsi utilisable en tant que ligature binaire *brevis-brevis* ascendante. Le *climacus* devra être abandonné car indistinguable d'une séquence de notes non ligaturées, donc le *porrectus* () a demeuré dans l'utilisation de son départ pour la forme descendante : le parallélogramme oblique  est connu sous le terme de *flexa* ; il indique toujours **deux** notes (hauteurs de départ et d'arrivée), non pas une gamme sur toutes les hauteurs entre les deux.

Les ligatures binaires démarrant par une *longa* ont été dessinées en adaptant les galbes existants *brevis-brevis* ou *brevis-longa*. Pour les galbes descendants *longa-breve* () et *longa-longa* (), la hampe initiale a été supromée ; pour les galbes ascendants *longa-breve* () et *longa-longa* () a été ajoutée une hampe descendante sur la droite de la première note.

Le *pes* étant difficile à dessiner avec des têtes vides si l'avant-dernière hauteur était proche (même constat avec un *porrectus*), les notes finales étaient remplacées par des *virgae* (par exemple, et  , peut-être inspirées du *scandicus*). Le galbe originel est toutefois resté en usage encore longtemps, bien qu'il ressemble à un *divisi* et qu'il soit pratiquement impossible d'en distinguer la signification en fin de section.

Des ligatures binaires pour paires de *semibreve* comme , et sont aussi apparues, reconnaissables par une hampe ascendante sur la gauche.

Le tableau suivant résume les ligatures binaires habituelles.

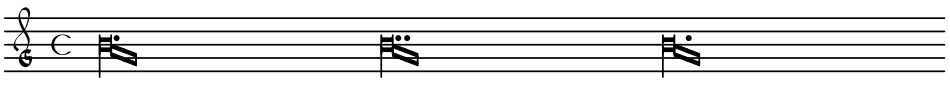
Longueurs	**S S**	**B B**	**L B**	**B L**	**L L**
Ascendante					
Descendante	,				

L'introduction des *maximae* dans les ligatures a été beaucoup plus simple dans la mesure où leur apparition est postérieure à l'ère du rythme modal. Elles n'ont donc pas besoin de se plier aux anciennes règles : elles peuvent intervenir n'importe où dans une ligature de forme habituelle, avec ou sans hampe descendante sur sa droite. Au sein même d'une ligature, brèves et longues apparaissaient toujours dans leur galbe habituel (longues ayant toujours une hampe descendante sur la droite), sauf lorsque les têtes d'une paire de brèves étaient fusionnées pour former une flexe (qui ne peut obligatoirement se trouver qu'en fin mais souvent utilisée en cours). Il arrivait rarement que les brèves initiales soient pourvues d'une hampe sur la gauche ou que les longues finales en aient une à droite même dans des configurations descendantes.

La ligature de neumes grégoriens ne pouvaient disparaître puisque la notation du plain-chant restait en usage – certains manuscrits contiennent en parallèle du plain-chant et de la notation mensurale. L'évolution des ligatures mensurales en a été grandement influencée, ce qui pourrait expliquer l'utilisation presque exclusive de brèves à l'intérieur même des ligatures. Le fait que le galbe de la flexe est principalement utilisé lorsque la seconde note est plus basse que celles qui l'entourent peut être vu comme un héritage des neumes.

Le tableau ci-dessus constitue presque un aide-mémoire pour la lecture de n'importe quelle ligature mensurale : les première et dernière notes devraient être déchiffrées à partir de cette table ; le reste peut se lire tel quel – pour des règles « exactes », voir Section 17.3.11 [Lecture des ligatures mensurales blanches], page 566. Le galbe de la flexe pose quelques difficultés pratiques : il peut s'avérer difficile de décider des hauteurs selon qu'elles se trouvent sur une ligne ou dans un espace adjacent ; les notes pourraient aussi être pointées (un tel point apparaît au-dessus ou au-dessous de la note suivante), et il n'est pas évident de déterminer si le point au-dessus d'une flexe concerne la première note de la flexe ou à la précédente :

```
\new PetrucciVoice {
  \relative {
    \[ b'\breve. a\breve g\breve \]
    \[ b\breve. a\breve. g\breve \]
    \[ b\breve a\breve. g\breve \]
  }
}
```



17.3.10 Ligatures mensurales

Les ligatures mensurales blanches sont prises en charge, avec des limitations.

La gravure des ligatures mensurales blanches s'obtient après avoir remplacé, dans le contexte `Voice` du bloc `layout`, le `Ligature_bracket_engraver` par le `Mensural_ligature_engraver`, comme ici :

```
\layout {
  \context {
    \Voice
    \remove Ligature_bracket_engraver
```

```
      \consists Mensural_ligature_engraver
    }
  }
```

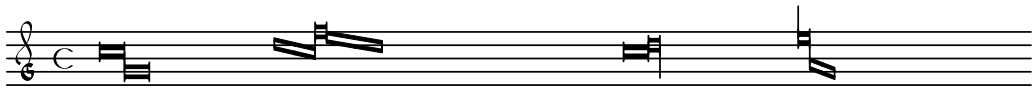
Dans ce qui suit, nous utilisons un contexte `PetrucciStaff` qui, entre autres, opère ce remplacement, et se rapproche de la typographie du *Harmonices Musices Odhecaton* d'Ottaviano Petrucci (Venise, 1501). Le contexte de voix qui lui est dévolu est un `PetrucciVoice`.

Lorsque le code ci-dessus est employé, l'aspect d'une ligature mensurale blanche est déterminé à partir des hauteurs et durées des notes qui la composent. Bien que cela demande un temps d'adaptation au nouvel utilisateur, cette méthode offre l'avantage que toute l'information musicale incluse dans la ligature est connue en interne. Ceci est non seulement important pour le rendu MIDI, mais aussi pour des questions de transcription automatisée d'une ligature.

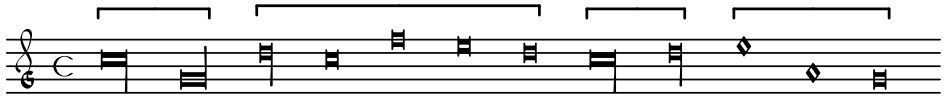
Par défaut, les hampes optionnelles et les flexes ne sont pas dessinées, et une longue finale ascendante est remplacée par une *virga*. L'impression d'une flexe s'obtient par affectation, pour la **seconde** note, de la propriété `ligature-flexa` ; le réglage de la largeur d'une flexe se gère par la propriété de tête de note `flexa-width`.

Par exemple,

```
\new PetrucciStaff \relative {
  \[ c''\maxima g \]
  \[ d'\longa
     \tweak ligature-flexa ##t
     \tweak flexa-width #3.2 c\breve f e
     \tweak flexa-width #4.3 d \]
  \[ c\maxima d\longa \]
  \[ e1 a, g\breve \]
}
```



Après avoir remplacé le `Mensural_ligature_engraver` par le `Ligature_bracket_engraver`, on obtient

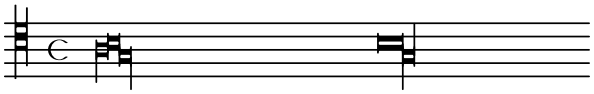


Des hampes optionnelles peuvent s'obtenir à l'aide des propriétés suivantes :

- `left-down-stem` pour une hampe descendante sur la gauche d'une brève en initiale (lorsque plus basse que la note qui la suit) ;
- `right-down-stem` pour une hampe descendante sur la droite d'une maxime ou d'une longue (qui est dépourvue de hampe dans le cas où le début ou la fin sont descendants) ;
- `right-up-stem` pour une hampe sur la droite d'une longue finale (lorsque plus basse que la note qui la précède).

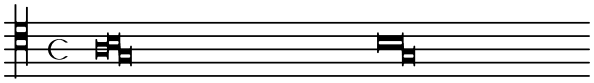
Voici un exemple illustrant ces retouches.

```
\new PetrucciStaff \relative {
  \clef "petrucci-c4"
  \[ \tweak left-down-stem ##t a\breve b
     \tweak right-down-stem ##t g\longa \]
  \[ \tweak right-down-stem ##t b\maxima
     \tweak right-up-stem ##t g\longa \]
}
```
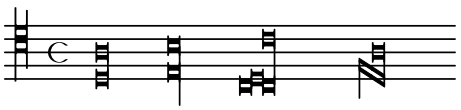


sans retoucher les ligatures, cela ressemblerait à

```
\new PetrucciStaff \relative {
  \clef "petrucci-c4"
  \[ a\breve b g\longa \]
  \[ b\maxima g\longa \]
}
```



Une longue fibake ascendante qui opte pour un style *pes* s'obtient en activant la propriété `ligature-pes`:
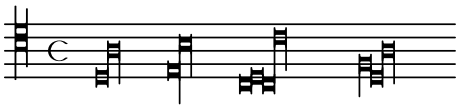
```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c4"
  \[ d\breve \tweak ligature-pes ##t a\longa \]
  \[ e \tweak ligature-pes ##t b \]
  \[ c\breve d c \tweak ligature-pes ##t c'\longa \]
  \[ f\breve d \tweak ligature-pes ##t a\longa \]
}
```



Dans ce dernier cas, LilyPond dessine les brèves comme des flexes, autrement la note initiale entrerait en collision avec la note finale. Sans retoucher les ligatures, cela ressemblerait à

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
```

```
  \clef "petrucci-c4"
  \[ d\breve a\longa \]
  \[ e b \]
  \[ c\breve d c c'\longa \]
  \[ f\breve d a\longa \]
}
```



Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

Manuel de notation : Section 17.2.2 [Ligatures], page 554, Section 17.4.7 [Neumes et ligatures grégoriennes], page 573.

Problèmes connus et avertissements

L’espacement horizontal peut laisser à désirer.

Les altérations peuvent se chevaucher avec les notes précédentes.

17.3.11 Lecture des ligatures mensurales blanches
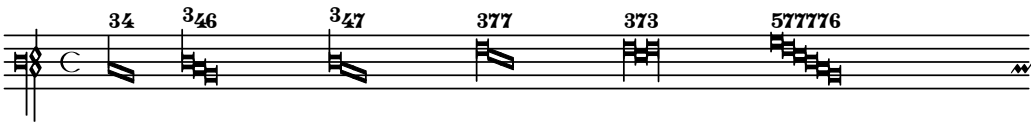
Les ligatures se lisent de gauche à droite et (si parfois deux notes paraissent verticalement alignée) de haut en bas. La ligature doit tout d’abord être décomposée en notes – une flexe signifie deux notes, chaque note peut être pointée (voir ci-dessus) ou colorée. De plus, chaque hampe est assignée à une note particulière : une hampe descendante en fin ou au sein de la ligature affecte la note qui précède ou, dans le cas de deux notes verticalement alignées, à la plus basse d’entre elles (comme serait une hampe sur la droite de cette note). Par contre, une hampe ascendante au début ou en cours de ligature s’attache à la note qui suit (comme l’est une hampe sur la gauche – les hampes ascendantes au sein de ligatures sont extrêmement rares).

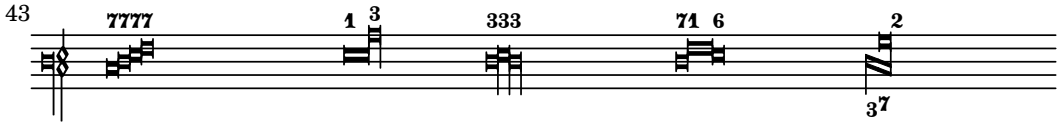
En suite est attachée à chaque note une durée selon les règles suivantes :

1. (règle de la *maxima*) Si la note ressemble à une maxime, sa longueur est alors `\maxima`, sinon
2. (règle du *pes*) Si la note est alignée au-dessus de la note précédente, sa longueur est `\longa`, sinon
3. (règle des hampes) Si la note est pourvue d’une hampe, sa longueur est semi-brève (`1`) pour une hampe ascendante sur la gauche, `\breve` pour une hampe descendante sur la gauche, et `\longa` pour une hampe sur la droite ; sinon
4. (complément à la règle des hampes) Si la note précédente est pourvue d’une hampe ascendante sur sa gauche, sa longueur est semi-brève (`1`), sinon
5. (règle du début descendant) Si la note est la première et qu’elle est plus haute que la note qui la suit, sa longueur est `\longa`, sinon
6. (règle de la fin descendante) Si la note est la finale et qu’elle est plus basse que l’avant-dernière et a une allure de brève, sa longueur est `\longa`, sinon
7. La longueur est `\breve`.

Il est à noter que des longueurs inférieures à la semi-brève ne sauraient être ligaturées.

Voici quelques exemples, indiquant quelle règle s’applique à quelle note :





En dehors de ces règles existent des restrictions ou conventions :

A. Des notes successives ne peuvent avoir la même hauteur.

B. Des notes alignées verticalement ne peuvent intervenir qu'en fin de ligature.

C. Une ligature comporte au moins deux notes (on pourrait éventuellement observer des  ressemblant à une ligature unaire, mais il s'agirait probablement de rectification d'erreurs de recopie, où une brève serait corrigée en semi-brève par adjonction d'une hampe).

D. Des semi-brèves peuvent apparaître au début, et seulement une paire – certains manuscrits anglais comportent des ligatures ayant quatre ou six semi-brèves.

E. Les flexes sont descendantes – néanmoins, la présence de flexes ascendants est omniprésente dans des manuscrits ne respectant pas cette convention.

F. Les flexes non finales sont tournantes ; autrement dit, les flexes descendantes sont suivies par une note plus haute, et les ascendantes par une note plus basse.

G. La seconde note d'une flexe ne saurait être une longue ; en d'autres termes, une flexe ne peut avoir de hampe sur sa droite.

H. Les notes intermédiaires sont des brèves ; s'il s'agit réellement d'une règle explicite de Tinctoris, elle ne correspond pas exactement à la pratique mais peut être perfectionnée à plusieurs égards :

I. Des longues et des maximes intermédiaires apparaissent dans des ligatures constituées uniquement de longues et de maximes ;

J. Les longues intermédiaires sont plus hautes que la note précédentes ;

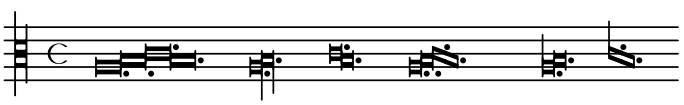
K. Les longues intermédiaires sont plus basses que la note suivante.

Il existe aussi une tendance selon laquelle les maximes dans une ligature mixte (non composée uniquement de maximes) sont pourvues d'une hampe sur leur droite, sauf si ses débuts et fins sont descendants.

Quelque exemples de l'application de ces restrictions :
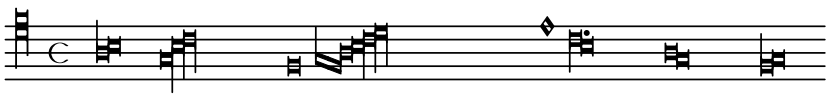
- Dans la *Missa Salve diva parens* d'Obrech, le même motif de quatre notes est écrit en longueurs décroissantes. Alors que les maximes (pointées) ou brèves peuvent constituer une ligature sur quatre notes, les longues et semi-brèves ne le peuvent pas (restrictions K et D).

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c3"
  \[ a\maxima. b c' b \]
  \[ a\longa. b \] \[ c' b \]
  \[ a\breve. b c' b \]
  \[ a1. b \] \[ c' \tweak ligature-flexa ##t b \]
}
```



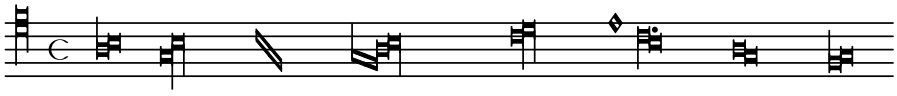
- L'*Amen* d'un Credo de Vaqueras est un long mélisme ; celui de la partie de ténor est écrit principalement en ligatures, respectant les restrictions D, J et K (mais ignorant H et I).

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c5"
  \[ f1 g \] \[ e\longa g a \] d\breve
  \[ e1 \tweak ligature-flexa ##t d f\breve g\longa a b \]
  c'1 \[ a\breve. g\longa \] \[ f e \] \[ d1 e \]
}
```



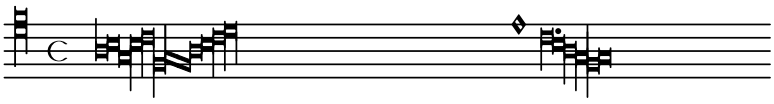
Observer la restriction H produit

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c5"
  \[ f1 g \] \[ e\longa g \] \[ a d\breve \]
  \[ e1 \tweak ligature-flexa ##t d f\breve g\longa \] \[ a b \]
  c'1 \[ a\breve. g\longa \] \[ f e \] \[ d1 e \]
}
```



alors qu'ignorer toutes les restrictions donne

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c5"
  \[ f1 g e\longa g a d\breve
     e1 \tweak ligature-flexa ##t d f\breve g\longa a b \]
  c'1 \[ a\breve. g\longa f e d1 e \]
}
```
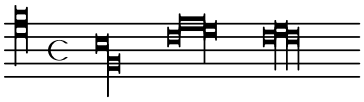


La règle 6 présente une énigme : comment dessiner une brève finale lorsque l'avant-dernière note est pourvue d'une hampe sur sa droite (ce qui est extrêmement rare en raison de la restriction K) ? La réponse simpliste est : « Stoppe la ligature après la hampe ». Il existe cependant des exemples où  doit se lire longue-brève (noter que l'ajout de la hampe sur la droite d'une longue en finale pourrait lever toute ambiguïté).

Une interprétation alternative consiste à considérer cette hampe comme la hampe attachée sur la droite de la dernière note, comme dans Trent 89, folio 97, dans lequel  doit se lire comme étant trois brèves.

LilyPond dessine `\breve` et `\longa` comme des têtes de *brevis* lorsque les notes finales descendent, mais permet l'activation de la propriété `right-down-stem` pour lever l'ambiguïté :

```
\layout {
  \context {
    \Score
    \override SpacingSpanner.base-shortest-duration =
      #(ly:make-moment 16 1)
  }
}
\new PetrucciStaff {
  \clef "petrucci-c5"
  \[ \tweak right-down-stem ##t g\longa d\breve \]
  \[ a\breve \tweak right-down-stem ##t c'\maxima b\longa \]
  \[ a\longa b \tweak right-down-stem ##t a \]
}
```



Tous ces exemples sont réalité. En fait, la deuxième ligature du dernier était à l'origine écrit  , et la hampe est une addition postérieure.

17.4 Typographie du chant grégorien

Si vous écrivez en notation grégorienne, le `Vaticana_ligature_engraver` se chargera de sélectionner les têtes de note appropriées ; il est donc inutile de spécifier le style à utiliser. Vous pouvez cependant spécifier par exemple le style `vaticana.punctum` pour obtenir des neumes punctums. De même, c'est le `Mensural_ligature_engraver` qui se chargera des ligatures mensurales.

Voir aussi

Glossaire musicologique : Section “ligature” dans *Glossaire*.

Manuel de notation : Section 17.2.2 [Ligatures], page 554, Section 17.3.10 [Ligatures mensurales], page 563.

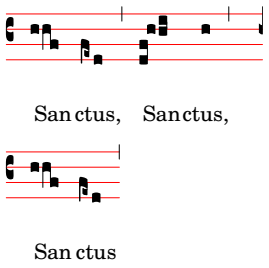
17.4.1 Contextes du chant grégorien

Les contextes prédéfinis `VaticanaVoice`, `VaticanaStaff`, `VaticanaLyrics` et `VaticanaScore`, permettent de graver le chant grégorien dans le style des éditions vaticanes. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates,

de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant, comme ci-dessous :

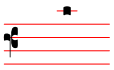
```
\new VaticanaScore {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new VaticanaLyrics \lyricsto "cantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}

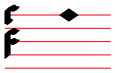
\layout {
  indent = 0
  ragged-last = ##t
}
```



17.4.2 Clefs grégoriennes

Le tableau suivant présente les différentes clefs grégoriennes que vous pouvez sélectionner avec la commande `\clef`. Certaines de ces clefs utilisent le même glyphe, attaché à l'une ou l'autre des lignes de la portée. Le chiffre porté en suffixe permet alors de les différencier ; la numérotation des lignes va de bas en haut. Vous pouvez néanmoins forcer le positionnement du glyphe sur une ligne, comme expliqué à la section Section 1.3.1 [Clefs], page 20. Dans la colonne exemple, la note suivant la clef est un do médium.

Description	**Clef disponible**	**Exemple**
Clef d'ut, style des éditions vaticanes	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Clef de fa, style des éditions vaticanes	`vaticana-fa1`, `vaticana-fa2`	
Clef d'ut, style Editio Medicaea	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	
Clef de fa, style Editio Medicaea	`medicaea-fa1`, `medicaea-fa2`	

Clef d'ut, style historique Hufnagel	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
Clef de fa, style historique Hufnagel	`hufnagel-fa1`, `hufnagel-fa2`	
Clef combinée ut/fa, style historique Hufnagel	`hufnagel-do-fa`	

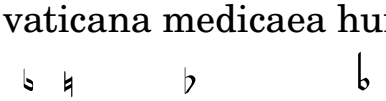
Voir aussi

Glossaire musicologique : Section "clef" dans *Glossaire*.

Manuel de notation : Section 1.3.1 [Clefs], page 20.

17.4.3 Altérations et armures grégoriennes

LilyPond dispose d'altérations pour les trois styles grégoriens :



Vous noterez que chacun de ces styles ne comporte pas toutes les altérations. LilyPond changera de style s'il est besoin d'une altération indisponible dans le style utilisé.

La manière de basculer entre les différentes formes est abordée dans Section 1.3.6 [Glyphes d'altération alternatifs], page 40.

Voir aussi

Glossaire musicologique : Section "accidental" dans *Glossaire*, Section "key signature" dans *Glossaire*.

Manuel de notation : Section 1.1.3 [Altérations], page 8, Section 1.3.5 [Altérations accidentelles automatiques], page 32, Section 1.3.2 [Armure], page 25, Section 1.3.6 [Glyphes d'altération alternatifs], page 40, Chapitre 1 [Hauteurs], page 3.

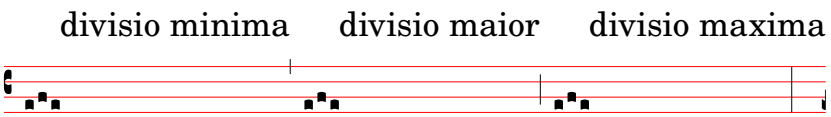
Référence des propriétés internes : Section "KeySignature" dans *Référence des propriétés internes*.

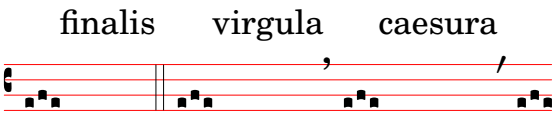
17.4.4 Divisions

Il n'existe pas de silence en notation grégorienne. On y parle plutôt de *divisions, pauses* ou *coupes*.

Une division – *divisio*, pluriel *divisiones* en latin – est un symbole ajouté à la portée et utilisé en chant grégorien pour séparer les phrases ou parties. *Divisio minima*, *divisio maior* et *divisio maxima* peuvent respectivement s'interpréter comme une pause courte, moyenne ou longue, à l'image des marques de respiration — cf. Section 3.2.3 [Signes de respiration], page 179. Le signe *finalis* n'est pas uniquement une marque de fin de chant ; il sert aussi à indiquer la fin de chaque partie dans une structure verset/répons.

Certaines éditions utilisent *virgula* ou *caesura* en lieu et place de *divisio minima* ; les contextes prédéfinis de chant grégorien sont configurés pour utiliser `\caesura` afin de produire une marque de césure.





Commandes prédéfinies

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Voir aussi

Glossaire musicologique : Section “caesura” dans *Glossaire*, Section “divisio” dans *Glossaire*.

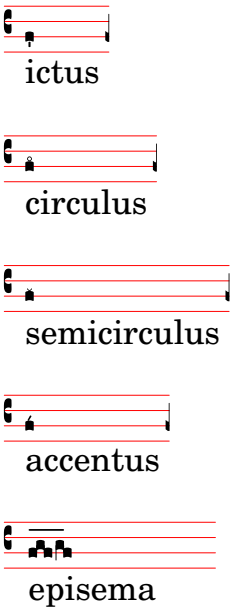
Manuel de notation : Section 3.2.3 [Signes de respiration], page 179.

17.4.5 Articulations grégoriennes

En plus des signes d’articulation standards décrits à la section Section 3.1.1 [Articulations et ornements], page 158, LilyPond fournit des articulations spécifiquement destinées au style des éditions vaticanes.

```
\new VaticanaScore {
  \new VaticanaVoice {
    \override Script.padding = -0.1
    a\ictus_"ictus " \break
    a\circulus_"circulus " \break
    a\semicirculus_"semicirculus "  \break
    a\accentus_"accentus " \break
    \[ a_"episema" \episemInitium \pes b
       \flexa a b \episemFinis \flexa a \]
  }
}

\layout {
  indent = 0
  ragged-last = ##t
}
```



Voir aussi

Manuel de notation : Section 3.1.1 [Articulations et ornements], page 158.

Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.

Référence des propriétés internes : Section “Episema” dans *Référence des propriétés internes*, Section “Episema_engraver” dans *Référence des propriétés internes*, Section “EpisemaEvent” dans *Référence des propriétés internes*, Section “Script” dans *Référence des propriétés internes*, Section “Script_engraver” dans *Référence des propriétés internes*, Section “ScriptEvent” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Certaines articulations sont verticalement trop proches de leurs têtes de note.

17.4.6 Points d’augmentation (*morae*)

Les points d’*augmentum*, ou *morae*, s’obtiennent avec la fonction `\augmentum`. Notez que cette fonction `\augmentum` est implémentée en tant que fonction unaire plutôt que comme un préfixe de note. Par conséquent, `\augmentum \virga c` ne donnera rien de particulier. Il faut l’utiliser avec la syntaxe `\virga \augmentum c` ou `\augmentum {\virga c}`. Par ailleurs, l’expression `\augmentum {a g}` constitue une forme abrégée de `\augmentum a \augmentum g`.

```
\new VaticanaScore {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Voir aussi

Manuel de notation : Section 3.2.3 [Signes de respiration], page 179.

Référence des propriétés internes : Section “Divisio” dans *Référence des propriétés internes*.

Morceaux choisis : Section “Musiques anciennes” dans *Morceaux choisis*.

17.4.7 Neumes et ligatures grégoriennes

Les neumes grégoriens, conformément au style des éditions vaticanes, sont pris en charge de façon assez limitée. Les ligatures élémentaires sont déjà disponibles, mais beaucoup de règles typographiques ne sont pas encore implémentées, notamment l’espacement horizontal des enchaînements de ligatures, l’alignement des paroles ou une gestion convenable des altérations.

Les têtes de note peuvent être *modifiées* ou *jointes*.

- L’aspect d’une tête de note se modifie en *préfixant* le nom d’une hauteur par l’une des commandes suivantes : `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.
- Une ligature, autrement dit la juxtaposition de notes, s’obtient en plaçant une commande de jointure `\pes` ou `\flexa` pour marquer une ligne mélodique respectivement ascendante ou descendante, entre les notes qui la composent.

  Une alternative à `\pes` et `\flexa`, qui fonctionnent en interne de la même manière, consiste à utiliser la commande `\~`.

Une hauteur sans qualificatif sera considérée comme un *punctum*. Tout autre neume, y compris ceux d’une seule note d’aspect particulier comme la *virga*, sera considéré en tant que ligature et devra répondre à la syntaxe `\[...\]`.

Neumes simples

- Le *punctum* représente l'aspect standard d'une note – dans le style *Vaticana*, il s'agit d'un carré plein légèrement incurvé pour une question d'esthétique. Existe aussi le *punctum inclinatum* – carré penché qui s'obtient grâce au préfixe `\inclinatum`. Un *punctum* standard peut se modifier par le préfixe `\cavum` qui l'évidera, ou le préfixe `\linea`, qui lui adjoindra une ligne verticale de part et d'autre.
- La *virga* dispose d'une hampe sur la droite. Elle s'obtient à l'aide du modificateur `\virga`.

Ligatures

Contrairement à la majorité des autres systèmes de notation neumatique, la manière de saisir les neumes n'a rien à voir avec leur apparence typographique ; elle se concentre plutôt sur le sens musical. Ainsi, `\[ a \pes b \flexa g \]` (ou `\[ a \~ b \~ g\]`) produit un *torculus* constitué de trois *punctums*, alors qu'une carrure haut-bas-haut telle que `\[ a \flexa g \pes b \]` (ou `\[ a \~ g \~ b\]`) produit un *porrectus* avec une flexe incurvée et un seul *punctum*. Il n'existe pas de commande à proprement parler qui permette de spécifier la courbe d'une flexe ; c'est la source musicale qui va le déterminer. Le fondement d'une telle approche réside dans la distinction que nous faisons entre les aspects musicaux de la source et le style de notation que nous voulons obtenir. De ce fait, la même source pourra être utilisée pour imprimer dans un autre style de notation grégorienne.

Neumes liquescents

Autre grande catégorie de notes que l'on trouve en grégorien, les neumes liquescents. Ils s'utilisent dans certaines circonstances, quand l'articulation d'une syllabe avec la suivante se fait par une « consonne liquide », sur la dernière note du groupe : M (omnis, summo), L, N, Y (ejus), NG (sanctus), W (autem, laudat). Ces consonnes ou semi-consonnes sont chantées à la hauteur correspondante, comme le seraient des voyelles, mais le chant est entravé par leur prononciation. De fait, les neumes liquescents ne sont jamais utilisés isolément (bien que rien ne l'interdise) et tombent toujours à la fin d'une ligature.

Les neumes liquescents peuvent se présenter graphiquement de deux façons différentes et relativement interchangeables : une note plus petite, ou une « bascule » verticale de la note principale. La première option s'obtient en créant un `pes` ou une `flexa` puis une modification de l'aspect de la deuxième note : `\[ a \pes \deminutum b \]`. La seconde option consiste à modifier l'aspect d'un neume sur note unique avec un `\auctum` tout en lui affectant une direction `\descendens` ou `\ascendens` : `\[ \auctum \descendens a \]`.

Signes spéciaux

Une troisième catégorie regroupe quelques signes dont la signification particulière diverge selon la source : la *quilisma*, l'*oriscus* et le *strophicus*. Ils s'obtiennent en préfixant la hauteur d'un `\quilisma`, `\oriscus` ou `\stropha`.

Il est virtuellement possible d'agglutiner autant de notes que voulu, y compris en les mélangeant avec des `\pes`, `\flexa`, `\virga`, `\inclinatum`, et de borner le tout par `\[` et `\]` pour produire une seule ligature. C'est d'ailleurs de cette manière que nous avons procédé pour générer le tableau qui suit. La création de ligatures est donc sans limite.

Notez bien que l'utilisation de ces signes en musique suit un certain nombre de règles, et que LilyPond n'effectue aucun contrôle à ce niveau. Par exemple, une *quilisma* se trouve toujours être la note intermédiaire d'une ligature ascendante et tombe habituellement sur un demi ton ; bien que cela soit tout à fait possible, mais parfaitement incorrect, rien ne vous empêche de créer une quilisma sur une seule note.

En plus des signes propres à la notation, LilyPond dispose des commandes `\versus`, `\responsum`, `\ij`, `\iij`, `\IJ` et `\IIJ`, qui permettent par exemple d'indiquer dans les paroles

des repères de section. Ces commandes font appel à des caractères unicode spécifiques qui ne seront reproduits que si vous utilisez une fonte qui en dispose.

Le tableau ci-dessous inventorie, bien que dans une certaine limite, les différents neumes contenus dans le second tome de l'Antiphonale Romanum (*Liber Hymnarius*) publié par l'abbaye de Solesmes en 1983. La première colonne énumère le nom des ligatures – forme normale en gras et forme liquescente en italique. La troisième colonne contient le code ayant permis de générer la ligature, se basant ici sur sol, la, si.

Neumes simples

Formes **Normale** et *Liquescente*	**Rendu**	**Code LilyPond**
Punctum		`\[ b \]`
		`\[ \cavum b \]`
		`\[ \linea b \]`
Punctum Auctum Ascendens		`\[ \auctum \ascendens b \]`
Punctum Auctum Descendens		`\[ \auctum \descendens b \]`
Punctum inclinatum		`\[ \inclinatum b \]`
Punctum Inclinatum Auctum		`\[ \inclinatum \auctum b \]`
Punctum Inclinatum Parvum		`\[ \inclinatum \deminutum b \]`
Virga		`\[ \virga b' \]`

Ligatures sur deux notes

Clivis vel Flexa		`\[ b \flexa g \]` (`\[ b \~ g \]`)
Clivis Aucta Descendens		`\[ b \flexa \auctum \descendens g \]`
Clivis Aucta Ascendens		`\[ b \flexa \auctum \ascendens g \]`

Cephalicus		`\[ b \flexa \deminutum g \]`
Podatus/Pes		`\[ g \pes b \]` (`\[ g \~ b \]`)
Pes Auctus Descendens		`\[ g \pes \auctum \descendens b \]`
Pes Auctus Ascendens		`\[ g \pes \auctum \ascendens b \]`
Epiphonus		`\[ g \pes \deminutum b \]`
Pes Initio Debilis		`\[ \deminutum g \pes b \]`
Pes Auctus Descendens Initio Debilis		`\[ \deminutum g \pes \auctum \descendens b \]`

Ligatures sur plusieurs notes

Torculus		`\[ a \pes b \flexa g \]` (`\[ a \~ b \~ g \]`)
Torculus Auctus Descendens		`\[ a \pes b \flexa \auctum` `\descendens g \]`
Torculus Deminutus		`\[ a \pes b \flexa \deminutum g \]`
Torculus Initio Debilis		`\[ \deminutum a \pes b \flexa g \]`
Torculus Auctus Descendens *Initio Debilis*		`\[ \deminutum a \pes b \flexa \auctum` `\descendens g \]`
Torculus Deminutus *Initio Debilis*		`\[ \deminutum a \pes b \flexa` `\deminutum g \]`
Porrectus		`\[ a \flexa g \pes b \]` `+`(`\[ a \~ g \~ b \]`)
Porrectus Auctus Descendens		`\[ a \flexa g \pes \auctum` `\descendens b \]`
Porrectus Deminutus		`\[ a \flexa g \pes \deminutum b \]`
Climacus		`\[ \virga b \inclinatum a` `\inclinatum g \]`
Climacus Auctus		`\[ \virga b \inclinatum a` `\inclinatum \auctum g \]`
Climacus Deminutus		`\[ \virga b \inclinatum a` `\inclinatum \deminutum g \]`
Scandicus		`\[ g \pes a \virga b \]`

Scandicus Auctus Descendens  `\[ g \pes a \pes \auctum \descendens b \]`

Scandicus Deminutus  `\[ g \pes a \pes \deminutum b \]`

Signes spéciaux

Quilisma  `\[ g \pes \quilisma a \pes b \]`

Quilisma Pes Auctus Descendens  `\[ g \quilisma g \pes \auctum \descendens b \]`

Oriscus `\[ \oriscus b \]`

Pes Quassus  `\[ \oriscus g \pes \virga b \]`

Pes Quassus Auctus Descendens  `\[ \oriscus g \pes \auctum \descendens b \]`

Salicus  `\[ g \oriscus a \pes \virga b \]`

Salicus Auctus Descendens  `\[ g \oriscus a \pes \auctum \descendens b \]`

(Apo)stropha  `\[ \stropha b \]`

Stropha Aucta  `\[ \stropha \auctum b \]`

Bistropha `\[ \stropha b \stropha b \]`

Tristropha `\[ \stropha b \stropha b \stropha b \]`

Trigonus  `\[ \stropha b \stropha b \stropha a \]`

Commandes prédéfinies

LilyPond dispose des préfixes suivants : `\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Les préfixes de note peuvent s'agglutiner, modulo quelques restrictions. Par exemple, on peut appliquer un `\descendens` ou un `\ascendens` à une note, mais pas les deux simultanément à une même note.

Deux notes adjacentes peuvent être reliées grâce aux commandes `\pes` ou `\flexa` pour marquer une ligne mélodique respectivement ascendante ou descendante. Une alternative à ces commandes consiste à utiliser `\~`.

Utilisez la fonction musicale unaire `\augmentum` pour ajouter des points d'augmentum.

Voir aussi

Glossaire musicologique : Section "ligature" dans *Glossaire*.

Manuel de notation : Section 17.2.2 [Ligatures], page 554, Section 17.3.10 [Ligatures mensurales], page 563.

Problèmes connus et avertissements

Lorsqu'un `\augmentum` apparaît dans une ligature en fin de portée, son placement vertical peut être erroné. Pour y remédier, ajoutez un silence invisible, `s8` par exemple, comme dernière note de cette portée.

L'`\augmentum` devrait être implémenté en tant que préfixe plutôt qu'en tant que fonction unaire, afin qu'`\augmentum` puisse s'intégrer avec d'autres préfixes dans n'importe quel ordre.
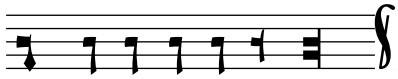
17.5 Typographie de notation kiévienne

17.5.1 Contextes de notation kiévienne

Tout comme pour les notations grégorienne et mensurale, les contextes prédéfinis `KievanVoice` et `KievanStaff` permettent de générer une partition en notation carrée. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant, comme ci-dessous :

```
% Font settings for Cyrillic
\paper {
  property-defaults.fonts.serif = "Linux Libertine O,serif"
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      c4 c c c c2 b\longa \fine
    }
    \new Lyrics \lyricsto "melody" {
      Го -- спо -- ди по -- ми -- луй.
    }
  >>
}
```



Господи помилуй.

Voir aussi

Glossaire musicologique : Section “notation kiévienne” dans *Glossaire*.
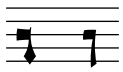
Problèmes connus et avertissements

LilyPond prend en charge la notation kiévienne du style synodal, correspondant au corpus du Saint Synode russe des années 1910, récemment réédité par les éditions du patriarcat de Moscou. LilyPond ne prend pas en charge les formes plus anciennes et moins répandues de notation kiévienne que l'on trouvait en Galicie pour noter le plain-chant ruthène.

17.5.2 Clefs kiéviennes

La notation kiévienne n'utilise qu'une seule clef – la clef « Tse-fa-ut » – qui indique la position du do :

```
\clef "kievan-do"
\kievanOn
c'
```



Voir aussi

Glossaire musicologique : Section “clef” dans *Glossaire*, Section “notation kiévienne” dans *Glossaire*.
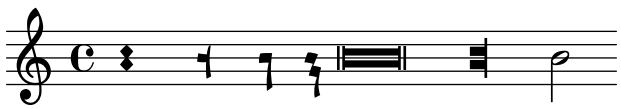
Manuel de notation : Section 1.3.1 [Clefs], page 20.

17.5.3 Notes kiéviennes

La notation kiévienne requiert l'utilisation d'un style de tête de note particulier et la désactivation des hampes et crochets classiques. La fonction `\kievanOn` se charge d'affecter les propriétés adéquates aux têtes de note, hampes et crochets. Un simple `\kievanOff` permet de retrouver le comportement par défaut de LilyPond.

En notation kiévienne, la note finale d'une pièce apparaît souvent sous la forme d'une `\longa`. L'indication d'un récitatif – plusieurs syllabes sont chantées sur une même hauteur – s'effectue à l'aide d'une `\breve`. Voici ce à quoi ressemblent les différentes notes kiéviennes :

```
\cadenzaOn
\kievanOn
b'1 b'2 b'4 b'8 b'\breve b'\longa
\kievanOff
b'2
```



Voir aussi

Glossaire musicologique : Section “notation kiévienne” dans *Glossaire*, Section “tête de note” dans *Glossaire*.

Manuel de notation : Section B.9 [Styles de tête de note], page 937.
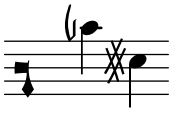
Problèmes connus et avertissements

LilyPond détermine automatiquement l'orientation des hampes. Il est cependant d'usage, en notation carrée, que les hampes des différentes notes d'un même mélisme aillent toutes dans le même sens ; il faudra donc en pareil cas définir manuellement la propriété `direction` de l'objet `Stem`.

17.5.4 Altérations kiéviennes

Le style `kievan` dispose d'un dièse et d'un bémol, tous deux différents du style par défaut ; il n'y a pas de bécarre en notation kiévienne. Bien que le dièse ne soit pas utilisé en notation synodale, on peut le trouver dans certains manuscrits plus anciens.

```
\clef "kievan-do"
\set Staff.alterationGlyphs =
  #alteration-kievan-glyph-name-alist
bes' dis'
```



Voir aussi
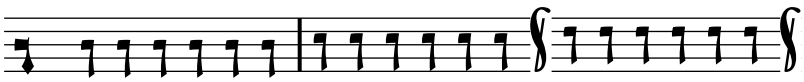
Glossaire musicologique : Section “altération” dans *Glossaire*, Section “notation kiévienne” dans *Glossaire*.

Manuel de notation : Section 1.1.3 [Altérations], page 8, Section 1.3.5 [Altérations accidentelles automatiques], page 32, Section 1.3.6 [Glyphes d'altération alternatifs], page 40, Section B.8 [La fonte Emmentaler], page 935.

17.5.5 Barres de mesure kiéviennes

Les pièces en notation kiévienne sont habituellement non-mesurées ; cependant, la commande `\caesura` insère une barre de phrase, et les commandes `\section` et `\fine` créent une barre de section ou finale.

```
\new KievanStaff {
  c'4 4 4 4 4 4 \caesura  % \bar "."
  d'4 4 4 4 4 4 \section  % \bar "k"
  e'4 4 4 4 4 4 \fine
}
```



Voir aussi

Manuel de notation : Section 2.5.1 [Barres de mesure], page 123, Section B.8 [La fonte Emmentaler], page 935.

17.5.6 Mélismes kiéviens

Les notes formant un mélisme kiévien sont habituellement rapprochées les unes des autres, les mélismes étant espacés les uns des autres. Ceci permet au chantre d'identifier aisément les structures mélodiques d'un chant *Znamenny*. Les mélismes sont traites par LilyPond comme des ligatures dont l'espacement est géré par le `Kievan_ligature_engraver`.

Le `Kievan_ligature_engraver` est activé par défaut pour les contextes `KievanVoice` et `KievanStaff`. Pour les autres contextes, il s'active au sein d'un bloc `layout` dans lequel est désactivé le `Ligature_bracket_engraver`.

```
\layout {
  \context {
    \Voice
    \remove Ligature_bracket_engraver
    \consists Kievan_ligature_engraver
```
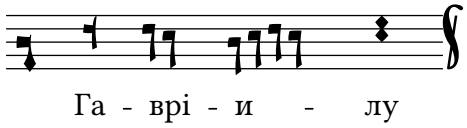
```
    }
  }
```

L'espacement des notes d'une ligature kiévienne se gère à l'aide de la propriété `padding` de `KievanLigature`.

Voici comment générer des ligatures en notation kiévienne :

```
% Font settings for Cyrillic
\paper {
  property-defaults.fonts.serif = "Linux Libertine O,serif"
}

\score {
  <<
    \new KievanVoice = "melody" \relative c' {
      e2 \[ e4( d4 ) \] \[ c4( d e  d ) \] e1 \fine
    }
    \new Lyrics \lyricsto "melody" {
      Га -- врі -- и -- лу
    }
  >>
}
```



Voir aussi

Glossaire musicologique : Section "ligature" dans *Glossaire*.

Manuel de notation : Section 17.2.2 [Ligatures], page 554, Section 17.3.10 [Ligatures mensurales], page 563, Section 17.4.7 [Neumes et ligatures grégoriennes], page 573.

Problèmes connus et avertissements

L'espacement des ligatures n'est pas des meilleurs.

17.6 Réédition de musique ancienne
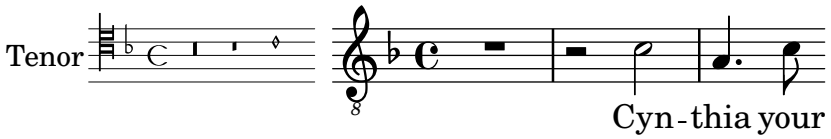
Travailler sur de la musique ancienne requiert bien souvent des tâches particulières et qui s'éloignent fortement de la notation moderne pour laquelle LilyPond est conçu. Nous allons aborder, au fil des paragraphes qui suivent, un certain nombre de cas particuliers et vous proposer des suggestions, voire des solutions aux problèmes que vous ne manquerez pas de rencontrer. Ceci inclut entre autres :

- comment réaliser un incipit, autrement dit un court extrait montrant ce à quoi ressemblait l'original, en introduction à la transcription d'une œuvre médiévale ;
- comment obtenir une présentation *Mensurstriche* comme on peut le voir dans nombre de transcriptions de musique polyphonique ;
- comment transcrire du grégorien en notation moderne ;
- comment obtenir à la fois une reproduction en notation ancienne et une édition en notation moderne à partir d'une même source.

17.6.1 Des incipits

Il est d'usage, lorsque l'on transcrit de la musique ancienne en notation moderne, d'indiquer aussi comment apparaissaient les silences ou notes initiaux dans la version originale, y compris la clef. Ceci s'appelle un *incipit*. La commande `\incipit` utilise le `indent` de la portée principale pour déterminer la place occupée par l'incipit accolé au nom d'instrument, et `incipit-width` pour déterminer la longueur de la portée d'incipit. Un argument optionnel permet d'indiquer de combien l'incipit devrait se décaler sur la gauche.

```
\score {
  \new Staff <<
    \new Voice = Tenor {
      \set Staff.instrumentName = "Tenor"
      \override Staff.InstrumentName.self-alignment-X = #RIGHT
      \incipit #1 { \clef "mensural-c4" \key f \major r\breve r1 c'1 }
      \clef "treble_8"
      \key f \major
      R1 r2 c'2 |
      a4. c'8
    }
    \new Lyrics \lyricsto Tenor { Cyn -- thia your }
  >>
  \layout
  {
    indent = 5\cm
    incipit-width = 3\cm
  }
}
```



Par défaut, LilyPond utilise un contexte 'MensuralStaff' pour rendre un incipit. Il est possible de recourir directement à d'autres contextes en libellant `\incipit \new` *typedecontexte* ... ; celui-ci sera alors englobé dans un 'MensuralStaff' uniquement s'il est susceptible de contenir le contexte en question.

Problèmes connus et avertissements

La propriété `instrumentName` doit se placer au sein de la musique de l'incipit à produire. Lorsqu'il n'y a pas de nom d'instrument, il faut cependant le définir avec `\set Staff.instrumentName = ""`.

Voir aussi

Morceaux choisis : Section "Notations anciennes" dans *Morceaux choisis*.

17.6.2 Mise en forme de la musique mensurale

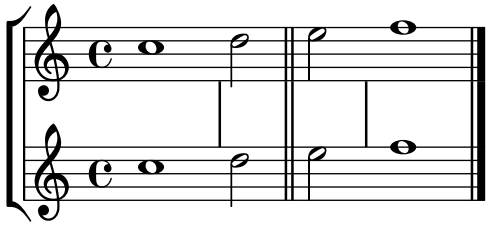
Mensurstriche, pour « lignes de mensuration », est le terme consacré lorsque les barres de mesure apparaissent uniquement entre les portées d'un système. Cette présentation permet de préserver l'aspect rythmique de l'original – par exemple sans couper une syncope par l'apparition d'une barre – tout en procurant l'aide que peuvent constituer les barres de mesure.

En musique mensurale, les barres de mesure ne traversent pas les portées. Pour obtenir ce résultat il faudra définir `measureBarType` à `"-span|"` et utiliser un regroupement de portées permettant l'extension des barres entre les portées, tel un `StaffGroup`.

```
\layout {
  \context {
    \Staff
    measureBarType = "-span|"
  }
}

music = \fixed c'' {
  c1
  d2 \section e2
  f1 \fine
}

\new StaffGroup <<
  \new Staff \music
  \new Staff \music
>>
```



17.6.3 Transcription de chant grégorien

Une transcription d'un chant grégorien en notation moderne s'obtient grâce à quelques simples artifices.

Hampes. Le contexte `GregorianTranscriptionVoice` ne crée pas de hampe. Ce comportement peut s'appliquer à d'autres en supprimant le graveur `Stem_engraver` du contexte de voix :

```
\layout {
  ...
  \context {
    \Voice
      \remove Stem_engraver
  }
}
```

Temps. En matière de chant non mesuré, plusieurs alternatives s'offrent à vous.

La suppression du `Time_signature_engraver` du contexte `Staff` ne produit aucun effet négatif. Une alternative serait de rendre la métrique transparente, ce qui par contre préservera l'espace qu'elle occupe.

Dans de nombreux cas, une clause `\set Score.timing = ##f` donne de bons résultats. On peut aussi utiliser `\cadenzaOn` et `\cadenzaOff`.

Les contextes de portée prédéfinis pour la musique ancienne ne créent pas de barre de mesure. On peut étendre ce comportement à tous les autres contextes à l'aide d'une clause

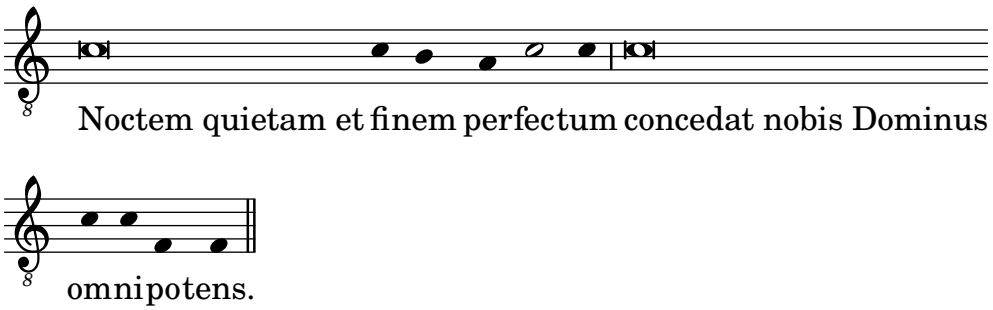
`\set Score.measureBarType = #'()` ou bien à une portée particulière grâce à un `\set Staff .measureBarType = #'()`.

Les contextes de portée prédéfinis pour la musique ancienne autorisent les sauts de lignes en l'absence de barre de mesure. On peut étendre ce comportement à tous les autres contextes à l'aide d'une clause `\set Score.forbidBreakBetweenBarLines = ##f` ou bien à des portées particulières grâce à un `\set Staff.forbidBreakBetweenBarLines = ##f`.

Dans de nombreuses transcriptions, le récitatif fait apparaître une brève au lieu de la répétition d'une même note. Le texte psalmodié se présente alors sous la forme d'une unique syllabe alignée à gauche :

```
chant = \relative {
  \clef "G_8"
  c'\breve c4 b4 a c2 c4 \divisioMaior
  c\breve c4 c f, f \finalis
}

words = \lyricmode {
  \tweak self-alignment-X #LEFT "Noctem quietam et"
    fi -- nem per -- fec -- tum
  \tweak self-alignment-X #LEFT "concedat nobis Dominus"
    om -- ni -- po -- tens.
}
\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "melody" {
      \chant
    }
    \new GregorianTranscriptionLyrics = "one" {
      \lyricsto "melody" \words
    }
  >>
}
```



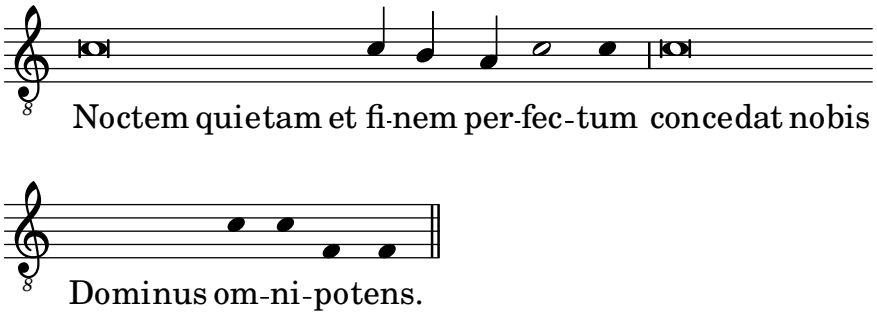
Ceci fonctionne bien tant que le texte ne risque pas de déborder de la ligne. Si tel était le cas, on pourrait plutôt ajouter des notes masquées comme ci-dessous.

Certaines transcriptions laissent néanmoins apparaître occasionnellement des hampes, notamment pour indiquer la transition entre un récitatif monodique et une phrase mélodique. Il suffit en pareil cas d'utiliser le `Stem_engraver` et, selon les besoins, recourir aux clauses `\omit Stem` et `\undo \omit Stem`.

```
chant = \relative {
  \clef "G_8"
  \omit Stem
  \omit Flag
  c'\breve*1/16 \hide NoteHead c8 c c c c
  \undo \hide NoteHead
  \undo \omit Stem \stemUp c4 b4 a
  \omit Stem c2 c4 \divisioMaior
  c\breve*1/16 \hide NoteHead c8 c c c c c c
  \undo \hide NoteHead c4 c f, f \finalis
}

verba = \lyricmode {
  No -- ctem qui -- e -- tam et fi -- nem per -- fec -- tum
  con -- ce -- dat no -- bis Do -- mi -- nus om -- ni -- po -- tens.
}

\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "melody" {
      \chant
    }
    \new GregorianTranscriptionLyrics = "one" {
      \lyricsto "melody" \verba
    }
  >>
  \layout {
    \context {
      \GregorianTranscriptionVoice
      \consists Stem_engraver
    }
  }
}
```

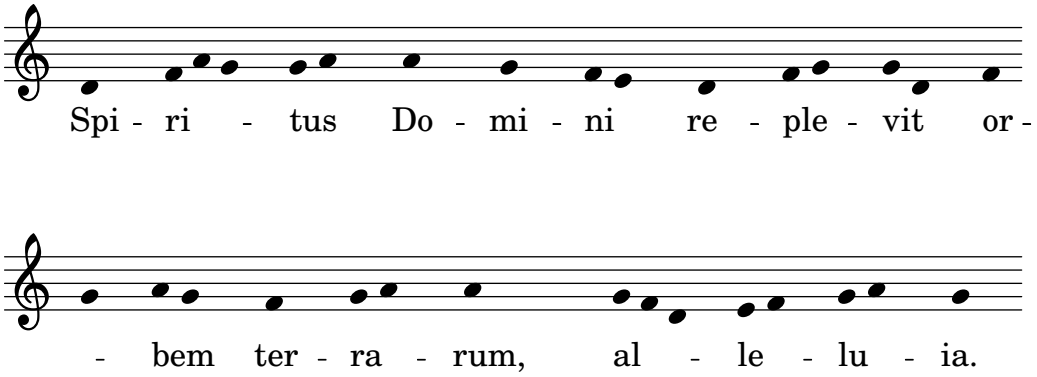


Autre situation courante, la transcription de chant neumatique contenant des mélismes, autrement dit, une psalmodie dans laquelle le nombre de syllabes varie selon les notes. Vous pourriez alors avoir envie d'indiquer clairement le découpage des groupes de syllabes ainsi que les subdivisions d'un mélisme. Le moyen pour y parvenir consiste à utiliser une métrique fixe, mettons `\time 1/4`, et de faire en sorte que chaque syllabe ou groupe de notes tienne dans une mesure, à l'aide de triolets ou de durées inférieures. Tant que les barres de mesure et autres éléments rythmiques restent transparents, et que l'espacement en regard des barres est accru, la représentation en notation moderne devrait être tout à fait satisfaisante.

Pour une répartition plus homogène de syllabes de longueur différente – telles que « -ri » et « -rum » – selon les groupes de note, une solution consiste à figer la propriété `X-extent` de l'objet `LyricText`. Ceci s'avère moins fastidieux que d'ajouter des syllabes sous forme de *markup*. Des ajustements supplémentaires peuvent se réaliser avec des « notes silencieuses » (`s`).

```
spiritus = \relative {
  \time 1/4
  d'4 \tuplet 3/2 { f8 a g } g a a4 g f8 e
  d4 f8 g g8 d f g a g f4 g8 a a4 s
  \tuplet 3/2 { g8 f d } e f g a g4
}

spirLyr = \lyricmode {
  \override Lyrics.LyricText.X-extent  = #'(0 . 3)
  Spi -- ri -- _ _ tus _ Do -- mi -- ni _
  re -- ple -- _ vit _ or -- _ bem _ ter -- ra -- _ rum,
  al -- _ _ le -- _ lu -- _ ia.
}
\score {
  \new GregorianTranscriptionStaff <<
    \new GregorianTranscriptionVoice = "chant" {
      \spiritus
    }
    \new GregorianTranscriptionLyrics = "one" {
      \lyricsto "chant" \spirLyr
    }
  >>
  \layout {
    \context {
      \GregorianTranscriptionStaff
      measureBarType = ""
      \override BarLine.X-extent = #'(-1 . 1)
      \hide TupletNumber
      \hide TupletBracket
    }
  }
}
```



17.6.4 Éditions ancienne et moderne à partir d'une même source

Recours aux balises pour produire une partition ancienne et moderne à partir de la même source

Grâce aux balises (*tags*), il est possible d'utiliser une même source pour produire une partition de musique mensurale et moderne. Dans cet exemple est créée la fonction `menrest` qui permettra de positionner les silence comme dans la version originale, tout en respectant leur position sur une portée standard.

Les balises permettent aussi de gérer d'autres différenciations selon les besoins, comme des « mesures de silence » (R1, R\breve, etc.) en notation moderne mais des silences normaux (r1, r\breve, etc.) en notation ancienne. L'action de convertir de la musique mensurale en version moderne est communément appelée « transcription ».

L'appel à `c4.\Be c8 c\Am` revient au même que `c4.[ c8 c]`. Néanmoins, il évite les avertissements si le début intervient sur une note dépourvue de ligature mais pourrait s'en voir affublée de par l'utilisation du `Completion_heads_engraver`.

[La ligne légèrement plus courte dans la portée mensurale permet d'éviter que le glyphe de custode ne soit raccourci lors de la génération des images. On peut l'éviter à l'aide d'un `\with-true-dimensions` comme ici.]

```
\layout {
  line-width = 150\mm
}

menrest = #(define-music-function (note) (ly:music?)
              #{
                \tag #'mens $(make-music 'RestEvent note)
                \tag #'mod $(make-music 'RestEvent note 'pitch '())
              #})
Be = \tag #'mod
       #(begin
          (ly:expect-warning (G_ "stem does not fit in beam"))
          (ly:expect-warning (G_ "beam was started here"))
          (make-span-event 'BeamEvent START))
Am = \tag #'mod ]

MenStyle = {
  \override Score.BarNumber.transparent = ##t
  \override Stem.neutral-direction = #up
  \omit Slur
  \omit Beam
}

finalis = \section

Music = \relative c'' {
  \key f \major
  g1 d'2 \menrest bes4 bes a2 \menrest r4 g4 fis4. fis8 fis4 fis \break
  g e f4.([ g8] a4[ g8 f] g2.\Be fis8 e\Am fis2) g\breve \finalis
}

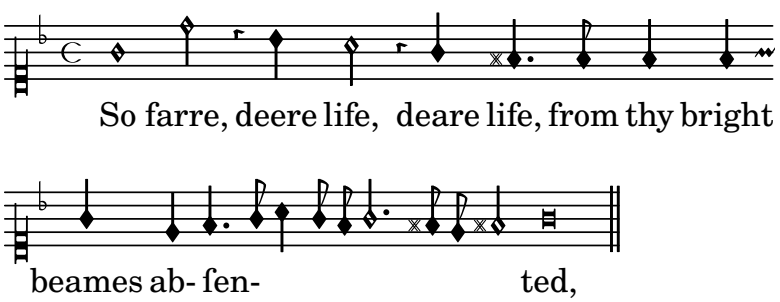
MenLyr = \lyricmode {
  So farre, deere life, deare life,
  from thy bright beames ab- en- ted,
}
ModLyr = \lyricmode {
```

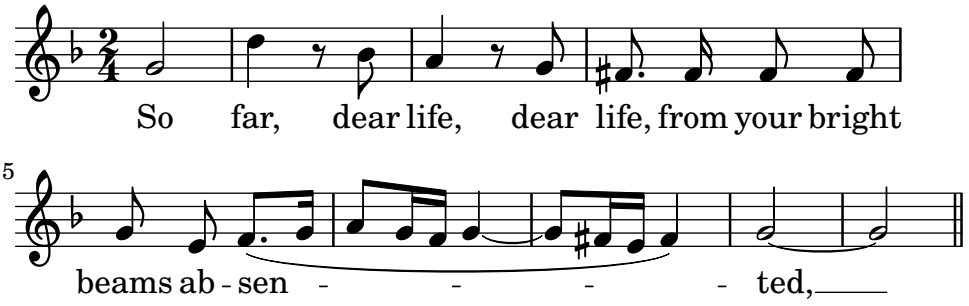
```
  So far, dear life, dear life,
  from your bright beams ab -- sen -- ted, __
}

\score {
  \keepWithTag #'mens {
    <<
      \new PetrucciStaff {
        \new PetrucciVoice = "Cantus" {
          \clef "petrucci-c1" \time 4/4 \MenStyle \Music
        }
      }
      \new Lyrics \lyricsto "Cantus" \MenLyr
    >>
  }
  \layout {
    \context {
      \PetrucciVoice
      % No longer necessary starting with version 2.25.23.
      \override Flag.style = #'mensural
    }
  }
}

\markup\vspace #1

\score {
  \keepWithTag #'mod {
    \new ChoirStaff <<
      \new Staff {
        \new Voice = "Sop" \with {
          \remove "Note_heads_engraver"
          \consists "Completion_heads_engraver"
          \remove "Rest_engraver"
          \consists "Completion_rest_engraver"
        } \shiftDurations 1 0 { \time 2/4 \autoBeamOff \Music }
      }
      \new Lyrics \lyricsto "Sop" \ModLyr
    >>
  }
}
```




So far, dear life, dear life, from your bright
5
beams ab - sen - - - - ted,

18 Musiques du monde

Ce chapitre a pour objet la notation des musiques traditionnelles autres qu'occidentales.

18.1 Noms des notes et altérations non-occidentaux

Nous allons voir ici comment saisir et imprimer des partitions dans d'autres formes que la musique occidentale, que les anglophones appellent aussi *Common practice period*.

18.1.1 Extension des systèmes de notation et d'accordage

Les formes de notation propres à la musique classique traditionnelle sont employées dans toutes sortes de musique autres que le « classique ». Nous en avons déjà parlé dans le chapitre Section 1.1 [Écriture des hauteurs de note], page 3, et plus particulièrement à la rubrique Section 1.1.5 [Nom des notes dans d'autres langues], page 11.

De nombreuses musiques autres qu'occidentales – et même certaines formes de musique traditionnelle occidentales – ont cependant recours à des systèmes de notation alternatifs ou étendus, qui ne s'intègrent pas forcément dans notre système standard.

Dans certains cas où la notation standard est utilisée, ces différences de hauteur seront implicites. Par exemple, la musique arabe est reproduite en notation standard et utilise des quarts de ton, l'altération réelle dépendant du contexte. Elle utilise traditionnellement la dénomination italienne, étendue dans le fichier `arabic.ly` par un certain nombre de macros – voir Section 18.2 [Musique arabe], page 591, pour plus de détails.

D'autres, par contre, font appel à une notation étendue, voire toute particulière. La *musique classique turque*, ou musique ottomane, utilise des formes mélodiques appelées *makamlar*, dans laquelle les tons sont divisés en neuf intervalles. Du point de vue actuel des pratiques de notation, il est possible d'utiliser les notes occidentales (do, ré, mi. . .) auxquelles on ajoutera l'altération spécifique à la musique turque. Ces différentes altérations sont définies dans le fichier `turkish-makam.ly`. Pour de plus amples informations, reportez-vous à la rubrique Section 18.3 [Musique classique turque], page 597.

Pour savoir où se trouvent les fichiers `hel-arabic.ly` et `makam.ly` sur votre système, reportez vous au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*.

Voir aussi

Glossaire musicologique : Section "Common Practice Period" dans *Glossaire*, Section "makamlar" dans *Glossaire*.

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section 1.1 [Écriture des hauteurs de note], page 3, Section 18.2 [Musique arabe], page 591, Section 18.4 [Musique classique perse], page 599, Section 18.3 [Musique classique turque], page 597, Section 1.1.5 [Nom des notes dans d'autres langues], page 11.

18.2 Musique arabe

Ce chapitre souligne les questions propres à la notation de la musique arabe.

18.2.1 Références pour la musique arabe

Jusqu'à nos jours, la musique arabe a principalement été transmise comme une tradition orale. Lorsqu'elle était transcrite, c'était en général sous forme de canevas sur lequel le rôle des interprètes était d'improviser substantiellement. La notation occidentale, cependant, est de plus en plus utilisée, avec quelques variations, pour transmettre et préserver la musique arabe.

Certains éléments de notation musicale occidentale, tels que les transcriptions d'accords ou de parties indépendantes, ne sont pas nécessaires pour retranscrire les pièces arabes les plus

traditionnelles. Il y a cependant quelques besoins spécifiques, tels que des intervalles se trouvant entre le demi-ton et le ton qui s'ajoutent aux intervalles mineurs ou majeurs utilisés dans la musique occidentale. Il est également nécessaire de regrouper et de noter un grand nombre de maqams (modes) différents qui font partie de la musique arabe.

En général, la notation de la musique arabe n'essaie pas d'indiquer précisément les micro-intervalles intervenant dans la pratique musicale.

Plusieurs particularités propres à la musique arabe sont traitées ailleurs :

- Les noms des notes et altérations (y compris les quarts de tons) peuvent être adaptés comme l'explique Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.
- Les armures peuvent également être adaptées comme expliqué dans Section 1.3.2 [Armure], page 25.
- Des métriques complexes peuvent nécessiter de grouper les notes manuellement, comme décrit dans Section 2.4.3 [Barres de ligature manuelles], page 117.
- Les *Takasim*, qui sont des improvisations rythmiquement libres, peuvent être écrites en omettant les barres de mesures, de la façon indiquée dans Section 2.3.5 [Musique sans métrique], page 94.

Voir aussi

Manuel de notation : Section 1.3.2 [Armure], page 25, Section 2.4.3 [Barres de ligature manuelles], page 117, Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.
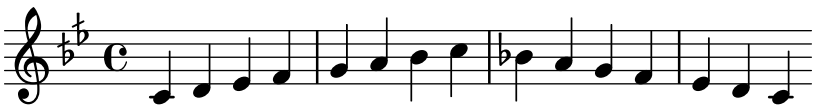
Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*.

18.2.2 Noms des notes en arabe

Les noms de note les plus traditionnels en arabe – tels que « rast », « dukah », « sikah », etc. – peuvent être très longs et ne conviennent pas à l'écriture de la musique.
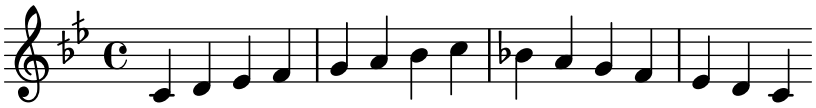
Les noms de note sont accessibles à travers le fichier `arabic.ly`. Voici comment écrire une gamme arabe *rast* :

```
\include "arabic.ly"
\relative {
  \key do \rast
  do' re misb fa | sol la sisb do | sib la sol fa | misb re do
}
```



Le fichier `arabic.ly` utilise les noms italiens, qui constituent le standard moderne des musiques arabes et sont largement utilisés par les musiciens. Il est toutefois possible de saisir la musique dans une autre langue dès lors que celle-ci est spécifiée après l'inclusion de `arabic.ly`. Par exemple, voici comment on peut écrire la gamme arabe *rast* :

```
\include "arabic.ly"
\language "english"
\relative {
  \key c \rast
  c' d eqf f | g a bqf c | bf a g f | eqf d c
}
```
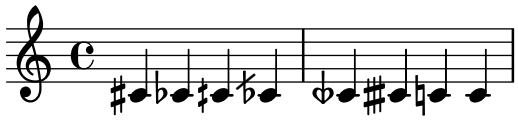


Le « rast » est une gamme heptatonique qui utilise des quarts de ton et est considéré comme la gamme centrale et la plus importante du « maqamat arabe ». Pour une liste exhaustive des gammes arabes disponibles, consultez Section 18.2.3 [Armures arabes], page 593.

L'utilisation des standards occidentaux pour noter la musique non occidentale est abordée dans Section 18.1 [Noms des notes et altérations non-occidentaux], page 591 ; consulter aussi Section 1.1.5 [Nom des notes dans d'autres langues], page 11,

Le symbole indiquant un demi-bémol ne correspond pas au symbole utilisé dans la notation arabe. Si le symbole particulier du demi-bémol arabe doit absolument être utilisé, il est possible de s'en approcher en faisant précéder la note par la commande `\dwn` définie dans le fichier `arabic.ly`. Cette méthode ne peut toutefois pas être utilisée pour modifier l'aspect du demi-bémol dans l'armure.

```
\include "arabic.ly"
\relative {
  \set Staff.extraNatural = ##f
  dod' dob dosd \dwn dob dobsb dodsd do do
}
```



Voir aussi

Manuel de notation : Section 18.2.3 [Armures arabes], page 593, Section 22.1 [Insertion de fichiers LilyPond], page 644, Section 1.1.5 [Nom des notes dans d'autres langues], page 11, Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

Fichiers d'initialisation : `ly/arabic.ly`.

Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*.

18.2.3 Armures arabes

Outre les armures mineures et majeures, les armures définies dans le fichier `arabic.ly` déterminent un certain nombre de groupes de maqams des plus courants.

En général, un maqam utilise l'armure de son groupe ou d'un groupe voisin et diverses altérations accidentelles sont indiquées tout au long de la musique. Par exemple, on trouve rarement le maqam *saba* ailleurs que dans un contexte de maqam *bayati*, tout en lui ajoutant une seule altération. Les maqams arabes autorisent peu de modulations en raison de la nature des instruments arabes.

Par exemple, pour indiquer l'armure d'une pièce en maqam muhayer :

```
\key re \bayati
```

Ici, *re* est la tonalité par défaut du maqam muayer, et *bayati* le nom du maqam de base dans le groupe.

Alors que l'armure correspond à un groupe, il est courant que le titre mentionne un maqam en particulier. Ainsi, dans cet exemple, le titre devrait faire apparaître le nom du maqam muhayer.

D'autres maqams du même groupe bayati, comme l'explique le tableau ci-dessous (*bayati*, *hussaini*, *saba,* et *ushaq*), peuvent être indiqués de la même manière. Ils sont autant de variations du maqam de base, le plus courant, du groupe (en l'occurrence, *bayati*). En général, c'est dans les tétracordes supérieurs que ces modes apparentés diffèrent, ou dans certains détails de disposition qui ne changent pas fondamentalement leur nature.

Certains maqams ne sont qu'une modulation de leur maqam de base. Ainsi, dans ce même groupe de *bayati*, du maqam *nawa*, dont la modulation est indiquée entre parenthèses dans le

tableau. Les maqams arabes n'admettent que des modulations limitées, en raison de la nature des instruments de musique arabes. Le *nawa* peut être indiqué comme suit :

```
\key sol \bayati
```

En musique arabe, le terme utilisé pour désigner un groupe maqam, tel que *bayati*, est également lui-même un maqam, généralement le plus important dans le groupe ; on peut le considérer comme un maqam de base.

Voici une suggestion de groupement qui relie les maqams les plus courants à leur armure :
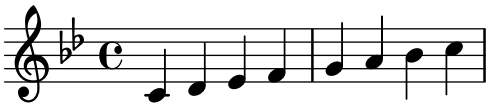
groupe maqam	**Armure**	**Tonique**	**Autres maqams dans le groupe (tonique)**
ajam	`major`	sib	jaharka (fa)
bayati	`bayati`	re	hussaini, muhayer, saba, ushaq, nawa (sol)
hijaz	`hijaz`	re	zanjaran (do)
hijaz kar	`hijaz_kar`	do	shahnaz, shad arban (sol)
huzam	`huzam`	misb	-
iraq	`iraq`	sisb	-
kurd	`kurd`	re	hijazkar kurd (do)
nahawand	`minor`	do	busalik (re), farah faza (sol)
nakriz	`nakriz`	do	nawa athar, hisar (re)
rast	`rast`	do	mahur, yakah (sol)
sikah	`sikah`	misb	-

Dans le cas où le maqam requis ne serait pas disponible, il est possible de le définir avant de l'utiliser. Voici, par exemple, comment définir et utiliser le maqam *zanjaran* :

```
\include "arabic.ly"

% For example on do: do reb mi fa sol la sib do
% reb and sib are FLAT
% You can also use SHARP, SEMI-FLAT, SEMI-SHARP
zanjaran = #`(
  (0 . ,NATURAL)
  (1 . ,FLAT)
  (2 . ,NATURAL)
  (3 . ,NATURAL)
  (4 . ,NATURAL)
  (5 . ,NATURAL)
  (6 . ,FLAT)
)

\relative {
  \key do \zanjaran
  do' reb mi fa sol la sib do
}
```



Dans certains cas particuliers, des maqams moins habituels sont définis dans le fichier `hel-arabic.ly` disponibles avec LilyPond.

Morceaux choisis

Armures inhabituelles

La commande `\key` détermine la propriété `keyAlterations` d'un contexte `Staff`.

Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété. Il s'agit en l'occurrence de définir une liste :

```
\set Staff.keyAlterations =
  #`(((octave . pas) . altération) ((octave . pas) . altération) ...)
```
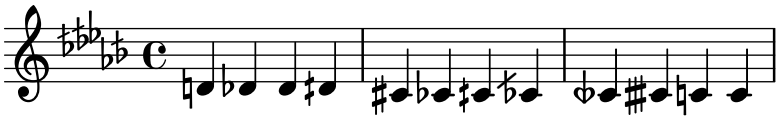
dans laquelle, et pour chaque élément, *octave* spécifie l'octave (0 pour celle allant du do médium au si supérieur), *pas* la note dans cette octave (0 pour do et 6 pour si), et *altération* sera `,SHARP` ou `,FLAT` ou `,DOUBLE-SHARP`, etc. (attention à la virgule en préfixe).

Une formulation abrégée – `(pas . altération)` – signifie que l'altération de l'élément en question sera valide quelle que soit l'octave.

En ce qui concerne les gammes microtonales dans lesquelles un « dièse » n'est pas d'un centième, *altération* se réfère à un deux-centième de ton entier.

```
\include "arabic.ly"

\relative do' {
  \set Staff.keyAlterations = #`((0 . ,SEMI-FLAT)
                                 (1 . ,SEMI-FLAT)
                                 (2 . ,FLAT)
                                 (5 . ,FLAT)
                                 (6 . ,SEMI-FLAT))
  % \set Staff.extraNatural = ##f
  re reb \dwn reb resd
  dod dob dosd \dwn dob |
  dobsb dodsd do do |
}
```



Voir aussi

Glossaire musicologique : Section “maqam” dans *Glossaire*, Section “bayati” dans *Glossaire*, Section “rast” dans *Glossaire*, Section “sikah” dans *Glossaire*, Section “iraq” dans *Glossaire*, Section “kurd” dans *Glossaire*.

Manuel d'initiation : Section “Hauteurs et armure” dans *Manuel d'initiation*.

Manuel de notation : Section 1.3.2 [Armure], page 25.

Fichiers d'initialisation : `ly/arabic.ly`, `ly/hel-arabic.ly`.

Référence des propriétés internes : Section “KeySignature” dans *Référence des propriétés internes*.

Morceaux choisis : Section “Musiques du monde” dans *Morceaux choisis*, Section “Hauteurs” dans *Morceaux choisis*.

18.2.4 Métriques arabes

Quelques formes de musique classique arabes et turques telles que *Semai* utilisent des métriques inhabituelles comme le 10/8. Ceci peut impliquer une manière de grouper les notes fort différente de la musique écrite existante, où les notes ne sont pas groupées par temps mais d'une façon

difficile à reproduire automatiquement. Il est possible d'y remédier en désactivant la ligature automatique et en groupant les notes manuellement. Lorsque l'enjeu n'est pas de reproduire exactement un texte existant, il est toujours possible d'ajuster le comportement de ligature automatique ou d'utiliser des chiffres de mesure complexes.

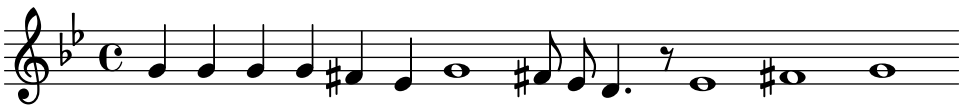
Morceaux choisis

Improvisation en musique arabe

Lorsque les improvisations ou *taqasim* sont temporairement libres, la métrique peut ne pas apparaître, auquel cas on utilisera un `\cadenzaOn`. Les altérations accidentelles devront alors être répétées en raison de l'absence de barre de mesure. Voici comment pourrait débuter une improvisation de *hijaz*.

```
\include "arabic.ly"

\relative sol' {
  \key re \kurd
  \accidentalStyle forget
  \cadenzaOn
  sol4 sol sol sol fad mib sol1 fad8 mib re4. r8 mib1 fad sol
}
```



Voir aussi

Glossaire musicologique : Section "semai" dans *Glossaire*, Section "taqasim" dans *Glossaire*.

Manuel de notation : Section 1.3.5 [Altérations accidentelles automatiques], page 32, Section 2.4.1 [Barres de ligature automatiques], page 104, Section 2.4.3 [Barres de ligature manuelles], page 117, Section 2.4.2 [Définition des règles de ligature automatique], page 107, Section 2.3.2 [Métrique], page 81, Section 2.3.5 [Musique sans métrique], page 94.

Fichiers d'initialisation : `ly/arabic.ly`.

Morceaux choisis : Section "Musiques du monde" dans *Morceaux choisis*.
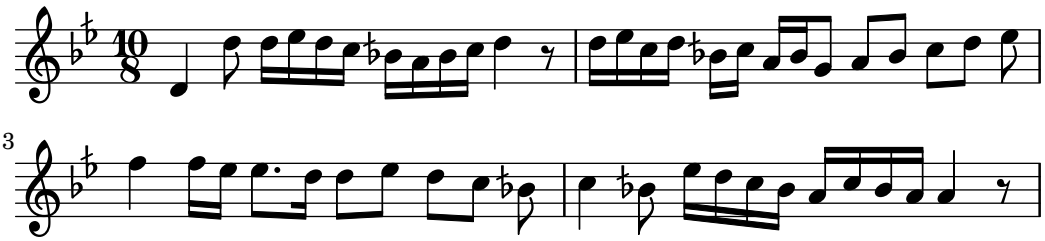
18.2.5 Exemple de musique arabe

Voici un modèle qui utilise également le début d'un *semai* turc courant dans l'éducation musicale arabe, pour illustrer quelques unes des particularités de la notation musicale arabe, comme des intervalles intermédiaires et des modes inhabituels traités dans ce chapitre.

```
\include "arabic.ly"
\score {
  \header {
    title = "Semai Muhayer"
    composer = "Jamil Bek"
  }
  \relative {
    \set Staff.extraNatural = ##f
    \set Staff.autoBeaming = ##f
    \key re \bayati
    \time 10/8

    re'4 re'8 re16 [misb re do] sisb [la sisb do] re4 r8
```

```
    re16 [misb do re] sisb [do] la [sisb sol8] la [sisb] do [re] misb
    fa4 fa16 [misb] misb8. [re16] re8 [misb] re  [do] sisb
    do4 sisb8 misb16 [re do sisb] la [do sisb la] la4 r8
  }
}
```



Voir aussi

Fichiers d'initialisation : `ly/arabic.ly`.

Morceaux choisis : Section "Musiques du mondes" dans *Morceaux choisis*.

18.2.6 Lectures complémentaires pour la musique arabe

Si tout le monde s'accorde à apparenter les maqams d'après leur tétracorde inférieur, parfois transposé, les méthodes de classification varient dans certains détails. Les sources ne sont pas entièrement cohérentes (parfois dans un même texte) quant à la manière d'indiquer l'armure de certains maqams. Il est courant, cependant, d'utiliser une armure par groupe plutôt qu'une armure différente pour chaque maqam.

- *La musique des Arabes* par Habib Hassan Touma [Amadeus Press, 1996], contient une étude des maqams et leur méthode de classification.
- Il existe également de nombreux sites web qui expliquent les maqams, dont quelques uns s'accompagnent d'exemples audio :
  - `https://www.maqamworld.com/`
  - `https://www.turath.org/`
- Des méthodes de luth arabe, l'oud, par les auteurs suivants, contiennent des exemples de compositions principalement turques et arabes,
  - Charbel Rouhana
  - George Farah
  - Ibrahim Ali Darwish Al-masri

18.3 Musique classique turque

Ce chapitre met en évidence des questions propres à la notation de la musique classique turque.

18.3.1 Références pour la musique classique turque

La musique classique turque s'est développée dans l'Empire Ottoman à peu près à la même période que la musique classique en Europe, et a continué jusqu'aux XXe et XXIe siècles comme une tradition vibrante et distincte avec sa propre théorie, ses propres formes et styles d'interprétation. Parmi ses caractéristiques remarquables, se trouve l'usage de micro-intervalles fondés sur des « commas » d'un neuvième de ton, dont sont dérivées les formes mélodiques *makam* (pluriel *makamlar*).

Quelques questions relatives à la musique classique turque sont traitées dans d'autres chapitres. Les noms de notes et altérations sont mentionnés dans Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

18.3.2 Noms de note en turc

La musique classique turque attribue traditionnellement un nom unique à chaque hauteur, et du fait de la division du ton en neuf parts, les makamlar emploient une échelle de hauteurs complètement différente des gammes et modes d'occident :
koma de 1/9 de ton entier, *eksik bakiye* (3/9), *bakiye* (4/9), *kücük mücenneb* (5/9), *büyük mücenneb* (8/9), *tanîni* (un ton entier) et *artık ikili* (12/9 ou 13/9 de ton).

D'un point de vue de notation moderne, il est pratique d'utiliser les positions occidentales des notes sur la portée (do, ré, mi. . .) avec des altérations spéciales qui haussent ou baissent les notes par intervalles de 1/9, 4/9, 5/9 et 8/9 de ton. Ces altérations sont définies dans le fichier `turkish-makam.ly`.

Pour plus d'information sur les formes de notation non-occidentales, reportez-vous au chapitre Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

Voir aussi

Glossaire musicologique : Section "makam" dans *Glossaire*, Section "makamlar" dans *Glossaire*.

Manuel de notation : Section 18.1 [Noms des notes et altérations non-occidentaux], page 591.

18.3.3 Armures turques

LilyPond prend en charge plus de 200 définitions d'armures de makam – nettement moins que ce qui peut être utilisé en musique classique turque –, chaque maqam disposant de sa propre tonique et hauteur finale (appelée « karar » en turc).

Il est important de garder à l'esprit la hauteur finale de chaque makam. Voici un exemple où *g* (sol) est la tonique par défaut et *rast* le nom du makam.

```
\key g \rast
```

Les altérations adéquates, koma bémol (*b1*) et koma dièse (`f4`), tous deux en relation avec la tonique *g*, seront affichés automatiquement.

Morceaux choisis

Exemple de makam turc

Ce canevas utilise le début d'un *saz semai* turc bien connu du répertoire aux fins d'illustrer certains éléments de la notation musicale turque.

```
#(set-default-paper-size "a6" 'landscape)

\include "turkish-makam.ly"
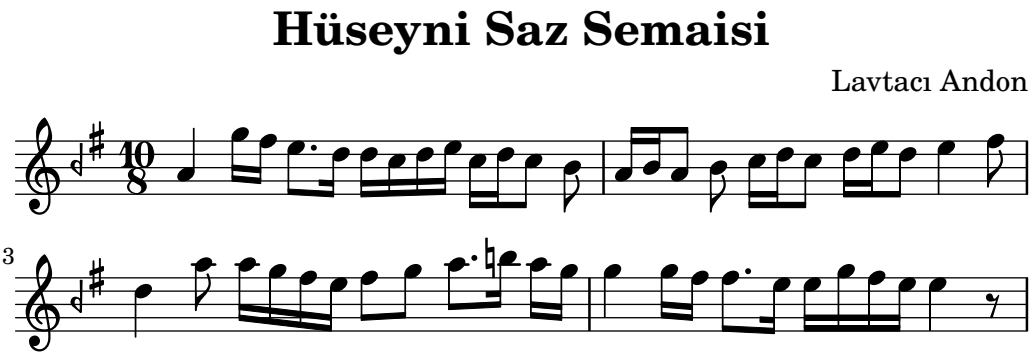
\header {
  title = "Hüseyni Saz Semaisi"
  composer = "Lavtacı Andon"
  tagline = ##f
}

\relative {
  \set Staff.extraNatural = ##f
  \set Staff.autoBeaming = ##f

  \key a \huseyni
  \time 10/8
```

```
  a'4 g'16[ fb] e8.[ d16] d[ c d e] c[ d c8] bfc |
  a16[ bfc a8] bfc c16[ d c8] d16[ e d8] e4 fb8 |
  d4 a'8 a16[ g fb e] fb8[ g] a8.[ b16] a16[ g] |
  g4 g16[ fb] fb8.[ e16] e[ g fb e] e4 r8 |
}

\layout {
  indent = 0
}
```



18.3.4 Lectures complémentaires pour la musique turque

- *Turk Musikisi Nazariyati ve Usulleri: Kudum Velveleleri* par Ismail Hakki Ozkan [(Kultur serisi, 41) (en turc) Paperback – 1986]

  contient des informations sur la théorie des makams et du usul.
- *Music of the Ottoman Court* par Walter Feldman [VWB Hardback – 1996]

  contient des informations sur l'histoire de la musique de cour ottomane.
- *Turkish Music Makam Guide* par Murat Aydemir [Pan Paperback – 2010]

  contient des informations en anglais sur les makams turcs et inclut deux disques compacts.

18.4 Musique classique perse

Ce chapitre met en évidence des questions propres à la notation de la musique classique perse.

18.4.1 Notation de la musique perse

La notation de musique classique perse utilise couramment deux altérations microtonales, *sori* et *koron*. Inventées par Ali-Naqi vers 1935, elles indiquent, respectivement à la hausse et à la baisse, un écart d'approximativement un quart de ton pour la hauteur considérée.



Le fichier `persian.ly`[1] fournit ce qu'il faut pour le koron et le sori. Ils s'obtiennent en ajoutant un « k » (koron) ou un « o » aux noms de note en anglais.

LilyPond prend en charge l'accordage de tous les modes perses majeurs dans toutes les tonalités nécessaires à la notation des *gushehs* (courts thèmes mélodiques) de tous les *dastgahs* (systèmes de musique modale).

[1] Il existe un autre fichier de prise en charge de la musique classique perse, plus ancien, créé par Kees van den Doel. Il n'est plus fonctionnel avec cette version de LilyPond ; bien que les noms de note y soient compatibles, la sélection des armures ne l'est plus.

La note qui suit immédiatement un koron est parfois[2] abaissée d'environ 20 centièmes. Bien que ce ne soit pas noté, ceci est considéré comme faisant partie de l'accordage. Néanmoins, et dans le but d'un meilleur rendu MIDI, on peut bémoliser un son en ajoutant un « v » au nom de la note (« *v*lat »). Cette note devrait par ailleurs être affublée d'un fort vibrato qui, lui aussi, sera intégré de manière perceptible (*sérialisme*).

Il n'existe pas d'autre particularité en matière d'accordage dans la musique perse. Dans la mesure où il s'agit de musique monophonique, la différence entre intonation juste (par exemple) et tempérament égal est purement académique – il n'y a pas d'accord où seraient notables des intervalles étrangers.

Voici les différents suffixes à porter aux nom des notes en anglais.

`ff`	double bémol
`f`	bémol
`k`	koron (environ quart de bémol, -3/10 de ton, 60 centièmes)
`o`	sori (environ quart de dièse, 2/10 de ton, 40 cents)
`s`	dièse
`x`	double dièse
`v`	20 centièmes de bémol (« vlat », non noté)
`fv`	bémol, tuned 20 cents down (notated as a normal flat)
`sv`	dièse, à 20 centièmes en dessous (noté comme un dièse normal ; on ne le trouve jamais en musique traditionelle iranienne)

18.4.2 Tonalités perses

Les dastgahs peuvent avoir plusieurs tonalités ; il est maintenant courant de nommer les tonalités selon la représentation des dastgahs qui les mettent en œuvre. Les tonalités perses qui suivent sont à fournir en second argument à la commande `\key`. Notez bien que, en pratique, ces tonalités ne sont pas toutes utilisées dans tous les tons.

`\shur`	gushehs « Shur » avec 5e degré naturel
`\shurk`	gushehs « Shur » avec 5e degré koron
`\esfahan`	
`\mokhalefsegah`	
`\chahargah`	
`\mahur`	identique à `\major` mais utilisé dans un contexte complètement different
`\delkashMahur`	

Dans un dastgah « Shur », le 5e degré peut être soit naturel, soit koron, raison pour laquelle deux tonalités sont disponibles (`\shur` et `\shurk`). Pour les dastgahs secondaires « Dashti », « Abuata » et « Bayat-e-tork » utilisent `\shur`, et le dasgah dérivé « Afshari » `\shurk`. « Nava » en sol est comme « Shur » en ré (à la quarte). La tonalité de « Segah » est `\shurk`. Les autres tonalités sont `\esfahan` (« Homayoun » en ut est comme « Esfahan » en sol, à la quinte), `\chahargah` et `\mokhalefsegah`. Les dastgahs « Mahur » et « Rast-panjgah » peuvent utiliser la même tonalité `\mahur`.
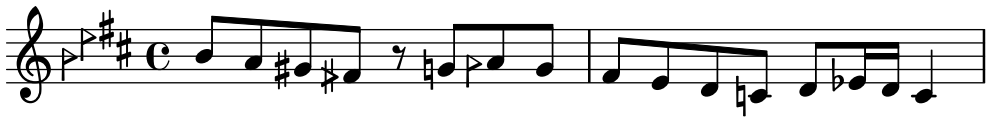
Toutes les différentes modulations de gushehs dans tous les dastaghs peuvent se noter avec ces tonalités, à l'exception du « Delkash » dans un « Mahur » mais, traditionnellement, les accidentelles particulières y sont alors notées explicitement (toutefois, `persian.ly` définit aussi la tonalité `\delkashMahur`).

À titre d'exemple, un Chahargah en ré peut se noter ainsi.

[2] Si l'intervalle entre la note qui précède le koron et celle qui le suit est d'une tierce mineure. Il en va de même pour la note en dessous de la finale du dastgah « Esfahan » selon certains (mais pas tous) musiciens iraniens.

```
\include "persian.ly"

\relative c' {
  \key d \chahargah
  bk'8 a gs fo r g ak g |
  fs ek d c d ef16 d c4 |
}
```
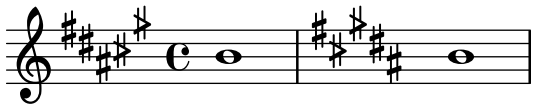


18.4.3 Armures perses

Par défaut, l'ordre des altérations dans une armure perse est la suivant : bémols suivis des korons, puis dièses suivis des soris (puis doubles bémols et enfin doubles dièses). L'ordre sera alors bémols ou korons (ou doubles bémols) puis dièses ou soris (ou doubles dièses). Autrement dit, korons et soris sont respectivement considérés comme bémols et dièses.

```
\include "persian.ly"

{
  \key b \chahargah b'1 |
  \set Staff.keyAlterationOrder = \persianAltKeyAlterationOrder
  \key b \chahargah b'1 |
}
```



18.4.4 Lectures complémentaires pour la musique perse

- *Traditional Persian Art Music* par Dariush Tala'i [Bibliotheca Persica, Costa Mesa CA, 2000]

  Les tonalités perses disponibles suivent de près cet ouvrage.
- *The Dastgah Concept in Persian Music* par Hormoz Farhat [Cambridge University Press, Cambridge, 1990]
- *Le répertoire-modèle de la musique iranienne* par Jean During [Sourush, Téhéran, 1995]

  Cet ouvrage contient la mesure des intervalles selon la pratique actuelle, par ailleurs compatible avec les tonalités de `persian.ly`.
- *Armoni-e Musiqi-e Iran* par Ali-Naqi Vaziri (en perse) [1935]
- *Scales and Modes Around the World* par Herman Rechberger [Fennica Gehrman, 2018, ISBN 978-952-5489-07-1]

Généralités en matière d'entrée et sortie

19 Modes de saisie

La manière dont sera interprétée la notation contenue dans un fichier source dépend du mode affecté à la saisie. Il existe, en règle générale, deux façons de spécifier le mode : une forme développée – par exemple `\chordmode` – et une forme abrégée – par exemple `\chords`. La forme développée s'utilise particulièrement lorsque la saisie fait l'objet d'une variable ou se trouve dans un contexte explicitement créé. La forme abrégée crée implicitement un contexte du type adéquate à la saisie et la lui transmet directement. Cette forme abrégée est tout à fait indiquée aux situations simples pour lesquelles nul n'est besoin de créer explicitement le contexte qui prendra en charge la saisie.

Mode accords

Ce mode, activé par la commande `\chordmode`, permet d'interpréter les saisies comme étant des accords, qui seront imprimés sous forme de notes sur une portée – voir Chapitre 15 [Notation des accords], page 522. La musique entrée en mode accords est rendue soit sous forme d'accords sur une portée pour un contexte `Staff`, soit sous forme de noms d'accord pour un contexte `ChordNames` ou sous forme de diagrammes pour un contexte `FretBoards`.

Le mode accords s'active aussi par la commande `\chords`, qui créera implicitement un nouveau contexte `ChordNames`. Le code saisi selon la syntaxe dévolue aux accords, sera interprété comme étant des accords nommés et sera alors rendu sous forme nominale dans ce contexte `ChordNames` – voir Section 15.2.1 [Impression des noms d'accord], page 527.

Mode percussions

Ce mode, activé par la commande `\drummode`, permet d'interpréter les saisies comme étant de la notation pour percussions – voir Section 13.1.2 [Notation de base pour percussions], page 500. Lorsqu'elle est entrée en mode percussions, la musique est rendue dans un contexte `DrumStaff`.

Le mode percussions s'active aussi par la commande `\drums`, qui créera implicitement un nouveau contexte `DrumStaff`. Le code saisi selon la syntaxe dévolue aux percussions, sera interprété comme étant de la notation pour percussions et alors rendu sous forme symbolique sur une portée de percussions – voir Section 13.1.2 [Notation de base pour percussions], page 500.

Mode figures

Ce mode, activé par la commande `\figuremode`, permet d'interpréter les saisies comme étant de la basse chiffrée (ou figurée) – voir Section 15.3.2 [Saisie de la basse chiffrée], page 541. Lorsqu'elle est entrée en mode figures, la musique est rendue sous forme de basse figurée dans un contexte `FiguredBass` ou dans un contexte `Staff`.

Le mode figures s'active aussi par la commande `\figures`, qui créera implicitement un nouveau contexte `FiguredBass`. Le code, saisi selon la syntaxe dévolue à la basse chiffrée, sera interprété comme étant des indication de basse chiffrée et sera alors rendu sous forme symbolique dans le contexte `FiguredBass` – voir Section 15.3.1 [Introduction à la basse chiffrée], page 540.

Modes frets et tablatures

Il n'existe pas de mode spécifique pour saisir des symboles de fret ou de tablature.

Notes ou accords saisis en mode note puis affectés à un contexte `TabStaff` seront rendus sous forme de diagramme de tablature – voir Section 12.1.3 [Tablatures par défaut], page 446.

Des diagrammes de fret viendront se positionner en surplomb d'une portée dès lors que les notes ou accords auront été saisis en mode note ou accord puis rendus dans un contexte `FretBoards` – voir Section 12.1.7 [Tablatures automatiques], page 490. Ils peuvent aussi se gérer sous forme de *markups* créés par la commande `\fret-diagram` – voir Section 12.1.5 [Tablatures sous forme d'étiquettes], page 469.

Mode paroles

Ce mode, activé par la commande `\lyricmode`, permet d'interpréter les saisies comme étant des syllabes, ayant éventuellement une durée, et des indications habituelles aux paroles – voir Chapitre 9 [Musique vocale], page 355. Lorsqu'il est entré en mode paroles, le texte est rendu sous forme de syllabes dans un contexte `Lyrics`.

Le mode paroles s'active aussi par la commande `\lyrics`, qui créera implicitement un nouveau contexte `Lyrics`. Le code saisi sera interprété comme étant des paroles et sera alors rendu sous forme de syllabes dans le contexte `Lyrics`.

Le mode paroles s'active aussi par la commande `\addlyrics`, qui créera un contexte `Lyrics` et ajoutera implicitement une commande `\lyricsto` afin d'associer les paroles qui suivent à la musique précédemment saisie – voir Section 9.1.4 [Durée automatique des syllabes], page 359.

Mode *markup*

Ce mode, activé par la commande `\markup`, permet d'interpréter les saisies comme étant des *markups* (annotations ou étiquettes) – voir Section A.1 [Commandes pour *markup*], page 837.

Mode notes

Le mode notes est le mode par défaut dans LilyPond. Il peut aussi s'activer par la commande `\notemode`. Les saisies seront interprétées comme étant des hauteurs, durées, *markups*, etc. qui seront rendues sous forme de notation musicale sur une portée.

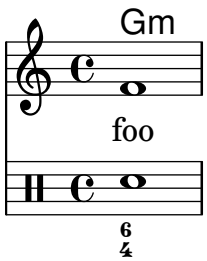
Nul n'est besoin de spécifier le mode notes de manière explicite, hormis dans certaines situations particulières, notamment lorsque vous êtes en mode paroles, accords, ou tout autre mode, et que vous deviez insérer un élément qui ne serait disponible que grâce à la syntaxe du mode notes.

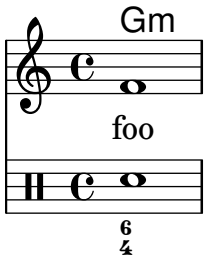
```
% Ceci...

<<
  \chords { g1:m }
  { f'1 }
  \lyrics { foo1 }
  \drums { sn1 }
  \figures { <6 4>1 }
>>

% ...est équivalent à

<<
  \new ChordNames \chordmode { g1:m }
  \new Voice \notemode { f'1 }
  \new Lyrics \lyricmode { foo1 }
  \new DrumStaff \drummode { sn1 }
  \new FiguredBass \figuremode { <6 4>1 }
>>
```




Gm
foo
6
4

20 Agencement du code

LilyPond traite des fichiers textuels. Ces fichiers portent par convention une extension `.ly`.

20.1 Structure d'une partition

Un bloc `\score` contient obligatoirement une seule expression musicale délimitée par des accolades[1] :
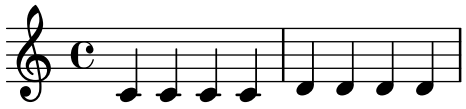
```
\score {
...
}
```

> **Note :** Il ne doit y avoir qu'**une seule** expression musicale globale dans un bloc `\score`, et elle **doit** être bornée par une paire d'accolades.

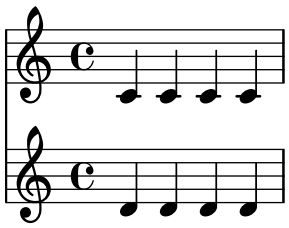
Cette unique expression musicale peut être de n'importe quelle taille et contenir d'autres expressions musicales aussi complexes soient elles. Voici quelques exemples d'expression musicale :

```
{ c'4 c' c' c' }
```

```
{
  { c'4 c' c' c' }
  { d'4 d' d' d' }
}
```



```
<<
  \new Staff { c'4 c' c' c' }
  \new Staff { d'4 d' d' d' }
>>
```



```
{
  \new GrandStaff <<
    \new StaffGroup <<
      \new Staff { \flute }
      \new Staff { \hautbois }
    >>
    \new StaffGroup <<
      \new Staff { \violonI }
      \new Staff { \violonII }
    >>
  >>
```

[1] Il existe aussi une commande de *markup* `\score` – voir [Scores within markup], page 888.

```
}
```

Les commentaires constituent l'une des rares exceptions à cette règle immuable – voir Section 20.5 [Structure de fichier], page 612, pour les autres. Qu'il s'agisse d'une seule ligne ou de tout un bloc – délimité par `%{ ... %}` – un commentaire peut se placer n'importe où dans le fichier source, aussi bien à l'intérieur qu'à l'extérieur du bloc `\score`, ou encore à l'intérieur ou à l'extérieur de l'expression musicale contenue dans un bloc `\score`.

Lorsqu'un fichier ne comprend qu'un bloc `\score`, celui-ci est implicitement inclus dans un bloc `\book`. Le bloc `\book` d'un fichier source permet la production d'au moins un fichier dont le nom sera, par défaut, déduit du fichier source : le traitement de `fandangopourelephants.ly` produira donc `fandangopourelephants.pdf`.

Pour de plus amples informations à propos du bloc `\book`, lisez Section 20.2 [Plusieurs partitions dans un même ouvrage], page 609, Section 20.3 [Plusieurs éditions pour une même source], page 611, et Section 20.5 [Structure de fichier], page 612.

Voir aussi

Manuel d'initiation : Section "La partition est une (unique) expression musicale composée" dans *Manuel d'initiation*, Section "Les expressions musicales en clair" dans *Manuel d'initiation*, Section "Travail sur les fichiers d'entrée" dans *Manuel d'initiation*.

20.2 Plusieurs partitions dans un même ouvrage

Un ouvrage peut se composer de plusieurs morceaux et de texte. C'est le cas des cahiers d'exercices ou d'une partie d'orchestre avec ses différents mouvements. Chaque mouvement fait l'objet d'un bloc `\score`,

```
\score {
  ...musique...
}
```

et le texte est contenu dans un bloc `\markup`,

```
\markup {
  ...texte...
}
```

Les différents mouvements et textes qui apparaissent dans un même fichier `.ly` ne composeront en principe qu'un seul fichier résultant.

```
\score {
  ...
}
\markup {
  ...
}
\score {
  ...
}
```

Attention cependant si vous travaillez avec `lilypond-book` : il vous faudra explicitement mentionner le bloc `\book`, en l'absence de quoi seul le premier `\score` ou `\markup` apparaîtra après traitement.

L'entête de chaque pièce peut se placer au sein du bloc `\score` ; le contenu du champ `piece` viendra s'imprimer avant chaque mouvement. De même, le titre de l'ouvrage peut se placer au sein du bloc `\book`. Dans le cas contraire, le contenu du bloc `\header` placé en début de fichier sera utilisé.

```
\header {
```

```
  title = "Huit miniatures"
  composer = "Igor Stravinsky"
}
\score {
  \header { piece = "Romance" }
  ...
}
\markup {
   ...texte du second couplet...
}
\markup {
   ...texte du troisième couplet...
}
\score {
  \header { piece = "Menuet" }
  ...
}
```

Plusieurs pièces seront regroupées dans un même « chapitre » à l'aide d'un bloc `\bookpart`. Ces différents « chapitres » sont séparés par un saut de page et peuvent comporter un titre à l'instar de l'ouvrage dès lors que vous y insérez un bloc `\header`.

```
\bookpart {
  \header {
    title = "Titre de l'ouvrage"
    subtitle = "Première partie"
  }
  \score { ... }
  ...
}
\bookpart {
  \header {
    subtitle = "Deuxième partie"
  }
  \score { ... }
  ...
}
```

De par leur conception, il n'est pas possible de définir des variables au sein d'un bloc `\book` ou `\bookpart` – c'est d'ailleurs aussi le cas pour les blocs `\score`. Ceci s'avère particulièrement important lorsque l'on veut répartir la musique dans de multiples fichiers à l'aide de variables « locales ». On peut toutefois adopter la structure suivante en pareil cas :

```
% mouvement1.ly
variableI = { ... }
bookpartI = \bookpart { \score { ... utilise \variableI ... } }

% mouvement2.ly
variableII = { ... }
bookpartII = \bookpart { \score { ... utilise \variableII ... } }

% principal.ly
\include "mouvement1.ly"
\include "mouvement2.ly"
\book {
```

```
  \bookpart { \bookpartI }
  \bookpart { \bookpartII }
}
```

De la même manière, un bloc `\layout` ne peut se trouver au sein d'un bloc `\book` ou `\bookpart`. Il doit se trouver au sein d'un bloc `\score` qui, lui, peut être inclus dans un bloc `\book` ou `\bookpart`.

20.3 Plusieurs éditions pour une même source

LilyPond produit un fichier par bloc `\book`. Lorsqu'aucun bloc `\book` n'est spécifié dans le fichier source, LilyPond considère que l'intégralité du fichier constitue un bloc `\book` unique, comme indiqué à la rubrique Section 20.5 [Structure de fichier], page 612.

Par défaut, LilyPond nomme le fichier produit à partir du nom du fichier source et, si nécessaire afin d'éviter les conflits, lui ajoute un suffixe – il s'agit en principe d'un pseudo numéro de version. Ainsi, le fichier `huitminiatures.ly` qui contiendrait

```
\book {
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
\book {
  \score { ... }
  \paper { ... }
}
```

générera

```
huitminiatures.pdf,
huitminiatures-1.pdf
huitminiatures-2.pdf.
```

20.4 Nom des fichiers de sortie

LilyPond vous permet de prendre le contrôle dans la dénomination des fichiers à générer, ainsi que de l'éventuel suffixe, à partir du bloc `\paper`.

```
\paper {
  output-filename = "mon_fichier_perso"
}

\book {
  \paper {
    output-suffix = "menuetto"
  }
  ...
}
\book {
  \paper {
    output-suffix = "scherzo"
  }
  ...
```

```
}
```

Ceci résultera en deux fichiers dénommés `mon_fichier_perso-menuetto.pdf` et `mon_fichier_perso-scherzo.pdf`. Une attention particulière doit être portée aux valeurs de `output-filename` et `output-suffix` qui doivent être valides pour votre système d'exploitation.

Dès lors que le nom du fichier existe déjà, ce qui peut arriver, par exemple, lorsque `output-filename` et `output-suffix` sont définis dans un bloc `\paper` global et no au sein de blocs `\book`, LilyPond lui adjoindra un suffixe numérique.

Notez bien que les variables prédéfinies de `\paper` (voir Section 26.1 [Le bloc \paper], page 691) doivent se placer avant les réglages de `output-filename` et `output-suffix`, comme ici :

```
bigMargin = \paper { top-margin = 10\cm }

\book {
  \paper {
    \bigMargin % doit intervenir en premier
    output-filename = "toto"
  }
}
```

20.5 Structure de fichier

Un fichier `.ly` peut contenir un certain nombre d'expressions de haut niveau. Les expressions de haut niveau sont les suivantes :

- Une définition de sortie, comme `\paper`, `\midi` et `\layout`. Ces définitions, lorsqu'elles se trouvent à un niveau supérieur, s'appliqueront à l'intégralité de l'ouvrage. Si l'une de ces expression apparaît à plusieurs reprises à un niveau supérieur, les différents contenus seront combinés, à ceci près qu'en cas de déclarations conflictuelles, la dernière aura préséance. Des informations complémentaires sont disponibles à la rubrique Section 27.1 [Le bloc \layout], page 705.
- Une expression Scheme pure, telle que `#(set-default-paper-size "a7" 'landscape)` ou `#(ly:set-option 'point-and-click #f)`.
- Un bloc `\header`, dont le contenu sera valide pour tout le fichier. Il comporte en général les valeurs par défaut des champs de titrage, tels le titre ou l'auteur entre autres, communs à tous les blocs `\book` inclus dans le fichier – voir Section 21.1.1 [Généralités en matière de titrages], page 615.
- Un bloc `\score` pour la partition. Cette partition sera assemblée avec les autres partitions se trouvant au même niveau pour composer le `\book`. Vous pouvez modifier ce comportement à l'aide de la variable `toplevel-score-handler` placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation `../scm/lily.scm`, et les réglages par défaut dans le fichier `../ly/declarations-init.ly`.
- Un bloc `\book` permet de regrouper naturellement plusieurs mouvements – autrement dit plusieurs blocs `\score` – dans un même document. Lorsqu'il y a plusieurs `\score`, LilyPond génère un seul fichier dans lequel les mouvements sont mis les uns à la suite des autres, ce pour chacun des blocs `\book` rencontrés. La seule raison qui peut vous demander d'expliciter plusieurs blocs `\book` dans un fichier `.ly` est lorsque vous avez besoin de générer différents documents à partir d'une même source. La présence explicite d'un bloc `\book` est aussi nécessaire lorsque vous travaillez sur un document `lilypond-book` qui reprendrait plusieurs `\score` ou `\markup` dans un même extrait. Vous pouvez modifier ce comportement à l'aide de la variable `toplevel-book-handler` placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation `../scm/lily.scm`.

- Un bloc \bookpart. Un ouvrage peut se découper en plusieurs parties à l'aide de blocs \bookpart, aussi bien pour alléger le travail de l'algorithme de calcul des sauts de page, que si les réglages du bloc \paper diffèrent d'une partie à l'autre.
- Une expression musicale telle que

```
{ c'4 d' e'2 }
```

Ce bout de code sera placé dans un \score et intégré à l'ouvrage en même temps que tous les autres \score ou expressions musicales. En d'autres termes, un fichier qui ne contiendrait que cette simple expression musicale sera traduit en

```
\book {
  \score {
    \new Staff {
      \new Voice {
        { c'4 d' e'2 }
      }
    }
    \layout { }
  }
  \paper { }
  \header { }
}
```

Vous pouvez modifier ce comportement à l'aide de la variable toplevel-music-handler placée en tête. Le gestionnaire par défaut est défini dans le fichier d'initialisation ../scm/lily.scm.

- Du texte sous forme de *markup* comme les paroles d'un couplet

```
\markup {
   2.  La première ligne du deuxième couplet.
}
```

De tels *markups* seront imprimés là où ils apparaissent, avant, après ou entre les expressions musicales.

- Une variable ou identifiant, ou bien une structure Scheme sous forme de liste associative imbriquée (alist).

```
toto = { c4 d e d }       % variable normale
"Cor 3" = { c4 d e d }    % variable entre guillemets
couplet.1 = { c4 d e d }  % liste associative imbriquée
```

Ces variables peuvent se réutiliser plus loin dans votre fichier en les faisant précéder d'une oblique inverse.

```
\toto
\"Cor 3"
\couplet.1
```

Le nom des identifiants ne doit pas comporter de chiffre (ASCII), de succession de caractères souligné (_) ou de tiret ni aucune espace. Tous les autres caractères Unicode sont permis, aussi bien latins, grecs, chinois que cyrilliques. Les souligné ou tiret isolés sont autorisés. Autrement dit, les variables CorIII ou Скрипка-II sont valides.

Toute combinaison de caractères est permise dès lors que le nom de la variable est borné par des guillemets informatiques. En pareil cas, les *antislashes* et guillemets doivent être « échappés » par un *antislash*. Sont donc valides : "toto tutu", "a-b-c" et "Cor 3".

Les listes associatives imbriquées permettent de contourner les limitation vues ci-dessus à propos du nommage des variables qui ne devraient comporter de nombre. Une liste associative imbriquée est un nom d'identifiant valide suivi d'un ou plusieurs jeux de point et clé.

Dans `mus.violon.1 = { a1 +}`, nous avons un identifiant appelé `mus` qui comporte une liste associative `violon` qui à son tour comporte une clé `1`, et auquel est finalement attribué la valeur `{ a1 }`. Pour `"1.2"."3.4".5` le nom de variable est `"1.2"` et les clés sont `"3.4"` et `5` – une telle construction, tout à fait valide, peut permettre de générer de manière automatique des noms de variable.

Aucun nom de variable ou de clé dans une liste associative ne doit correspondre à un nom de note. Si cela se révélait nécessaire, il faudrait l'entourer de guillemets informatiques ou, encore mieux, opter pour un autre nom ou une autre clé.

Pour des raisons techniques, accéder à des listes associatives imbriquées peut échouer dans certaines circonstances. En particulier, on ne peut y accéder aux niveaux de premier rang – autrement dit en dehors d'un bloc de musique. Si `lilypond` venait à à avorter avec un message d'erreur « mauvais type d'expression », il faudrait remplacer la liste associative imbriquée par une variable entre guillemets.

Voici trois éléments que vous pouvez placer à un niveau supérieur :

```
\layout {
  % pas en pleine largeur
  ragged-right = ##t
}

\header {
   title = "Do-re-mi"
}

{ c'4 d' e2 }
```

Vous pouvez placer, n'importe où dans votre fichier, les instructions suivantes :

- `\version`
- `\include`
- `\sourcefilename`
- `\sourcefileline`
- Une ligne de commentaire, introduite par le signe %.
- Un bloc de commentaire, délimité par `%{ ... %}`.

Vous pouvez insérer des espaces dans votre fichier source afin de lui apporter une meilleure lisibilité. Les espaces superflus sont normalement ignorés. Notez cependant qu'il est des cas où l'espace est requis pour éviter tout risque d'erreur :

- Autour d'une accolade, qu'elle soit ouvrante ou fermante ;
- Après chaque commande ou variable, autrement dit tout élément qui commence par un \ ;
- Après tout élément qui sera interprété comme une expression Scheme, autrement dit tout élément qui commence par un '#' ;
- Pour séparer les éléments d'une expression Scheme ;
- En mode parole – `lyricmode` – avant et après les commandes `\override` et `\set`.

Voir aussi

Manuel d'initiation : Section "Organisation des fichiers LilyPond" dans *Manuel d'initiation*.

Manuel de notation : Section 21.1.1 [Généralités en matière de titrages], page 615, Section 27.1 [Le bloc `\layout`], page 705.

21 Titres et entêtes

La plupart de la musique qui est éditée comporte un titre et le nom de son compositeur ; certains ouvrages dispensent beaucoup plus d'informations.

21.1 Création de titres et entête ou pied de page

21.1.1 Généralités en matière de titrages

Chaque bloc `\book` apparaissant dans un même fichier source résultera en un fichier indépendant, comme indiqué à la rubrique Section 20.5 [Structure de fichier], page 612. Chacun de ces fichiers résultants comporte trois endroits où placer des titrages : le **titrage de l'ouvrage** au début de chaque recueil (*book*), les **titrages de partie** au début de chaque partie (*bookpart*) et les **titrages de morceau** avant chaque pièce (*score*).

La valeur des champs de titrage `title` (le titre) et `composer` (le compositeur) se définissent dans des blocs `\header` – la syntaxe appropriée et la liste des différents champs disponibles par défaut sont à la section Section 21.1.2 [Mise en forme par défaut des titrages subalternes], page 618. Les titrages d'un ouvrage, de ses parties ou des morceaux qu'il contient peuvent tous comporter les même champs bien que, par défaut, le titrage d'un morceau se limite à `piece` et `opus`.

Les blocs `\header` peuvent se placer à quatre endroits différents qui formeront une hiérarchie descendante :

- En tête du fichier source, avant même tout bloc `\book`, `\bookpart` ou `\score` ;
- Au sein d'un bloc `\book` et en dehors de tout bloc `\bookpart` ou `\score` qu'il contient ;
- Au sein d'un bloc `\bookpart` et en dehors de tout bloc `\score` qu'il contient ;
- Au sein d'un bloc `\score`.

La valeur des différents champs sera filtrée en respectant cette hiérarchie ; les valeurs persisteront à moins d'être écrasées par une autre valeur à un niveau inférieur.

- Le titre d'un ouvrage découle des champs définis en tête de fichier source, modifiés par les champs définis au sein du bloc `\book`. Les champs résultants serviront à affecter un titre de recueil à l'ouvrage, si tant est que quoi que ce soit génère une page au début de cet ouvrage, avant la première partie – un simple saut de page forcé (`\pageBreak`) suffit.
- Le titre d'une partie découle des champs définis en tête du fichier source, modifiés par les champs définis au sein du bloc `\book` puis par ceux définis au sein du bloc `\bookpart`. Les valeurs qui en résulteront permettront d'imprimer les titrages de partie pour cette partie.
- Le titre d'un morceau découle des champs définis en tête du fichier source, modifiés par les champs définis au sein du bloc `\book` puis par ceux définis au sein du bloc `\bookpart`, et enfin par ceux définis au sein du bloc `\score`. Les valeurs qui en résulteront permettront d'imprimer les titrages de morceau pour ce morceau. Notez toutefois que, pour un morceau, seuls les champs `piece` et `opus` seront imprimés, à moins d'avoir valorisé à `#t` la variable `print-all-headers` dans la section `\paper`.

Nul n'est besoin de fournir un bloc `\header` à chacun des quatre niveaux ; on peut se passer aussi bien de l'un d'eux que de tous. Dans la même veine, un fichier source simpliste peut ne pas mentionner de bloc `\book` ou `\bookpart` qui seront alors créés implicitement.

Lorsque l'ouvrage ne comporte qu'un seul morceau, le bloc `\header` devrait prendre place en tête de fichier, de telle sorte que soit produit un titrage de partie qui met à disposition tous les champs de titrage.

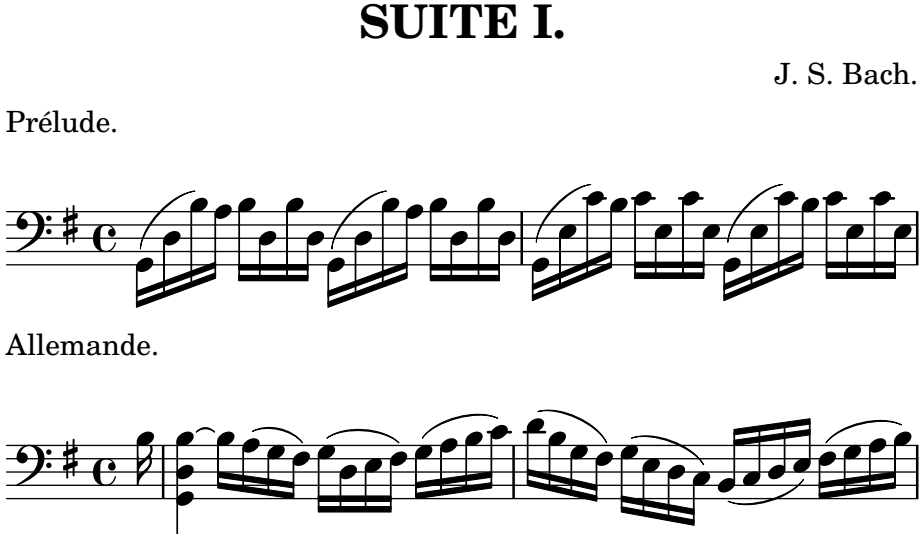
Lorsque l'ouvrage comporte plusieurs morceaux, différents arrangements du bloc `\header` permettent d'obtenir différents styles de publication musicale. Par exemple, si la publication

comprend plusieurs pièces du même compositeur, un bloc `\header` placé en tête de fichier définira le titre de l'ouvrage et le compositeur, que l'on complétera par un bloc `\header` dans chaque bloc `\score` pour définir les champs `piece` et `opus`, comme ici :

```
\header {
  title = "SUITE I."
  composer = "J. S. Bach."
}

\score {
  \header {
    piece = "Prélude."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \repeat unfold 2 { g,16( d' b') a b d, b' d, } |
    \repeat unfold 2 { g,16( e' c') b c e, c' e, } |
  }
}

\score {
  \header {
    piece = "Allemande."
  }
  \new Staff \relative {
    \clef bass
    \key g \major
    \partial 16 b16 |
    <g, d' b'~>4 b'16 a( g fis) g( d e fis) g( a b c) |
    d16( b g fis) g( e d c) b(c d e) fis( g a b) |
  }
}
```



Des agencements plus élaborés sont aussi réalisables. Par exemple, les champs appartenant au titrage principal d'un ouvrage peuvent se reporter dans chaque bloc `\score`, certains étant modifiés voire supprimés manuellement :

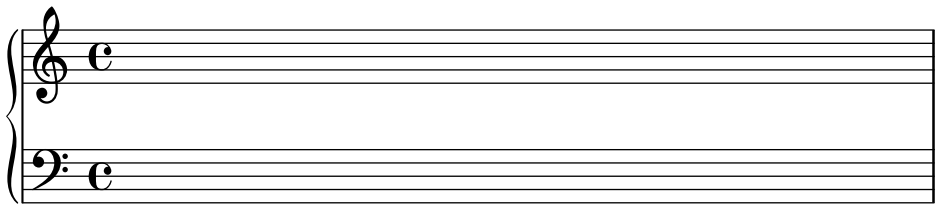
```
\book {
  \paper {
    print-all-headers = ##t
  }
  \header {
    title = "DAS WOHLTEMPERIRTE CLAVIER"
    subtitle = "TEIL I"
    % Pas de mention spéciale par défaut pour cet ouvrage
    tagline = ##f
  }
  \markup { \vspace #1 }
  \score {
    \header {
      title = "PRAELUDIUM I"
      opus = "BWV 846"
      % Pas de sous-titre pour ce morceau
      subtitle = ##f
    }
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
  \score {
    \header {
      title = "FUGA I"
      subsubtitle = "A 4 VOCI"
      opus = "BWV 846"
      % Pas de sous-titre pour ce morceau
      subtitle = ##f
    }
    \new PianoStaff <<
      \new Staff { s1 }
      \new Staff { \clef "bass" s1 }
    >>
  }
}
```

DAS WOHLTEMPERIRTE CLAVIER

TEIL I

PRAELUDIUM I

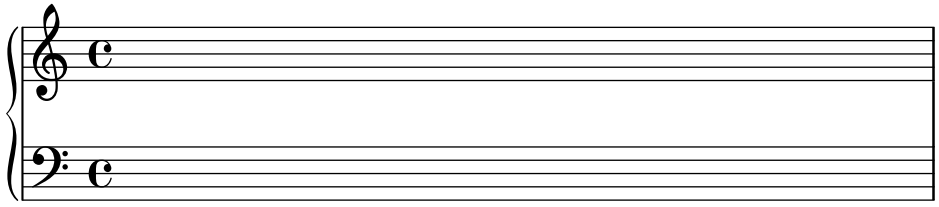
BWV 846



FUGA I

A 4 VOCI

BWV 846



Voir aussi

Manuel de notation : Section 21.1.2 [Mise en forme par défaut des titrages subalternes], page 618, Section 21.2.2 [Mise en forme personnalisée des titrages], page 624, Section 20.5 [Structure de fichier], page 612.

21.1.2 Mise en forme par défaut des titrages subalternes

L'exemple suivant recense les différentes variables imprimables attachées au bloc `\header`. Notez bien que l'espacement vertical par défaut entre les différentes composantes des entêtes est optimisé pour des entrées d'une seule ligne. Si l'un des éléments devait comprendre plusieurs lignes, par exemple un compositeur sur deux lignes, l'ajout d'un `\vspace` au champ en question sera peut-être nécessaire pour ajuster l'espacement vertical. Une alternative consiste à se définir sa propre mise en forme personnalisée – voir Section 21.2.2 [Mise en forme personnalisée des titrages], page 624.

```
\book {
  \header {
    % Les champs suivants sont centrés
    dedication = "Dédicace"
    title = "Titre"
    subtitle = "Sous-titre"
    subsubtitle = "Sous-sous-titre"
    % Les champs suivants sont répartis sur une même ligne, et
    % le champ "instrument" apparaîtra sur les pages suivantes
    instrument = \markup \with-color #green "Instrument"
    poet = "Librettiste"
```

```
    composer = "Compositeur"

    % Les champs suivants sont en opposition sur la même ligne
    meter = "Tempo"
    arranger = "Arrangeur"

    % Les champs suivants sont centrés en bas de page
    tagline = "Le « tagline » ou mention spéciale va en pied de dernière page"
    copyright = "Le copyright va en pied de première page"
  }
  \score {
    { s1 }
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 1"
      opus = "Opus 1"
    }
  }
  \score {
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 2 sur la même page"
      opus = "Opus 2"
    }
    { s1 }
  }
  \pageBreak
  \score {
    \header {
    % Les champs suivants sont en opposition sur la même ligne
      piece = "Pièce 3 sur une nouvelle page"
      opus = "Opus 3"
    }
    { s1 }
  }
}
```

Dédicace

Titre

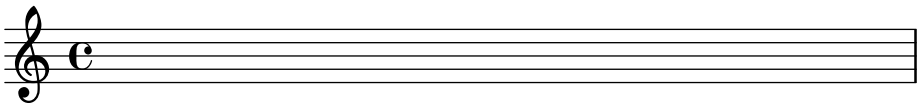
Sous-titre

Sous-sous-titre

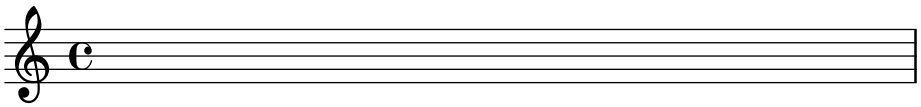
Librettiste **Instrument** Compositeur

Tempo Arrangeur

Pièce 1 Opus 1

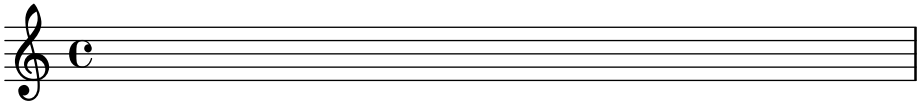


Pièce 2 sur la même page Opus 2



Le copyright va en pied de première page

Pièce 3 sur une nouvelle page Opus 3



Le « tagline » ou mention spéciale va en pied de dernière page

Quelques précisions :

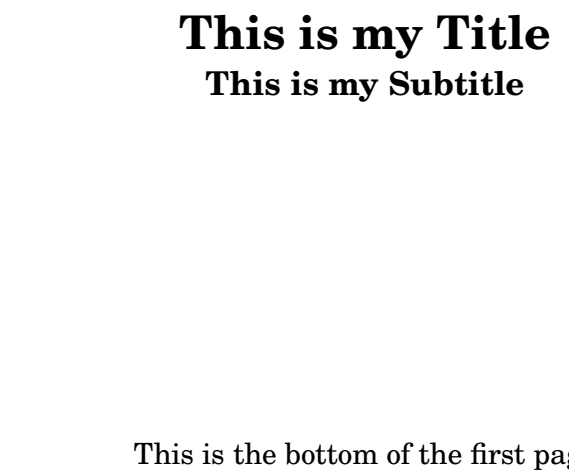
- Le nom de l'instrument sera répété en tête de chaque page.
- Seuls seront imprimés les champs `piece` et `opus` inclus dans un bloc `\score` dès lors que la variable `print-all-headers` reste désactivée (valeur à `#f`).
- Les champs d'un bloc `\header` qui n'auront pas été alimentés seront absents, de façon à ne pas gaspiller d'espace.
- Par défaut, `scoreTitleMarkup` place les champs `piece` et `opus` de part et d'autre sur une même ligne.

Les possibilités de modifier la mise en forme par défaut sont abordées à la rubrique Section 21.2.2 [Mise en forme personnalisée des titrages], page 624.

Un bloc `\book` qui commencerait directement par un bloc `\bookpart` ne verra pas ses titrages apparaître puisqu'il n'y a aucune page où imprimer le titre. Si toutefois le titre de l'ouvrage est requis, le bloc `\book` devra commencer par un *markup* ou une commande `\pageBreak`.
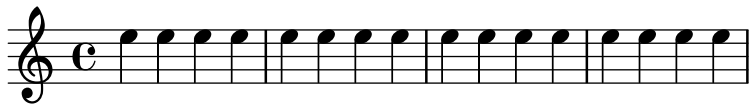
La variable `breakbefore` activée dans un bloc `\header` situé dans un bloc `\score` force le saut de page avant le morceau contenu dans ce `\score`. Vous pourrez ainsi séparer le titre principal de la musique.

```
\book {
  \header {
    title = "This is my Title"
    subtitle = "This is my Subtitle"
    copyright = "This is the bottom of the first page"
  }
  \score {
    \header {
      piece = "This is the Music"
      breakbefore = ##t
    }
    \repeat unfold 4 { e'' e'' e'' e'' }
  }
}
```



2

This is the Music



LilyPond v2.27.4

Voir aussi

Manuel d'initiation : Section "Organisation des fichiers LilyPond" dans *Manuel d'initiation*.

Manuel de notation : Section 21.2.2 [Mise en forme personnalisée des titrages], page 624, Section 20.5 [Structure de fichier], page 612.

Fichiers d'initialisation : `ly/titling-init.ly`.

21.1.3 Mise en forme par défaut des entête et pied de page

Les entête et pied – *header* et *footer* – sont des lignes de textes qui apparaissent en haut et en bas de chaque page, indépendamment du texte de l'ouvrage. Ils sont contrôlés par les variables suivantes, attachées au bloc `\paper` :
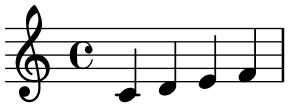
- `oddHeaderMarkup` – entête de page impaire
- `evenHeaderMarkup` – entête de page paire
- `oddFooterMarkup` – pied de page impaire
- `evenFooterMarkup` – pied de page paire

Ces variables *markup* sont définies dans le fichier `ly/titling-init.ly`, et de manière suivante par défaut :

- Les numéros sont placés en haut à gauche (si pair) ou à droite (si impair) de chaque page à compter de la deuxième.
- Le contenu du champ `instrument` est centré en haut de chaque page à compter de la deuxième.
- Le texte du `copyright` est centré au bas de la première page.
- Le contenu du champ `tagline` – mention spéciale – se place au bas de la dernière page, ou bien sous le `copyright` s'il n'y a qu'une seule page.

Le texte de la mention spéciale par défaut se modifie en alimentant le champ `tagline` au niveau du bloc `\header` principal.

```
\book {
  \header {
    tagline = "... la notation musicale pour Tous"
  }
  \score {
    \relative {
      c'4 d e f
    }
  }
}
```



... la notation musicale pour Tous

Pour supprimer le `tagline`, il suffit de lui assigner la valeur `#f`.

Voir aussi

Manuel de notation : Section 21.2.3 [Mise en forme personnalisée des entête et pied de page], page 627.

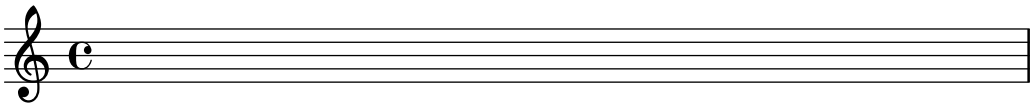
21.2 Titrages personnalisés

21.2.1 Mise en forme personnalisée des champs de titrage

Toutes les commandes de mise en forme d'un `\markup` permettent de personnaliser le texte des entête, pied de page ou éléments de titrage contenus dans un bloc `\header`.

```
\score {
  \header {
    piece = \markup { \fontsize #4 \bold "PRAELUDIUM I" }
    opus = \markup { \italic "BWV 846" }
  }
  { s1 }
}
```

PRAELUDIUM I *BWV 846*



Voir aussi

Manuel de notation : Section 8.2 [Mise en forme du texte], page 329.

21.2.2 Mise en forme personnalisée des titrages

L'utilisation de commandes `\markup` au sein d'un bloc `\header` permet de modifier aisément l'apparence du texte, mais n'influence en rien le positionnement précis des éléments de titrage. L'accès au positionnement des champs de titrage est géré par les deux variables suivantes, attachées au bloc `\paper` :

- `bookTitleMarkup`
- `scoreTitleMarkup`

Le positionnement des titres, avec les valeurs par défaut de ces variables `\markup`, est illustré à la rubrique Section 21.1.2 [Mise en forme par défaut des titrages subalternes], page 618.

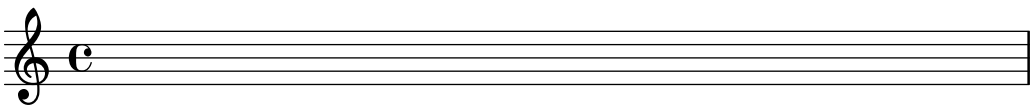
Voici les réglages par défaut de `scoreTitleMarkup`, tels que définis dans le fichier `ly/titling-init.ly` :

```
scoreTitleMarkup = \markup \column {
  \if \should-print-all-headers { \bookTitleMarkup \hspace #1 }
  \fill-line {
    \fromproperty #'header:piece
    \fromproperty #'header:opus
  }
}
```

Ceci aura donc pour effet de positionner les champs `piece` et `opus` sur la même ligne, en opposition :

```
\score {
  \header {
    piece = "PRAELUDIUM I"
    opus = "BWV 846"
  }
  { s1 }
}
```

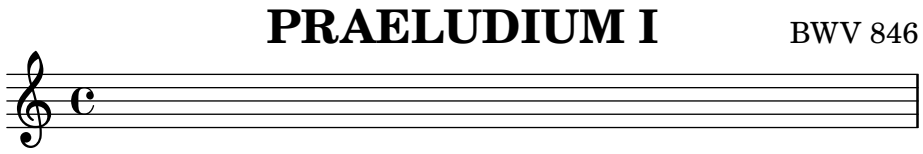
PRAELUDIUM I BWV 846



Voici comment redéfinir le `scoreTitleMarkup` de telle sorte que le champ `piece`, dont nous modifions la taille et la graisse, se place au centre de cette ligne :

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
```

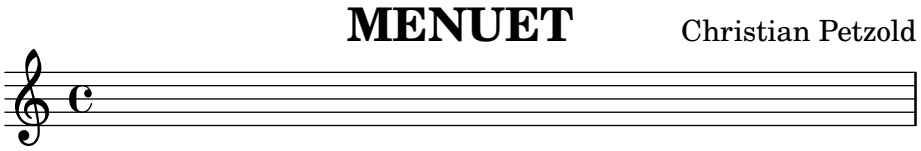
```
      piece = "PRAELUDIUM I"
      opus = "BWV 846"
    }
    { s1 }
  }
}
```
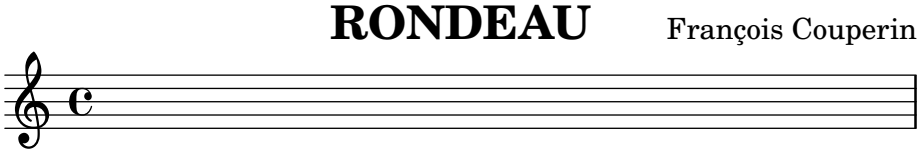


Les champs normalement absents du \header d'un bloc \score seront toutefois imprimés dès lors que vous aurez activé l'instruction print-all-headers au sein du bloc \paper. Le principal inconvénient de cette fonction réside dans le fait que les champs dévolus au titrage des parties devront être supprimés dans chacun des blocs \score de votre fichier source – voir Section 21.1.1 [Généralités en matière de titrages], page 615.

Afin d'éviter ce désagrément, ajoutez le champ que vous désirez voir apparaître à la définition de scoreTitleMarkup. Nous allons, dans l'exemple suivant, ajouter au scoreTitleMarkup le champ composer, normalement associé au bookTitleMarkup ; chaque \score pourra alors mentionner un compositeur différent.
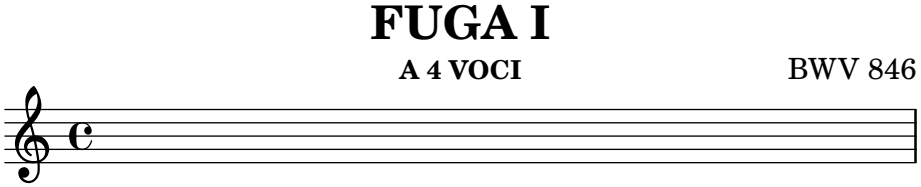
```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \fontsize #4 \bold \fromproperty #'header:piece
        \fromproperty #'header:composer
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "MENUET"
      composer = "Christian Petzold"
    }
    { s1 }
  }
  \score {
    \header {
      piece = "RONDEAU"
      composer = "François Couperin"
    }
    { s1 }
  }
}
```





Rien ne vous empêche de créer votre propre champ personnalisé, puis d'y faire référence dans la définition du *markup*.

```
\book {
  \paper {
    indent = 0\mm
    scoreTitleMarkup = \markup {
      \fill-line {
        \null
        \override #`(direction . ,UP)
        \dir-column {
          \center-align \fontsize #-1 \bold
            \fromproperty #'header:mycustomtext %% User-defined field
          \center-align \fontsize #4 \bold
            \fromproperty #'header:piece
        }
        \fromproperty #'header:opus
      }
    }
  }
  \header { tagline = ##f }
  \score {
    \header {
      piece = "FUGA I"
      mycustomtext = "A 4 VOCI" %% User-defined field
      opus = "BWV 846"
    }
    { s1 }
  }
}
```



Voir aussi

Manuel de notation : Section 21.1.1 [Généralités en matière de titrages], page 615.

21.2.3 Mise en forme personnalisée des entête et pied de page

L'utilisation de commandes `\markup` au sein d'un bloc `\header` permet de modifier aisément l'apparence du texte, mais n'influence en rien le positionnement précis des entête et pied de page. L'accès au positionnement des champs concernés est géré par les quatre variables suivantes, attachées au bloc `\paper` :

- `oddHeaderMarkup`
- `evenHeaderMarkup`
- `oddFooterMarkup`
- `evenFooterMarkup`

L'instruction `\if` au sein d'un `\markup` permet d'ajouter des éléments au texte des entêtes et pieds de page définis dans le bloc `\paper`, et ce uniquement lorsque certaines conditions sont vérifiées. En voici la syntaxe :

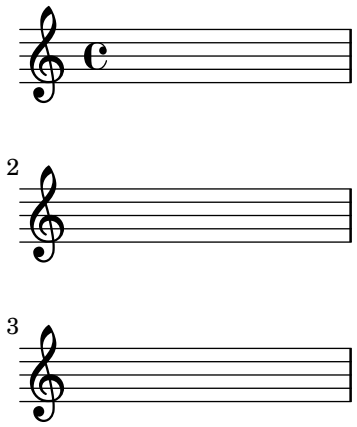
```
\if \condition argument
```

La *condition* est testée à chaque fois que le *markup* où elle apparaît est évalué. Le *markup argument* sera imprimé si et seulement si cette condition est remplie.

LilyPond dispose d'ores et déjà d'un certain nombre de conditions relatives à la numérotation des pages (la première, la dernière, une page spécifique, etc.). Le test renversant la condition s'obtient en remplaçant le `\if` par un `\unless`.

L'exemple suivant illustre la manière de centrer son numéro au bas de chaque page. Il nous faut tout d'abord annuler les définitions de `oddHeaderMarkup` et `evenHeaderMarkup` en les désactivant par un `#f`. Nous redéfinissons ensuite `oddFooterMarkup` pour qu'il contienne le numéro de page, centré. Enfin, nous appliquons le même paramétrage au `\evenFooterMarkup`.

```
\book {
  \paper {
    print-page-number = ##t
    print-first-page-number = ##t
    oddHeaderMarkup = ##f
    evenHeaderMarkup = ##f
    oddFooterMarkup = \markup {
      \fill-line {
        \if \should-print-page-number
          \fromproperty #'page:page-number-string
      }
    }
    evenFooterMarkup = \oddFooterMarkup
  }
  \score {
    \new Staff { s1 \break s1 \break s1 }
  }
}
```



1

Voici une liste des procédures prédéfinies utilisables avec `\if` ou `\unless`.

Syntaxe	**Condition testée**
`\on-first-page`	c'est la première page du *book*.
`\on-last-page`	c'est la dernière page du *book*.
`\on-first-page-of-part`	c'est la première page de la partie.
`\on-last-page-of-part`	c'est la dernière page de la partie.
`\on-page` *nombre*	ceci est la page *nombre*
`\single-page`	ce *book* tient sur une page.
`\should-print-page-numbers-global`	il faut imprimer les numéros de page.[1]
`\should-print-page-number`	il faut imprimer le numéro de cette page.
`\should-print-all-headers`	`print-all-headers` est vrai.

Voir aussi

Manuel de notation : Section A.1.5 [Conditional markup], page 892, Section 21.1.1 [Généralités en matière de titrages], page 615, Section 21.1.2 [Mise en forme par défaut des titrages subalternes], page 618.

fichiers d'initialisation : `../ly/titling-init.ly`.

21.3 Création des métadonnées des fichiers de sortie

En plus de s'imprimer sur la partition, les variables du bloc `\header` permettent de générer les métadonnées des fichiers de sortie. Dans le cas d'un fichier PDF, ces métadonnées pourront être affichées par le lecteur en tant que propriétés du document. Quel que soit le type de fichier de sortie, seules seront analysées les variables déterminées dans le `\header` du bloc définissant le fichier à générer, ainsi que celles des blocs hiérarchiquement supérieurs. Pour les fichiers PDF, seules les définitions du `\header` en dehors ou au niveau d'un bloc `\book` affecteront les métadonnées des documents PDF ; pour les fichiers MIDI seront utilisées les définitions jusqu'au niveau `\score`.

Par exemple, affecter « Symphony I » à la propriété `title` dans le bloc `\header` donnera aussi ce titre au document PDF et à la séquence MIDI.

```
\header {
```

[1] `\should-print-page-numbers-global` peut être différent de `\should-print-page-number` pour la première page de l'ouvrage, selon le réglage affecté à `print-first-page-number` dans le bloc `\paper`.

```
  title = "Symphony I"
}
```

Lorsque le titre imprimé diffère de celui affiché en tant que propriété du PDF, devra être renseignée la propriété `pdftitle`.

```
\header {
  title = "Symphony I"
  pdftitle = "Symphony I by Beethoven"
}
```

Les variables `title`, `subject`, `keywords`, `subtitle`, `composer`, `arranger`, `poet`, `author` et `copyright` initialisent toutes les propriétés PDF, qu'il suffit de préfixer d'un « pdf » pour affecter aux propriétés PDF une valeur divergente de la sortie imprimable.

La propriété PDF `Creator` prend automatiquement la valeur « LilyPond » additionnée du numéro de version ; les valeurs de `CreationDate` et `ModDate` sont définies à la date et l'heure courantes – `ModDate` peut être écrasé par la variable de `\header` `moddate` (ou `pdfmoddate`) pour un horodatage PDF valide.

La variable `title` détermine aussi le nom de la séquence MIDI. L'utilisation de la variable `miditítle` permet d'attribuer à la séquence MIDI un nom différent de celui attribué au fichier imprimable.

21.4 Notes de bas de page

Les notes de bas de page sont utiles dans bien des situations. Dans tous les cas, un « appel de note » vient se placer en référence dans le texte ou la musique, et le « texte de la note » est reporté au bas de la page, isolé de la musique par un trait horizontal. L'apparence de ce séparateur peut se modifier à l'aide de la variable de papier `footnote-separator-markup` – voir Section 26.6.4 [Variables supplémentaires d'en-tête et *markup*], page 702.

Selon qu'elle est référencée dans une expression musicale ou dans du texte indépendant, une note de bas de page sera créée suivant une procédure différente.

21.4.1 Notes de bas de page dans une expression musicale

Généralités sur l'annotation de musique

Il existe deux catégories d'annotation concernant une expression musicale :

Les annotations événementielles
: se rattachent à des événements particuliers, comme une note individuelle, un élément d'interprétation (doigté, accent ou nuance) ou des événements postérieurs (liaison, ligature manuelle). Une note de bas de page événementielle se libelle généralement sous la forme :

```
[position] \footnote [marque] décalage annotation musique
```

Les annotations temporelles
: se rapportent à un point particulier du déroulement d'un contexte musical. Certaines commandes, telles `\time` et `\clef`, ne reposent pas sur un événement pour la création de l'objet métrique ou clef. Il en va de même pour un accord : sa hampe ou ses crochets ne sont créés qu'à la fin d'un moment (plus exactement au travers de l'un des événements note qui le composent). Il n'est pas possible de connaître assurément lequel des événements note d'un accord est plus particulièrement à l'origine de la hampe ou du crochet. Il est donc plus aisé, pour de tels éléments, d'utiliser des annotations temporelles.

Une annotation temporelle permet d'annoter des objets de rendu sans se référer à un événement. Elle se libelle généralement sous la forme :

```
\footnote [marque] décalage annotation [Contexte.]nom-grob
```

Les arguments, quelle que soit la catégorie d'annotation, peuvent se définir ainsi :

position Lorsque la commande `\footnote` s'applique à un élément d'interprétation ou un événement rattaché, et uniquement dans ces cas, elle doit être précédée d'un indicateur de positionnement (‘`-`’, ‘`_`’ ou ‘`^`’) de façon à rattacher *musique* (avec sa marque) à la note ou au silence qui précède.

marque Un *markup* ou une chaîne de caractères identifiant l'annotation tant au niveau de l'appel que de la note qui apparaîtra au bas de la page. L'absence de cet élément – ou une valeur de `\default` – incrémentera automatiquement le compteur. Ce compteur est par défaut réinitialisé à chaque page comportant une annotation. La numérotation peut être continue sur l'ouvrage en désactivant la variable `reset-footnotes-on-new-page` – voir Section 26.6.4 [Variables supplémentaires d'en-tête et *markup*], page 702.

décalage Une paire de nombres – ‘`#'(2 . 1)`’ par exemple – spécifiant le décalage de la marque, en abscisse et en ordonnée, par rapport au point de référence. Des valeurs positives décalent vers la droite ou le haut, des valeurs négatives vers la gauche ou le bas ; des valeurs à zéro centrent la marque sur le point de référence. Le décalage s'exprime en espace de portée.

Contexte Le contexte auquel appartient l'objet à annoter. Cet argument peut être omis dès lors qu'il s'agit d'un contexte de bas niveau tel que `Voice`.

nom-grob Le type d'objet à annoter – ‘`Flag`’ par exemple. Lorsque cet élément est spécifié, c'est l'objet en question qui servira de point de référence, même s'il trouve son origine non pas directement dans une expression musicale mais dans tout objet du type spécifié intervenant à cet instant précis de la partition.

annotation
un *markup* ou une chaîne de caractères qui sera reporté au bas de la page.

musique l'élément qui fait l'objet du commentaire, qu'il s'agisse d'un événement musical, de l'un des constituants d'un accord ou d'un événement rattaché.

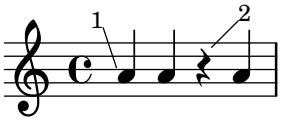
Notes de bas de page événementielles

Ce type de note de bas de page s'attache à un objet de rendu généré directement par l'événement correspondant à *musique*. Il répond à la syntaxe :

```
\footnote [marque] décalage annotation musique
```

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page événementielles"
  \markup \null
  \relative c'' {
    \footnote #'(-1 . 3) "Une note." a4
    a4
    \footnote #'(2 . 2) "Un silence." r4
    a4
  }
}
```

Notes de bas de page événementielles
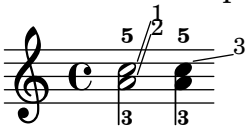


[1]Une note.
[2]Un silence.

Lorsqu'un accord fait l'objet d'une note de bas de page événementielle, chacune des hauteurs qui le composent se voit individuellement affublée de cette même annotation, ce qui n'est pas ce à quoi on pourrait s'attendre. Une des notes *au sein* de l'accord peut toutefois se voir attribuer une annotation :

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page événementielles"
  \markup \null
  \relative c'' {
    \footnote #'(1 . 3) "Un accord." <a-3 c-5>2
    <a-3 \footnote #'(3 . 0.5) "Une note dans un accord." c-5>4
  }
}
```

Notes de bas de page événementielles



[1]Un accord.
[2]Un accord.
[3]Une note dans un accord.

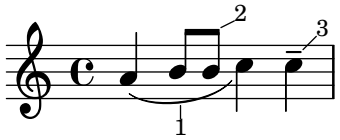
Lorsque l'annotation concerne un événement postérieur ou une articulation, la commande `\footnote` **doit** être précédée d'un indicateur de position ('-', '_' ou '^') et suivie de l'événement postérieur ou l'articulation comme argument *musique*. Dans ce cas, la commande `\footnote` peut se considérer comme une copie de son dernier argument auquel on attache une annotation. La syntaxe consacrée est :

position \footnote [*marque*] *décalage annotation musique*

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page événementielles"
  \markup \null
  \relative {
    a'4_\footnote #'(0 . -1) "Une liaison arbitrairement en dessous." (
    b8^\footnote #'(1 . 0.5) "Une ligature manuelle forcée en haut." [
    b8 ]
    c4 )
    c-\footnote #'(1 . 1) "Tenuto." --
  }
```

```
}
```

Notes de bas de page événementielles



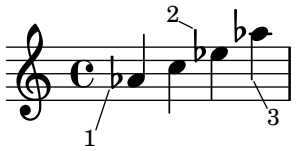
[1]Une liaison arbitrairement en dessous.
[2]Une ligature manuelle forcée en haut.
[3]Tenuto.

Notes de bas de page temporelles

Lorsque la note de bas de page se réfère à un objet de rendu résultant d'un événement – `Accidental` ou `Stem` découlent d'un `NoteHead` –, l'argument *nom-grob* de l'objet en question est requis après le texte de l'annotation, en lieu et place de *musique* :

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page temporelles"
  \markup \null
  \relative c'' {
    \footnote #'(-1 . -3) "Un bémol." Accidental
    aes4 c
    \footnote #'(-1 . 0.5) "Un autre bémol." Accidental
    ees
    \footnote #'(1 . -2) "Une hampe." Stem
    aes
  }
}
```

Notes de bas de page temporelles



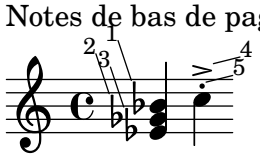
[1]Un bémol.
[2]Un autre bémol.
[3]Une hampe.

Notez bien que, lorsque *nom-grob* est spécifié, tous les objets de ce type qui se trouvent à ce même instant se verront attacher une annotation :

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page temporelles"
  \markup \null
  \relative c' {
    \footnote #'(-1 . 3) "Un bémol." Accidental
    <ees ges bes>4
    \footnote #'(2 . 0.5) "Une articulation." Script
    c'->-.
```

```
  }
}
```

Notes de bas de page temporelles
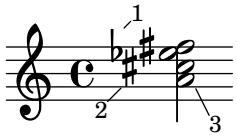


[1]Un bémol.
[2]Un bémol.
[3]Un bémol.
[4]Une articulation.
[5]Une articulation.

Une note incluse dans un accord peut individuellement se voir attribuer une annotation événementielle. Une tête de note (`NoteHead`) est le *seul* objet directement généré par un constituant d'accord ; elle peut donc être affectée d'une annotation événementielle. Tous les autres objets constituant un accord sont générés indirectement. La commande `\footnote` ne dispose pas d'une syntaxe permettant de spécifier *à la fois* un type d'objet *et* un événement particulier auquel s'attacher. De tels objets pourront toutefois faire l'objet d'une annotation temporelle, préfixée d'un `\single` afin d'annoter l'événement directement consécutif :

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page temporelles"
  \markup \null
  \relative c'' {
    < \footnote #'(1 . -2) "Un la." a
      \single \footnote #'(-1 . -1) "Un dièse." Accidental
      cis
      \single \footnote #'(0.5 . 0.5) "Un bémol." Accidental
      ees fis
    >2
  }
}
```

Notes de bas de page temporelles



[1]Un bémol.
[2]Un dièse.
[3]Un la.

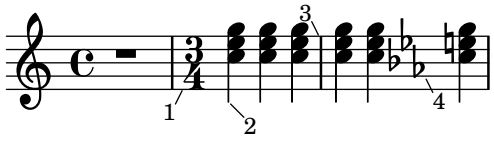
Note : Lorsque plusieurs notes de bas de page se rapportent à un même empilement vertical comme ci-dessus, elles sont numérotées et apparaîtront selon l'ordre vertical des éléments présentés, autrement dit celui positionné le plus haut en premier, non dans leur ordre d'apparition dans le fichier source.

Les objets de rendu tels que changement de clef ou d'armure tirent leur origine dans la modification d'une propriété plutôt que d'un véritable événement. D'autres, comme les barres ou numéros de mesure, dépendent directement de la temporisation. C'est la raison pour laquelle de tels objets doivent s'annoter en fonction de leur survenance au fil de la musique. Les notes de bas de page temporelles sont la solution à privilégier lorsqu'il s'agit d'annoter les hampes ou ligatures affectant des accords : bien qu'une telle fonctionnalité puisse s'appliquer à l'un des événements constituant l'accord, rien ne laisse présager lequel serait le plus approprié.

En matière de note de bas de page temporelle, l'objet de rendu considéré doit toujours être mentionné explicitement, ainsi que le contexte si l'objet est créé dans un autre contexte que celui du plus bas niveau.

```
\book {
  \header { tagline = ##f }
  \markup "Notes de bas de page temporelles"
  \markup \null
  \relative c'' {
    r1 |
    \footnote #'(-0.5 . -1) "Changement de métrique." Staff.TimeSignature
    \time 3/4
    \footnote #'(1 . -1) "Hampe de l'accord." Stem
    <c e g>4 q q
    \footnote #'(-0.5 . 1) "Barre de mesure." Staff.BarLine
    q q
    \footnote #'(0.5 . -1) "Changement d'armure." Staff.KeySignature
    \key c \minor
    q
  }
}
```

Notes de bas de page temporelles
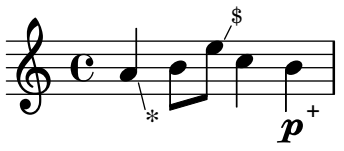


[1]Changement de métrique.
[2]Hampe de l'accord.
[3]Barre de mesure.
[4]Changement d'armure.

Les appels de note peuvent être personnalisés, et le trait reliant l'objet à l'appel supprimé :

```
\book {
  \header { tagline = ##f }
  \markup "Appels de note personnalisés"
  \markup \null
  \relative c' {
    \footnote "*" #'(0.5 . -2) \markup { \italic "* La première note." }
    a'4 b8
    \footnote \markup { \super "$" } #'(0.5 . 1)
      \markup { \super "$" \italic " La deuxième note." }
    e c4
    \once \override Score.Footnote.annotation-line = ##f
    b-\footnote \markup \tiny "+" #'(0.1 . 0.1)
```

```
      \markup { \super "+" \italic " Éditorial." } \p
  }
}
```

Appels de note personnalisés



* *La première note.*
$ *La deuxième note.*
+ *Éditorial.*

D'autres exemples de personnalisation des appels de note sont donnés à la rubrique Section 21.4.2 [Notes de bas de page dans du texte indépendant], page 635.

21.4.2 Notes de bas de page dans du texte indépendant

De telles notes de bas de page affectent les *markup* extérieurs aux expressions musicales. Il n'est pas nécessaire en pareil cas d'indiquer un point de référence par un trait ; l'appel de note vient juste s'accoler au *markup* qui fait l'objet de l'annotation. Les appels de note peuvent être gérés automatiquement, auquel cas ils seront numériques, ou bien manuellement en fournissant un indicateur particulier.

Les notes de bas de page concernant du texte indépendant se gèrent différemment selon qu'elles sont automatiques ou manuelles.

Notes de bas de page automatiques dans du texte

La syntaxe consacrée dans le cas d'une gestion automatique des appels de note est :

```
\markup { ... \auto-footnote texte annotation ... }
```

Ses éléments sont :
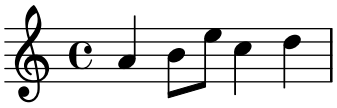
texte le *markup* ou la chaîne de caractères sur lequel porte l'annotation ;

annotation
un *markup* ou une chaîne de caractères constituant le texte de l'annotation qui sera reportée en bas de page.

Par exemple :

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple"
    \auto-footnote "tune" \italic " By me."
    "is shown below.  It is a"
    \auto-footnote "recent" \italic " Aug 2012."
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune[1] is shown below. It is a recent[2] composition.



[1] *By me.*
[2] *Aug 2012.*

Notes de bas de page personnalisées dans du texte

La syntaxe consacrée dans le cas d'une gestion personnalisée des appels de note est :

```
\markup { ... \footnote appel annotation ... }
```

Ses les éléments sont :
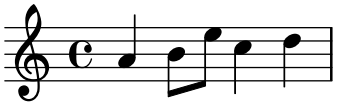
appel un *markup* ou une chaîne de caractères représentant l'appel de note affecté à ce point de référence. Notez bien que cette marque ne sera **pas** reproduite automatiquement avant le texte proprement dit de l'annotation.

annotation un *markup* ou une chaîne de caractères constituant le texte de l'annotation qui sera reportée en bas de page, précédé de l'*appel*.

N'importe quel caractère simple tel que '*' ou '+' peut s'utiliser en tant qu'appel de note, comme nous l'avons vu à la rubrique Section 21.4.1 [Notes de bas de page dans une expression musicale], page 629. D'autres caractères particuliers sont accessibles sous forme de raccourci – voir Section 22.4.3 [Équivalents ASCII], page 665 :

```
\book {
  \paper { #(include-special-characters) }
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote "*" \italic "* By me."
    "is shown below.  It is a recent"
    \footnote \super &dagger; \concat {
      \super &dagger; \italic " Aug 2012."
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune * is shown below. It is a recent † composition.
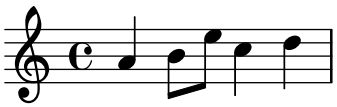


* *By me.*
† *Aug 2012.*

Un appel de note peut aussi se libeller sous la forme d'un point de code unicode – voir Section 22.4.2 [Unicode], page 664 :

```
\book {
  \header { tagline = ##f }
  \markup {
    "A simple tune"
    \footnote \super \char##x00a7 \concat {
      \super \char##x00a7 \italic " By me."
    }
    "is shown below.  It is a recent"
    \footnote \super \char##x00b6 \concat {
      \super \char##x00b6 \italic " Aug 2012."
    }
    "composition."
  }
  \relative {
    a'4 b8 e c4 d
  }
}
```

A simple tune § is shown below. It is a recent ¶ composition.



§ *By me.*
¶ *Aug 2012.*

Voir aussi

Manuel d'initiation : Section "Objets et interfaces" dans *Manuel d'initiation*.

Manuel de notation : Section 8.1.2 [Commentaires textuels], page 320, Section 22.4.3 [Équivalents ASCII], page 665, Section 8.1.5 [Indications textuelles], page 323, Section 7.2.2 [Info-bulle], page 309, Section B.12 [Liste des caractères spéciaux], page 943, Section 22.4.2 [Unicode], page 664.

Référence des propriétés internes : Section "Footnote" dans *Référence des propriétés internes*, Section "FootnoteEvent" dans *Référence des propriétés internes*, Section "Footnote_engraver" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les notes de bas de page ne peuvent que s'empiler l'une au-dessus de l'autre ; elles ne seront jamais présentées à la queue leu leu.

Les notes de bas de page peuvent générer des chevauchements quand elles sont trop nombreuses sur une même page.

21.5 Notes en ligne

Les notes en ligne fonctionnent comme les notes de bas de page en ceci qu'elles servent à annoter la musique, mais sont différents puisqu'elles apparaissent au-dessus ou au-dessous du système dans lequel se trouve l'objet auquel elles se rattachent.

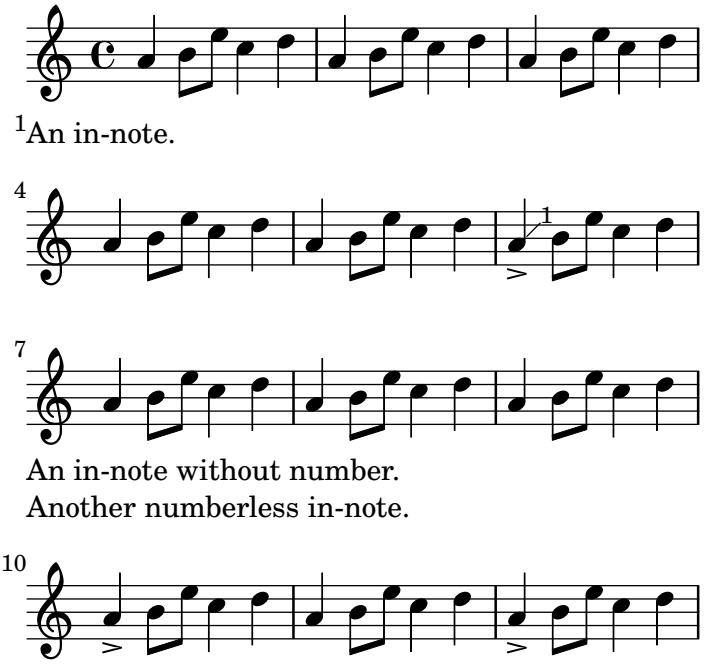
Avant de créer une note en ligne, la propriété `footnote` de l'objet `Footnote` doit être affectée d'un `#f`. La distance entre deux notes en ligne est gérée par la variable de papier `in-note-padding`, et l'écartement par rapport au système associé par la variable `in-note-system-padding`. Le positionnement des notes en ligne au-dessous du système concerné s'obtient en réglant la variable `in-note-direction` sur `DOWN`.

```
music = { a4 b8 e c4 d }

\book {
  \relative c'' {
    \override Score.Footnote.footnote = ##f

    \repeat unfold 5 \music
    \footnote #'(1 . 1) "An in-note." NoteHead
    <>-> \repeat unfold 4 \music
    \footnote "" #'(0 . 0) "An in-note without number." NoteHead
    <>-> \repeat unfold 2 \music
    \footnote "" #'(0 . 0) "Another numberless in-note." NoteHead
    <>-> \music
  }

  \paper {
    in-note-system-padding = 5
    in-note-padding = 2
    tagline = ##f
  }
}
```



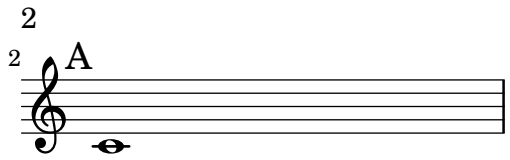
21.6 Référencement des numéros de page

LilyPond vous permet, à l'aide de la commande `\label`, d'insérer des points de référence dans un ouvrage, aussi bien en dehors qu'au fil de la musique. Ce point de référence pourra être ensuite repris à l'intérieur d'un *markup* ; vous pourrez même y ajouter le numéro de page grâce à la commande de *markup* `\page-ref`.

```
\header { tagline = ##f }
\book {
  \label #'firstScore
  \score {
    {
      c'1
      \pageBreak \mark A \label #'markA
      c'1
    }
  }

  \markup { Le premier mouvement débute à la page
            \page-ref #'firstScore "0" "?" }
  \markup { Le repère A est à la page \page-ref #'markA "0" "?" }
}
```





Le premier mouvement débute à la page 1
Le repère A est à la page 2

L'instruction `\page-ref` prend trois arguments :

1. le point de référence, sous la forme d'un symbole Scheme, comme par exemple `#'firstScore`,
2. un « emporte-pièce » afin d'estimer la longueur totale du *markup*, et
3. un texte de remplacement au cas où la référence ne serait pas retrouvée.

La présence de l'emporte-pièce est rendue nécessaire par le fait que les *markups* sont générés avant que les sauts de page ne soient positionnés. Bien que le numéro de page en question ne soit pas encore déterminé, LilyPond doit connaître les dimensions de ce *markup*. Vous pouvez, lorsque l'ouvrage contiendra plus de dix pages, stipuler un emporte-pièce sur deux caractères – soit `"00"`.

Lorsque la taille du texte final diffère de l'emporte-pièce, il peut s'avérer utile de configurer son alignement horizontal par rapport à l'espace réservé, à l'aide de la propriété `x-align`. Le réglage par défaut est un alignement sur la droite.

```
\markup {
  \box
    \page-ref #'foo "???" "?" " right-aligned (default)"
}
\markup {
  \box
    \override #`(x-align . ,LEFT)
    \page-ref #'foo "???" "?" " left-aligned"
}
\markup {
  \box
    \override #'(x-align . -2.5)
    \page-ref #'foo "???" "?" " left outside"
}
```

? right-aligned (default)

? left-aligned

? left outside

Dans cet exemple, l'emporte-pièce '`???`' est plus large que le texte de remplacement '`?`'. Le texte de remplacement est utilisé parce que le label `#'foo` n'existe pas.

La propriété `x-align` peut prendre n'importe quel nombre. Les symboles prédéfinis `LEFT`, `CENTER` et `RIGHT` permettent d'aligner respectivement sur la gauche, au centre, et sur la droite de l'emporte-pièce.

Commandes prédéfinies

`\label`, `\page-ref`.

21.7 Table des matières

La commande `\markuplist \table-of-contents` vous permettra de générer une table des matières. Les éléments qui la composeront sont créés par la commande `\tocItem`, insérée indépendamment ou au sein d'une expression musicale.

```
\markuplist \table-of-contents
\pageBreak

\tocItem \markup "Première partie"
\score {
  {
    c'4  % ...
    \tocItem \markup "Passage spécifique de la première partie"
    d'4  % ...
  }
}

\tocItem \markup "Seconde partie"
\score {
  {
    e'4 % ...
    \tocItem actI \markup "Acte I"
    f'4 % ...
    \tocItem actI.sceneI \markup "Scène 1"
    g'4 % ...
    \tocItem actI.sceneI.recitativo \markup "Récit."
    a'4 % ...
  }
}
```

Un libellé peut, facultativement, être associé à un élément particulier ou de façon hiérarchique dans une liste de libellés existants, terminant alors par le libellé de cet élément. Ce dernier cas permet de marquer l'élément comme « enfant » d'éléments précédemment libellés et ainsi laisser transparaître la structure de la partition dans la table des matières.

Les *markups* dévolus à la mise en forme de la table des matières se définissent dans le bloc `\paper`. LilyPond dispose de trois *markups* prédéfinis :

- `tocTitleMarkup`

  Utilisé pour mettre en forme le titre de la table des matières.

  ```
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table of Contents" \null }
    \null
  }
  ```

- `tocItemMarkup`

  Utilisé pour mettre en forme les éléments au sein de la table des matières.

  ```
  tocItemMarkup = \markup \fill-line {
    \fromproperty #'toc:text \fromproperty #'toc:page
  }
  ```

- `tocFormatMarkup`

  Utilisé pour mettre en forme les entrées de premier niveau dans la table des matières si tant est qu'existent plusieurs niveaux hiérarchiques. Il s'agit ici d'une procédure, comme abordé dans Section “Construction d'un markup en Scheme” dans *Extension de LilyPond*.

```
tocFormatMarkup = #make-bold-markup
```

- `tocIndentMarkup`

  Utilisé pour définir comment sera mise en évidence la hiérarchie. Ce *markup* sera imprimé zéro, une ou plusieurs fois selon le niveau de chacune des entrées.

  ```
  tocIndentMarkup = \markup \hspace #4
  ```

Toutes ces variables sont adaptables.

Voici comment, par exemple, franciser le titre :

```
\paper {
  tocTitleMarkup = \markup \huge \column {
    \fill-line { \null "Table des matières" \null }
    \hspace #1
  }
```

L'exemple suivant illustre la manière de modifier la taille des éléments de la table des matières :

```
tocItemMarkup = \markup \large \fill-line {
  \fromproperty #'toc:text \fromproperty #'toc:page
}
```

Notez bien la manière de référencer le libellé et le numéro de page dans la définition de `tocItemMarkup`.

Grâce à la commande `\tocItemWithDotsMarkup`, l'élément et son numéro de page peuvent se rejoindre par une ligne pointillée :

```
\header { tagline = ##f }
\paper {
  tocItemMarkup = \tocItemWithDotsMarkup
}

\book {
  \markuplist \table-of-contents
  \tocItem \markup { Allegro }
  \tocItem \markup { Largo }
  \markup \null
}
```

Table of Contents

Allegro 1
Largo 1

Au-delà de ces mécanismes de mise en forme, il est possible de définir d'autres commandes et *markups* afin de construire une table plus élaborée. Dans l'exemple qui suit, nous créons un nouveau style d'élément dans le but de mentionner les actes et scènes dans la table des matières d'un opéra :

Commençons par définir une nouvelle variable de type `markup` – appelée `tocActMarkup` – au sein du bloc `\paper`.

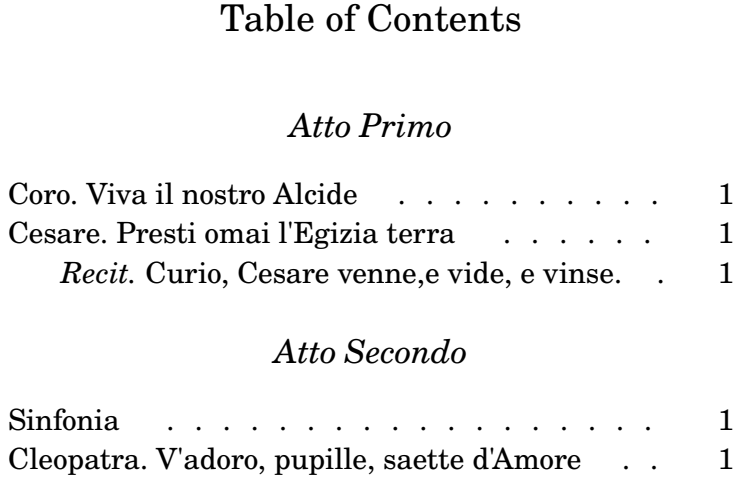
```
\paper {
  tocActMarkup = \markup \large \column {
    \hspace #1
    \fill-line { \null \italic \fromproperty #'toc:text \null }
    \hspace #1
```

```
  }
}
```

Créons ensuite une fonction musicale (`tocAct`) utilisant la nouvelle définition de *markup* `tocActMarkup`, tout en lui autorisant de définir un libellé pour chaque acte.

```
tocAct =
  #(define-music-function (label text) (symbol? markup?)
     (add-toc-item! 'tocActMarkup text label))
```

Dans un fichier LilyPond, l'utilisation de cette définition personnalisée, avec quelques adaptations aux réglages par défaut, pourrait ressembler à ceci :



Cet exemple illustre par ailleurs l'utilisation de la commande `\fill-with-pattern` dans le cadre d'une table des matières.

Commandes prédéfinies

`\table-of-contents`, `\tocItem`, `tocItemMarkup`, `tocTitleMarkup`, `tocFormatMarkup`, `tocIndentMarkup`.

Voir aussi

Fichiers d'initialisation : `../ly/toc-init.ly`.

22 Travail sur des fichiers texte

22.1 Insertion de fichiers LilyPond

Lorsqu'un projet prend de l'importance en volume, il est judicieux de le scinder en plusieurs fichiers, auxquels vous ferez référence avec un simple

```
\include "autrefichier.ly"
```

Une ligne `\include "autrefichier.ly"` dans un fichier revient à recopier intégralement le contenu de `autrefichier.ly` à l'endroit même ou est placée l'instruction `\include`. Vous pouvez par exemple écrire un fichier individuel par instrument, puis les regrouper pour former le fichier « conducteur ». Les différentes variables définies dans les fichiers séparés seront normalement reprises et utilisables dans le fichier formant le conducteur. Les sections balisées dans les fichiers individuels peuvent être réutilisées en différents endroits de la partition, comme expliqué à la rubrique Section 22.2 [Différentes éditions à partir d'une même source], page 645.

Lorsque le fichier auquel il est fait référence se trouve dans le même répertoire, donner seulement son nom en argument à la commande `\include` suffit. S'il se trouve ailleurs, vous devrez indiquer le chemin d'accès, absolu ou relatif, en respectant toutefois la syntaxe UNIX – autrement dit, le séparateur de répertoire est une oblique normale `/` et non l'oblique inverse `\` de DOS ou Windows. Par exemple, si le fichier `truc.ly` se trouve dans le répertoire supérieur au répertoire de travail, la ligne devra être

```
\include "../truc.ly"
```

ou bien, si les fichiers correspondant aux parties d'orchestre se trouvent dans le sous-répertoire `parties` relativement au répertoire courant, vous devrez mentionner

```
\include "parties/VI.ly"
\include "parties/VII.ly"
etc.
```

Les fichiers à inclure peuvent eux-mêmes contenir des instructions `\include`. Ces instructions `\include` de second niveau seront, par défaut, interprétées par rapport à leur situation dans l'arborescence. Tel sera, par exemple, le cas d'une bibliothèque générale « libA » créée pour utiliser des sous-fichiers à l'aide d'inclusions dans un fichier catalogue, comme ici :

```
libA/
  libA.ly
  A1.ly
  A2.ly
  ...
```

puis le fichier catalogue, `libA.ly`, qui contient

```
\include "A1.ly"
\include "A2.ly"
...
```

Tout fichier `.ly` peut désormais consulter l'intégralité de cette bibliothèque grâce à un simple

```
\include "~/libA/libA.ly"
```

Vous pouvez toutefois influer sur ce comportement de manière globale à l'aide de l'option `-drelative-includes=#f` en ligne de commande ou en ajoutant une clause `#(ly:set-option 'relative-includes #f)` en tête du fichier principal. En pareil cas, le chemin à suivre pour chacune des commandes `\include` sera pris relativement au fichier principal. Selon l'endroit où `relative-includes` est valorisé à `#t` ou `#f`, la commande `\include` permettra d'incorporer des fichiers contenus dans l'arborescence du répertoire principal et des fichiers situés ailleurs.

Vous pouvez inclure des fichiers dont vous spécifierez le chemin d'accès sur la ligne de commande au moment de lancer la compilation. L'appel à ces fichiers ne mentionnera alors que leur nom. Par exemple, si vous voulez compiler avec cette méthode le fichier `principal.ly` qui inclut des fichiers situés dans le sous-répertoire `parties`, placez vous dans le répertoire contenant `principal.ly`, puis tapez

```
lilypond --include=parties principal.ly
```

tout en ayant bien dans `principal.ly`

```
\include "VI.ly"
\include "VII.ly"
 etc.
```

Lorsqu'un fichier est voué à être inclus dans nombre de partitions, vous pouvez le placer dans le répertoire de LilyPond `../ly`. Attention : ce répertoire varie selon votre installation, comme indiqué au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*. Ce fichier sera inclus dès lors que vous fournirez uniquement son nom en argument à la fonction `\include`. C'est par exemple le cas du fichier de définition particulier `gregorian.ly`.

Au moment où vous lancez LilyPond, un certain nombre de fichiers se retrouvent inclus par défaut ; il suffit d'activer le mode verbeux en faisant `lilypond --verbose` pour s'en rendre compte. Vous verrez ainsi défiler, en plus de nombreuses informations, le nom d'un certain nombre de fichiers et de chemins d'accès. Les fichiers les plus important sont mentionnés au chapitre Section "Autres sources de documentation" dans *Manuel d'initiation*. Si vous venez à les modifier, rappelez-vous qu'ils seront écrasés à l'installation d'une nouvelle version de LilyPond.

Vous trouverez quelques exemples simples d'utilisation de la commande `\include` au chapitre Section "Conducteurs et parties" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "Conducteurs et parties" dans *Manuel d'initiation*.

Problèmes connus et avertissements

Lorsque vous incluez un fichier qui porte le même nom que l'un des fichiers d'initialisation de LilyPond, le fichier de la distribution de LilyPond aura préséance.

22.2 Différentes éditions à partir d'une même source

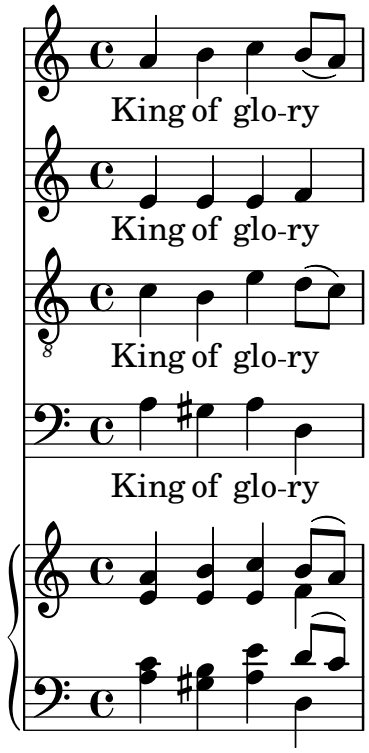
Plusieurs méthodes permettent de générer différentes versions d'une partition à partir d'une même source. Les variables – ou identifiants – sont sûrement le moyen le plus simple de combiner de différentes manières des passages relativement longs, alors que les balises permettront de sélectionner de courts fragments selon leur utilisation.

Quelle que soit la méthode utilisée, séparer la notation de la structure de la partition vous donnera plus de liberté dans l'agencement de l'ouvrage final, puisque vous ne reviendrez pas sur la musique qui le compose.

22.2.1 Utilisation de variables

Un fragment musical identifié par une variable est réutilisable à divers endroits de la partition, comme nous l'avons vu à la rubrique Section "Organisation du code source avec des variables" dans *Manuel d'initiation*. Par exemple, une partition pour chœur *a cappella* comporte souvent une réduction pour piano reprenant toutes les voix ; il s'agit de la même musique, et vous ne devrez donc la saisir qu'une seule fois. D'autre part, la musique issue de deux variables peut se combiner sur une seule portée, comme nous l'avons vu à la rubrique Section 5.2.5 [Regroupement automatique de parties], page 238. Prenons l'exemple suivant :

```
sopranoMusic = \relative { a'4 b c b8( a) }
altoMusic = \relative { e'4 e e f }
tenorMusic = \relative { c'4 b e d8( c) }
bassMusic = \relative { a4 gis a d, }
allLyrics = \lyricmode { King of glo -- ry }
<<
  \new Staff = "Soprano" \sopranoMusic
  \new Lyrics \allLyrics
  \new Staff = "Alto" \altoMusic
  \new Lyrics \allLyrics
  \new Staff = "Tenor" {
    \clef "treble_8"
    \tenorMusic
  }
  \new Lyrics \allLyrics
  \new Staff = "Bass" {
    \clef "bass"
    \bassMusic
  }
  \new Lyrics \allLyrics
  \new PianoStaff <<
    \new Staff = "RH" {
      \set Staff.printPartCombineTexts = ##f
      \partCombine \sopranoMusic \altoMusic
    }
    \new Staff = "LH" {
      \set Staff.printPartCombineTexts = ##f
      \clef "bass"
      \partCombine \tenorMusic \bassMusic
    }
  >>
>>
```



Générer une partition chorale ou la réduction pour piano ne requiert que de modifier la structure des éléments, sans aucunement toucher à la musique.

Dans le cas d'une partition relativement longue, vous pouvez isoler la définition des différentes variables dans des fichiers séparés que vous rappellerez ensuite, comme indiqué à la rubrique Section 22.1 [Insertion de fichiers LilyPond], page 644.

22.2.2 Utilisation de balises

La commande `\tag` affecte un nom à une expression musicale.

```
\tag #'toto { ... }
```

Les expressions ainsi balisées peuvent aisément se manipuler comme un tout.

Conserver et exclure de la musique balisée

Si la commande `\keepWithTag` permet de conserver une expression musicale balisée, la commande `\removeWithTag` exclura la musique balisée.

Filtre	**Résultat**
Musique balisée précédée de `\keepWithTag #'`*nom* ou `\keepWithTag #'(`*nom1 nom2...*`)`	Musique non balisée et musique balisée par l'un des noms de balise fournis seront incluses ; la musique balisée autrement est exclue.
Musique balisée précédée de `\removeWithTag #'`*nom* ou `\removeWithTag #'(`*nom1 nom2...*`)`	Musique non balisée et fragments appelés autrement que par l'un des noms fournis seront inclus ; la musique balisée par l'un des noms mentionnés est exclue.

Musique balisée non précédée de
`\keepWithTag`
ou
`\removeWithTag`
	Musique balisée et non balisée seront incluses.

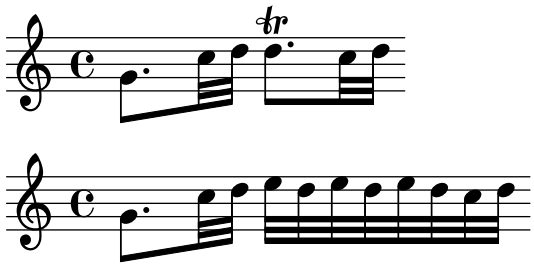
L'argument des commandes `\tag`, `\keepWithTag` et `\removeWithTag` doit être un symbole ou une liste de symboles (tel que `#'conducteur` ou `#'(violonI violonII)`), suivi d'une expression musicale. Si, *et seulement si* les symboles sont des identifiants LilyPond valides (caractères alphabétiques uniquement, sans chiffre, souligné ou tiret) qui ne peuvent se confondre avec des notes, le préfixe `#'` peut s'omettre et, pour raccourcir, une liste de symboles peut utiliser la virgule en séparateur – autrement dit, `\tag #'(violinI violinII)` peut s'écrire `\tag violinI,violinII`. Ceci s'applique aussi bien pour `\keepWithTag` que pour `\removeWithTag`.

Les commandes de balisage sont des fonctions musicales ; elles ne peuvent donc s'utiliser pour filtrer des éléments qui ne sont pas des expressions musicales tels que des blocs `\book` ou `\score`.

Dans l'exemple qui suit, nous obtenons deux versions du même extrait, l'une pour le conducteur, l'autre pour l'instrumentiste qui, elle, comportera les ornements développés.

```
music = \relative {
  g'8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \keepWithTag #'trills \music
}
\score {
  \keepWithTag #'expand \music
}
```

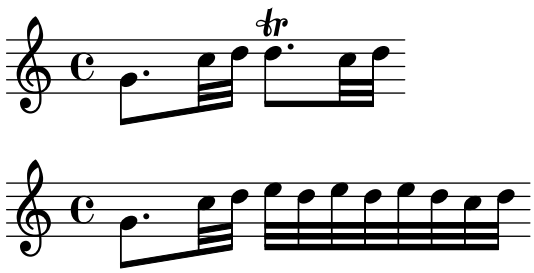


Il est parfois plus aisé d'exclure des fragments :

```
music = \relative {
  g'8. c32 d
  \tag #'trills { d8.\trill }
  \tag #'expand { \repeat unfold 3 { e32 d } }
  c32 d
 }

\score {
  \removeWithTag #'expand \music
}
\score {
```

```
  \removeWithTag #'trills \music
}
```



Lorsque des balises concernent des alternatives de durée non nulle, ces alternatives sont souvent conceptuellement simultanées. Il vaut alors mieux mettre ces alternatives dans des expressions musicales simultanées de telle sorte que l'expression musicale ait la même durée quelle que soit la balise retenue. Ceci est tout particulièrement important lorsque les balises sont utilisées conjointement avec des commandes telles que \cueDuring.

```
outputTypeTag = "isScore"
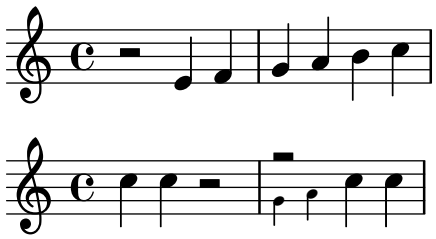
firstInstrument = \relative c' {
  <<
    \tag #'isPart {
      \cueDuring "quoteSecondInstrument" #UP { r2 } }
    \tag #'isScore { r2 }
  >>
  e4 f |
  g4 a b c |
}

secondInstrument= \relative c'' {
  c4 c r2 |
  \cueDuring "quoteFirstInstrument" #DOWN { r2 }
  c4 c |
}

\addQuote quoteFirstInstrument \firstInstrument
\addQuote quoteSecondInstrument \secondInstrument

\new Staff {
  \keepWithTag \outputTypeTag \firstInstrument
}

\new Staff {
  \keepWithTag \outputTypeTag \secondInstrument
}
```



Ce principe de filtrage peut s'appliquer aux articulations, textes, etc. Il suffit de positionner

```
-\tag #ma-balise
```

avant l'articulation ou le texte, comme ici :
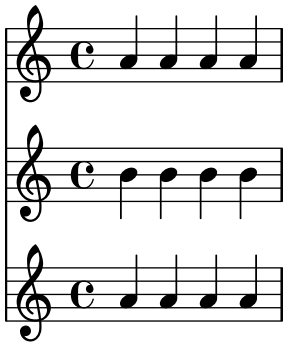
```
c1-\tag #'doigt ^4
c1-\tag #'gaffe ^"Attention !"
```

Ceci définira une note avec une indication conditionnelle de doigté ou un texte.

Vous pouvez baliser différemment la même expression musicale en saisissant plusieurs `\tag` ou bien en combinant plusieurs balises dans une liste :

```
music = \relative c'' {
  \tag AA \tag AB { a4 a a a }
  \tag BB,both { b4 b b b }
}
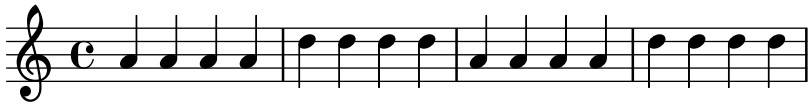
<<
  \keepWithTag AA \music
  \keepWithTag BB \music
  \keepWithTag AB \music
>>
```



L'application concomitante de plusieurs filtres `\removeWithTag` à la même expression musicale permet d'exclure plusieurs balisages. Une liste fournie en argument à un unique `\removeWithTag` produira le même effet :

```
music = \relative c'' {
  \tag AA { a4 a a a }
  \tag BB { b4 b b b }
  \tag CC { c4 c c c }
  \tag DD { d4 d d d }
}

\new Voice {
  \removeWithTag BB \removeWithTag CC \music
  \removeWithTag BB,CC \music
}
```
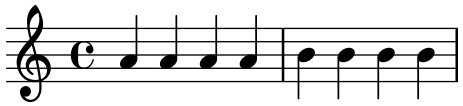


L'application de plus d'un filtre `\keepWithTag` à la même expression musicale aboutit à l'exclusion de **tous** les balisages, ce qui n'est certainement pas l'effet attendu : si le premier filtre exclut tous les autres balisages, l'application du second exclura les effets du premier. L'utilisation

d'une unique commande \keepWithTag avec une liste de balises est en pareil cas des plus pertinentes : seront exclus tous les fragments non concernés par l'une quelconque des balises mentionnées. Le code suivant, à titre d'exemple, imprimera les \tags *violinI* et *violinII*, mais ni *viola* ni *cello*.

```
music = \relative c'' {
  \tag violinI { a4 a a a }
  \tag violinII { b4 b b b }
  \tag viola { c4 c c c }
  \tag cello { d4 d d d }
}

\new Staff {
  \keepWithTag violinI,violinII \music
}
```



Utilisation de regroupements de balises

Bien que \keepWithTag soit efficace pour gérer *un* jeu d'alternatives, le rejet de musique filtrée par des balises étrangères se révèle problématique lorsque les \tag sont utilisés à plusieurs fins. Des « groupements de balises » peuvent alors être déclarés à l'aide de \tagGroup. Un code tel que

```
\tagGroup violinI,violinII,viola,cello
```

met les quatre balises dans un seul regroupement. Par conséquent, taper
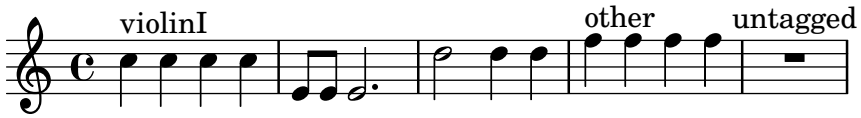
```
\keepWithTag violinI ...
```

ne prendra désormais en compte que la musique concernée par la balise violinI du groupe de filtres : tout élément de la musique qui serait balisé par l'un des autres filtres de ce jeu sera rejeté, comme le montre l'exemple suivant.

```
\tagGroup violinI,violinII
\tagGroup original,arranged

music = \relative {
  \tag violinI { c''4^"violinI" c c c }
  \tag violinII { a2 a }
  \tag original { e8 e e2. }
  \tag arranged { d'2 d4 d }
  \tag other { f^"other" f f f }
  R1^"untagged"
}

\new Voice {
  \keepWithTag violinI \music
}
```



Il est à noter que les balises ne peuvent être incluses que dans un seul regroupement.

Il est possible de modifier des regroupements de balises. Une autre manière de les définir consiste à utiliser \tagGroupRef dans le but de références le groupement pour modification ultérieure. Libeller

```
tg = \tagGroupRef violinI,violinII
```

revient à mentionner

```
tg = #'(violinI violinII)
\tagGroup \tg
```

Diverses possibilités permettent de modifier des regroupements de balises existant.

- L'annulation de tous les effets induits de tous les \tagGroup ou \tagGroupRef s'obtient simplement par un \resetTagGroups.

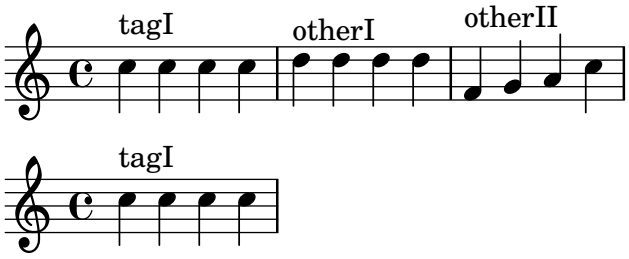
```
\tagGroup tagI,tagII

music = \relative {
  \tag tagI { c''4^"tagI" c c c }
  \tag tagII { a2^"tagII" a }
  \tag otherI { d4^"otherI" d d d }
  \tag otherII { f,4^"otherII" g a c }
}

\new Voice {
  \keepWithTag tagI \music
}

\resetTagGroups

\new Voice {
  \keepWithTag tagI \music
}
```



- L'annulation des effets d'un regroupement particulier s'obtient à l'aide d'un \resetTagGroup auquel est fourni le regroupement considéré, donc toutes ses balises. \tagGroupRef se révèle très pratique pour définir le regroupement de balises.

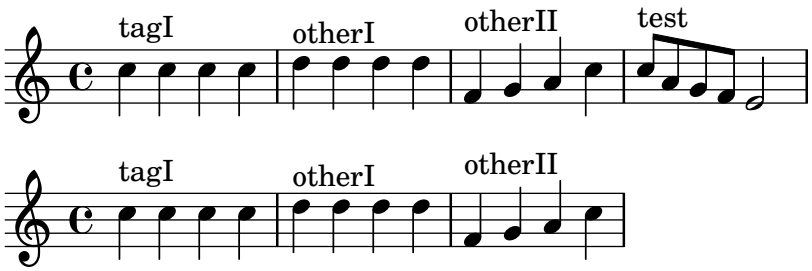
```
tgI = \tagGroupRef tagI,tagII
tgII = \tagGroupRef otherI,otherII

music = \relative {
  \tag tagI { c''4^"tagI" c c c }
  \tag tagII { a2^"tagII" a }
  \tag otherI { d4^"otherI" d d d }
  \tag otherII { f,4^"otherII" g a c }
  \tag test { c8^"test" a g f e2 }
}
```

```
% 'otherI', 'otherII' and 'test' are not in the same group
% as 'tagI' so they stay untouched.
\new Voice {
  \keepWithTag tagI \music
}

\resetTagGroup \tgI

% now only 'otherI' and 'otherII' are in a tag group,
% so '\keepWithTag tagI' filters 'tagII' and 'test'.
\new Voice {
  \keepWithTag tagI \music
}
```



- L'ajout ou le retrait d'une balise à un regroupement à l'aide de \addToTagGroup et \removeFromTagGroup constituent d'autres moyens de modifier des regroupements.

```
tgI = \tagGroupRef tagI,tagII
tgII = \tagGroupRef otherI,otherII

music = \relative {
  \tag tagI { c''4^"tagI" c c c }
  \tag tagII { a2^"tagII" a }
  \tag otherI { d4^"otherI" d d d }
  \tag otherII { f,4^"otherII" g a c }
  \tag test { c8^"test" a g f e2 }
}

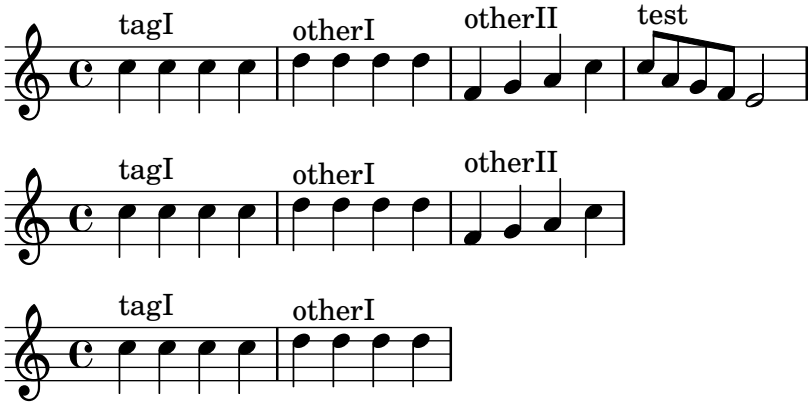
% 'otherI', 'otherII' and 'test' are not in the same group
% as 'tagI' so they stay untouched.
\new Voice {
  \keepWithTag tagI \music
}

\addToTagGroup \tgI test

% now 'tagI', 'tagII' and 'test' are in the same group
% so 'tagII' and 'test' are removed.
\new Voice {
  \keepWithTag tagI \music
}

\removeFromTagGroup \tgII otherII
\addToTagGroup \tgI otherII
```

```
% now 'otherII' is removed from 'tgII' and
% added to 'tgI' so only 'otherI' is in its
% own tag group and is not filtered.
\new Voice {
  \keepWithTag tagI \music
}
```



Préposition et apposition a de la musique balisée

Il peut arriver que vous ayez besoin de raccorder quelque chose en un point particulier d'une expression musicale. Les commandes \pushToTag et \appendToTag permettent d'insérer du matériau, qu'il soit antérieur ou postérieur, à des constructions musicales existantes. Les différentes possibilités sont les suivantes :

Musique séquentielle ou simultanée
: Lorsqu'a été balisée l'intégralité d'une construction {...} ou <<...>>, peuvent venir s'insérer, avant ou après, des expression musicales.

Accords
: Lorsqu'a été balisé un accord <...>, peuvent venir s'y ajouter, avant ou après, d'autres notes ou des articulations, ces dernières pour l'accord dans sa globalité.
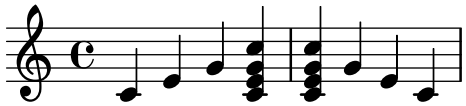
Notes et silences
: Lorsque la musique balisée est une note (y compris à l'intérieur d'un accord), ou un silence, peuvent venir s'y ajouter, avant ou après, d'autres articulations.

  Afin d'ajouter d'autres *notes*, il est préférable de les placer dans une construction d'accord et baliser *l'accord*. Notez bien qu'il n'est pas possible de baliser une simple *articulation* et y ajouter quelque chose, puisqu'il ne s'agit pas d'une liste ; il vaut alors mieux baliser la note.

Ces deux instructions sont affectées d'une balise, le matériau à raccorder à chaque instance de la balise, et l'expression balisée.

```
music = { \tag #'here { \tag #'here <<c''>> } }

{
  \pushToTag #'here c'
  \pushToTag #'here e'
  \pushToTag #'here g' \music
  \appendToTag #'here c'
  \appendToTag #'here e'
  \appendToTag #'here g' \music
}
```
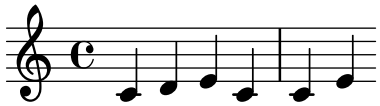


Remplacement de musique balisée

Il est possible de remplacer de la musique balisée par une autre musique.

```
music = { c' \tag #'ici { d' } e' }
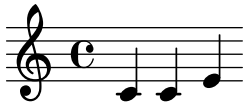
{
  \music
  \replaceWithTag #'ici c' \music
}
```



Il faut bien garder à l'esprit que la balise disparaît avec son remplacement ; elle ne peut donc être remplacée qu'une seule fois. Le filtrage aussi n'est plus possible à la suite du remplacement.

```
music = { c' \tag #'ici { d' } e' }

{
  \removeWithTag #'ici % sans effet à cause du remplacement de balise
  \replaceWithTag #'ici c'
  \music
}
```



Balise et *markup*

La commande \tag peut s'utiliser aussi dans des \markup. LilyPond fournit les commandes \keep-with-tag, \remove-with-tag, \push-to-tag, \append-to-tag et \replace-with-tag qui se comportent comme celles utilisées pour les expressions musicales.

```
test = \markup {
  \tag #'AA a
  \tag #'BB b
  \tag #'CC c
}

\markup { \keep-with-tag #'BB \test }
\markup { \remove-with-tag #'BB \test }
\markup { \push-to-tag #'CC pre \test }
\markup { \append-to-tag #'CC post \test }
\markup { \replace-with-tag #'CC sub \test }
```

b

a c

a b pre c
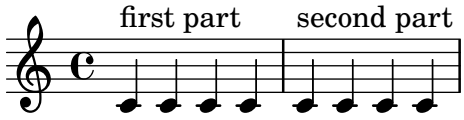
a b c post

a b sub

Les commandes musicales \keepWithTag et \removeWithTag filtrent aussi les balises des portions \markup incluses dans la musique.

```
music = \relative {
  c'4^\markup { \tag #'one first \tag #'two second part } c c c
}

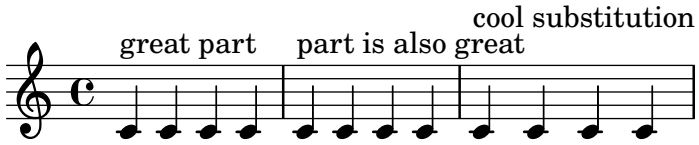
{
  \keepWithTag #'one \music
  \removeWithTag #'one \music
}
```



Il est aussi possible d'insérer ou ajouter quelque chose à un \markup attaché aux objets musicaux. On ne peut toutefois utiliser \pushToTag, \appendToTag ou \replaceWithTag puisqu'ils ne traitent que de la musique ; cependant, \pushToTagMarkup, \appendToTagMarkup et \replaceWithTagMarkup sont conçus pour rajouter un *markup*.

```
music = \relative {
  c'4^\markup { \tag #'part part } c c c
}
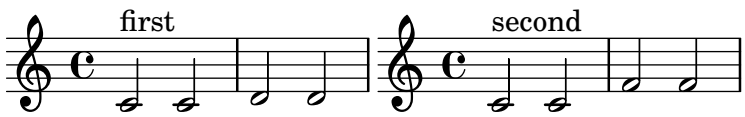
{
  \pushToTagMarkup #'part "great" \music
  \appendToTagMarkup #'part \markup { is also great } \music
  \replaceWithTagMarkup #'part \markup { cool substitution } \music
}
```



Le filtrage des balises fonctionne aussi lorsque de la musique comprise dans un bloc \score est membre d'une commande de *markup*.

```
music = \relative {
  c'2^\markup { \tag #'first first \tag #'second second } c
  \tag #'first { d d }
  \tag #'second { f f }
}

\markup {
  \keep-with-tag #'first \score { \music }
  \remove-with-tag #'first \score { \music }
}
```



Une attention particulière doit toutefois être portée lors de l'utilisation de balises avec des listes de *markups*. Les fonctions de filtrage fonctionnent comme prévu, mais les commandes telles que \push-to-tag, \append-to-tag et \replace-with-tag posent problème.

```
\markup {
  \remove-with-tag #'test { a \tag #'test { b c } d }
}
\markup {
  \push-to-tag #'test "twice" { a \tag #'test { b c } d }
}
\markup {
  \replace-with-tag #'test "twice" { a \tag #'test { b c } d }
}
```

a d

a twice b twice c d

a twice twice d

La raison de ce comportemenet tient au fait que LilyPond transforme

```
\tag #'test { b c }
```

de manière interne en

```
\tag #'test b
\tag #'test c
```

et donc insère deux fois le texte donné. Le balisage d'une liste à laquelle doit s'insérer ou s'ajouter quelque chose s'obtient en utilisant la commande \tag-list.

```
\markup {
  \push-to-tag #'test "once" { a \tag-list #'test { b c } d }
}
```

a once b c d

Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section 22.1 [Insertion de fichiers LilyPond], page 644, Section 5.2.5 [Regroupement automatique de parties], page 238.

Problèmes connus et avertissements

L'application d'un \relative à une expression musicale obtenue par filtrage à l'aide de \keepWithTag ou \removeWithTag peut générer des changements d'octave, puisque seules les hauteurs récupérées dans ce filtre seront prises en considération. Une instruction \relative qui précède les commandes \keepWithTag ou \removeWithTag permet d'éviter ce risque, dans la mesure où elle viendra « recaler » ces hauteurs récupérées.

Même lors de l'utilisation de regroupements de balises, les commandes telles que \keepWithTag ne sont pas cumulatives. du code comme

```
\keepWithTag violonI,original \laMusique
```

ne produira pas le même résultat que

```
\keepWithTag violonI \keepWithTag original \laMusique
```

22.2.3 Globalisation des réglages

Vous pouvez regrouper dans un fichier indépendant vos réglages personnels que vous inclurez au besoin :

```
lilypond -dinclude-settings=MES_REGLAGES.ly MA_PARTITION.ly
```

Vous pouvez ainsi stocker dans un fichier séparé vos réglages en matière de format de papier, de fontes utilisées ou vos définitions particulières, que vous pourrez charger par des options `-dinclude-settings`. Selon le fichier de réglages que vous mentionnerez, vous obtiendrez facilement différentes éditions à partir d'une même source quelle qu'elle soit.

Cette technique peut s'utiliser en combinaison avec des feuilles de styles, comme indiqué au chapitre Section "Feuilles de style" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Feuilles de style" dans *Manuel d'initiation*, Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section 22.1 [Insertion de fichiers LilyPond], page 644.

22.3 Utilisation de fonctions musicales

Une adaptation ou un affinage qui devient récurrent parce que doit s'appliquer à différentes expressions musicales peut faire l'objet d'une *fonction musicale*. Nous ne traiterons ici que des fonctions de *substitution*, dont le but est de substituer une variable en un bout de code LilyPond. D'autres fonctions, plus complexes, sont abordées au chapitre Section "Fonctions musicales" dans *Extension de LilyPond*.

22.3.1 Syntaxe d'une fonction de substitution

La rédaction d'une fonction chargée de substituer du code LilyPond à une variable est chose relativement aisée. Une telle fonction est de la forme

```
fonction =
#(define-music-function
     (arg1 arg2...)
     (type1? type2?...)
   #{
     ...musique...
   #})
```

où

argN — Le *n*ième argument.

typeN? — Un *type de prédicat* Scheme pour lequel *argN* doit renvoyer `#t`.

...musique... — Du code LilyPond tout ce qu'il y a de plus ordinaire, avec des '`$`' (là où seule une construction LilyPond est autorisée) et des '`#`' (lorsqu'il s'agit d'une valeur en Scheme, d'un argument de fonction musicale ou de musique faisant partie d'une liste) pour référencer les arguments (par ex. '`#arg1`').

La liste des types de prédicat est aussi obligatoire. Voici quelques uns des types de prédicat les plus utilisés dans les fonctions musicales :

boolean?

cheap-list? *(au lieu de* « list? »*, pour accélérer le traitement)*

ly:duration?
ly:music?
ly:pitch?
markup?
number?
pair?
string?
symbol?

Une liste plus fournie est disponible à l'annexe Section B.25 [Types de prédicats prédéfinis], page 989. Vous pouvez par ailleurs définir vos propres types de prédicat.

Voir aussi

Manuel de notation : Section B.25 [Types de prédicats prédéfinis], page 989.

Manuel d'extension : Section "Fonctions musicales" dans *Extension de LilyPond*.

Fichiers d'initialisation : `lily/music-scheme.cc`, `scm/c++.scm`, `scm/lily.scm`.
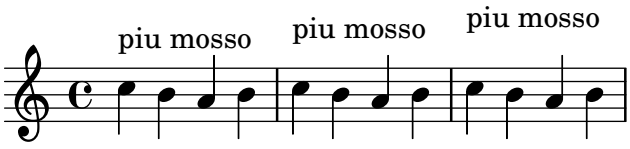
22.3.2 Exemples de fonction de substitution

La présente rubrique regroupe quelques exemples de fonction substitutive. Le propos est ici d'illustrer les possibilités qu'offrent les fonctions de substitution simple.

Dans ce premier exemple, nous définissons une fonction dans le but de simplifier le réglage du décalage d'une annotation (un `TextScript`).

```
padText =
#(define-music-function
     (padding)
     (number?)
   #{
     \once \override TextScript.padding = #padding
   #})
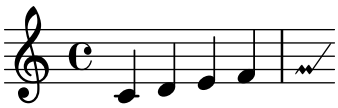
\relative {
  c''4^"piu mosso" b a b
  \padText 1.8
  c4^"piu mosso" b a b
  \padText 2.6
  c4^"piu mosso" b a b
}
```



Nous pouvons utiliser autre chose que des nombres au sein d'une fonction, y compris une expression musicale :

```
custosNote =
#(define-music-function
     (note)
     (ly:music?)
   #{
     \tweak NoteHead.stencil #ly:text-interface::print
     \tweak NoteHead.text
        \markup \musicglyph "custodes.mensural.u0"
     \tweak Stem.stencil ##f
     #note
   #})
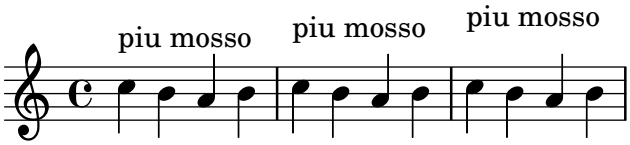
\relative { c'4 d e f \custosNote g }
```



Ces fonctions sont toutes deux des expressions uniques simples dans lesquelles seul le dernier élément d'un appel à une fonction ou une dérogation est absent. Dans ce cas particulier de définition d'une fonction, une syntaxe alternative et plus simple autorise à se cantonner à écrire la partie constante de l'expression et remplacer son dernier élément, absent, par `\etc` :
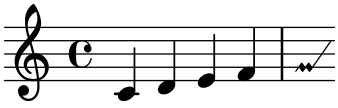
```
padText =
  \once \override TextScript.padding = \etc

\relative {
  c''4^"piu mosso" b a b
  \padText 1.8
  c4^"piu mosso" b a b
  \padText 2.6
  c4^"piu mosso" b a b
}
```




```
custosNote =
  \tweak NoteHead.stencil #ly:text-interface::print
  \tweak NoteHead.text
     \markup \musicglyph "custodes.mensural.u0"
  \tweak Stem.stencil ##f
  \etc
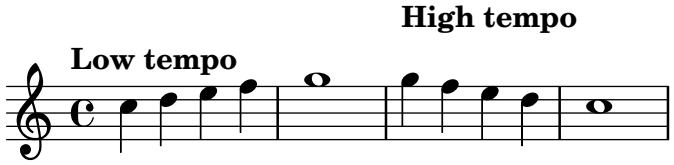
\relative { c'4 d e f \custosNote g }
```



Une fonction de substitution peut traiter plusieurs arguments :

```
tempoPadded =
#(define-music-function
     (padding tempotext)
     (number? markup?)
   #{
     \once \override Score.MetronomeMark.padding = #padding
     \tempo \markup { \bold #tempotext }
   #})

\relative {
  \tempo \markup { "Low tempo" }
  c''4 d e f g1
  \tempoPadded 4.0 "High tempo"
  g4 f e d c1
}
```
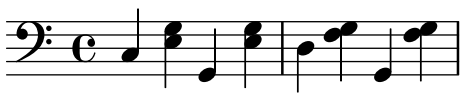


22.3.3 Comment éviter le partage d'expressions musicales

Lors de l'écriture de fonctions musicales, il est important de respecter la règle suivante : une même expression musicale de doit pas se retrouver à plusieurs endroits. Voici, à titre d'exemple, une fonction problématique :

```
simpleAccompaniment =
#(define-music-function
   (bass-1 bass-2 chord) (ly:music? ly:music? ly:music?)
   #{
     #bass-1 #chord #bass-2 #chord
   #})

{
  \clef bass
  \simpleAccompaniment c g, <e g>
  \simpleAccompaniment d g, <f g>
}
```
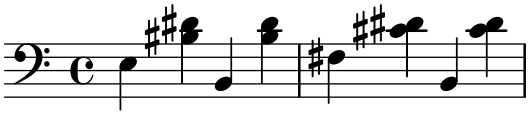


Le problème avec cette fonction est évident si le résultat est transposé :

```
simpleAccompaniment =
#(define-music-function
   (bass-1 bass-2 chord) (ly:music? ly:music? ly:music?)
   #{
     #bass-1 #chord #bass-2 #chord
   #})

\transpose c e {
  \clef bass
```

```
  \simpleAccompaniment c g, <e g>
  \simpleAccompaniment d g, <f g>
}
```
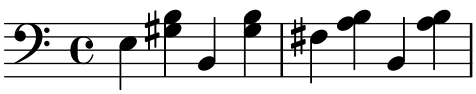


Alors que les notes de basse sont correctes, l'accord n'est pas transposé comme il faut – il est en fait transposé deux fois. La raison tient au fait que l'expression musicale *chord* est utilisé deux fois dans le résultat de la fonction sans le recopier. Les fonctions telles que `\transpose` modifient directement l'objet musical – dans le cas de `\transpose`, les hauteurs sont changées. Lorsqu'un même objet musical est réutilisé, les modifications appliquées à l'un des endroits où il est utilisé affectent les deux puisqu'ils concernent le même objet. Dans le cas présent, `\transpose` rencontre l'objet deux fois, donc le transpose deux fois.

L'un des moyens de corriger cette fonction consiste à utiliser '$' au lieu de '#' pour référencer les variables, ce qui en réalise une copie. La différence entre '#' et '$' est abordée dans Section "Syntaxe Scheme dans LilyPond" dans *Extension de LilyPond*.

```
simpleAccompaniment =
#(define-music-function
   (bass-1 bass-2 chord) (ly:music? ly:music? ly:music?)
   #{
     $bass-1 $chord $bass-2 $chord
   #})

\transpose c e {
  \clef bass
  \simpleAccompaniment c g, <e g>
  \simpleAccompaniment d g, <f g>
}
```



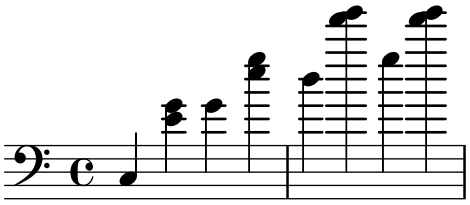
22.3.4 Fonctions de subsitution et octave relative

Lorsque `\relative` s'applique à une expression musicale, celle-ci est parcourue à la recherche de notes, et les hauteurs sont modifiées dans leur ordre d'apparition, l'octave de chaque hauteur étant changée selon sa marque d'octaviation ('' et ',') et la hauteur qui précède. Dans le cadre d'une fonction substitutive, ceci peut mener à des situations où la musique est « relativisée » de manière surprenante parce que le résultat de la fonction utilise les paramètres plusieurs fois ou dans un ordre différent. Examinons la fonction suivante et comment son résultat réagit au `\relative`.

```
simpleAccompaniment =
#(define-music-function
   (bass-1 bass-2 chord) (ly:music? ly:music? ly:music?)
   #{
     $bass-1 $chord $bass-2 $chord
   #})

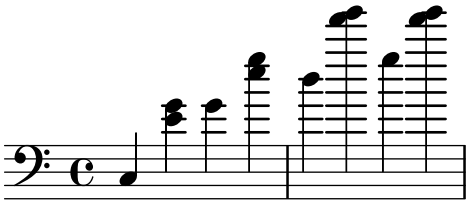
\relative {
  \clef bass
```

```
  \simpleAccompaniment c g <e' g>
  \simpleAccompaniment d g, <f' g>
}
```



Dans cet exemple, le résultat est identique à
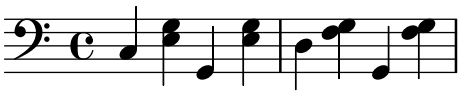
```
\relative {
  \clef bass
  c <e' g> g <e' g>
  d <f' g>  g, <f' g>
}
```



Ce n'est toutefois pas le résultat auquel on s'attendrait à l'utilisation la fonction `\simpleAccompaniment`. La hauteur `g,` est relative à la première hauteur de l'accord qui précède, `<e' g>`, bien qu'il soit libellé après le `c` de la source. De toute évidence, les hauteurs devraient être relatives selon l'ordre dans lequel elles sont transmises à la fonction, non pas selon leur ordre d'apparition en sortie de la fonction. Le moyen d'y parvenir est un recours à la fonction Scheme `make-relative`. Ses arguments sont : une liste de variables, une expression de référence, et une expression musicale principale. L'expression de référence est réputée être une représentation de la façon dont les variables auront été saisies. Il s'agit la plupart du temps d'une simple expression de la forme `#{ ... #}` contenant les variables dans l'ordre. Attention à ne pas réaliser de copie dans l'expression de référence – en particulier, il faut y utiliser '`#`', pas '`$`'. L'exemple précédent peut se corriger à l'aide de `make-relative` de la manière suivante :

```
simpleAccompaniment =
#(define-music-function
   (bass-1 bass-2 chord) (ly:music? ly:music? ly:music?)
   (make-relative
    (bass-1 bass-2 chord)
    #{ #bass-1 #bass-2 #chord #}
    #{ $bass-1 $chord $bass-2 $chord #}))

\relative {
  \clef bass
  \simpleAccompaniment c g <e' g>
  \simpleAccompaniment d g, <f' g>
}
```



22.4 Caractères spéciaux

22.4.1 Codage du texte

LilyPond utilise le jeu de caractères défini par le consortium Unicode et la norme ISO/CEI 10646. Chaque caractère est identifié par un nom unique et associé à un point de code, ce qui permet dans l'absolu de couvrir tous les langages. Unicode permet de coder tous les caractères utilisés par toutes les langues écrites du monde. LilyPond utilise le codage UTF-8 (UTF pour *Unicode Transformation Format*) qui permet de représenter les caractères latins sur un octet et les autres sur une longueur allant jusqu'à quatre octets.
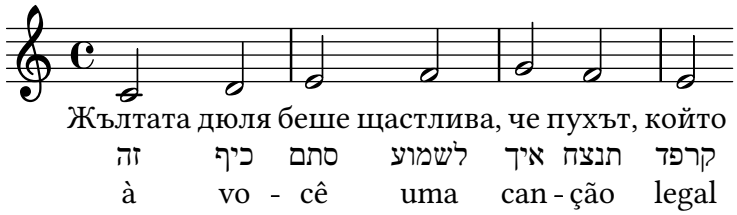
L'apparence réelle des caractères est déterminée par les glyphes ou graphèmes tels que définis dans les différentes polices disponibles. Une police, ou une fonte, définit la mise en correspondance d'un sous-ensemble de points de code unicode en glyphes. LilyPond recourt à la bibliothèque Pango pour assurer le rendu des textes multilingues.

LilyPond n'effectue aucune conversion d'encodage que ce soit. Ceci implique donc que tout texte – un titre, des paroles ou même une instruction musicale – comportant des caractères non ASCII soit codé en UTF-8. Le plus sûr moyen de saisir du texte de la sorte consiste à utiliser un éditeur supportant l'unicode et à enregistrer vos fichier en UTF-8. C'est le cas pour la plupart des éditeurs actuels, que ce soit vim, Emacs, jEdit et Gedit. Tous les systèmes Windows postérieurs à NT utilisent Unicode en natif ; même Notepad est capable d'éditer et sauvegarder un fichier en UTF-8 – sans parler de l'excellente alternative qu'est BabelPad.

La compilation d'un fichier LilyPond comportant des caractères non ASCII qui n'aurait pas été enregistré dans l'encodage UTF-8 vous renverra l'erreur

```
FT_Get_Glyph_Name () erreur : invalid argument
```

Voici un exemple utilisant du texte en cyrillique, en hébreux et en portugais.



22.4.2 Unicode

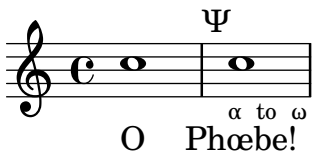
Lorsque vous avez besoin d'un caractère dont vous connaissez le point de code mais que votre éditeur ne permet pas de saisir directement, vous pouvez utiliser les instructions `\char ##xhhhh` ou `\char #dddd` au sein d'un bloc `\markup` – `hhhh` et `dddd` correspondant respectivement à la valeur hexadécimale ou décimale. Même s'il est inutile de saisir les zéros superflus, il est de bon ton de stipuler les quatre caractères formant la représentation hexadécimale. Évitez cependant l'encodage UTF-8 d'un point de code après un `\char` ; les encodages UTF-8 comprennent un bit supplémentaire indiquant le nombre d'octets. Une table de correspondance entre les codes Unicode et le nom des caractères ainsi que leur code hexadécimal est disponible sur le site du consortium Unicode, `https://www.unicode.org/`.

Par exemple, `\char ##x03BE` et `\char #958` correspondent tous deux au caractère unicode U+03BE, dénommé « Greek Small Letter Xi ».

Quel que soit le point de code spécifié de cette manière, il ne vous sera alors pas nécessaire d'enregistrer votre fichier en UTF-8. Vous devrez toutefois disposer d'une fonte contenant ce caractère qui soit accessible à LilyPond.

L'exemple suivant illustre la manière d'insérer un caractère sous sa forme hexadécimale, à la fois dans un repère textuel, dans une articulation, dans des paroles et dans du texte indépendant.

```
\score {
  \relative {
    c''1
    \textMark \markup { \char ##x03A8 }
    c1_\markup { \tiny { \char ##x03B1 " to " \char ##x03C9 } }
  }
  \addlyrics { O \markup { \concat { Ph \char ##x0153 be! } } }
}
\markup { "Copyright 2008--2026" \char ##x00A9 }
```



Copyright 2008--2026 ©

Le signe *copyright* dans le champ de titrage consacré s'inscrit de la manière suivante :

```
\header {
  copyright = \markup { \char ##x00A9 "2008" }
}
```

22.4.3 Équivalents ASCII

Dès lors que vous aurez inclus la liste de leur équivalent ASCII, LilyPond reconnaîtra un certain nombre de caractères spéciaux :

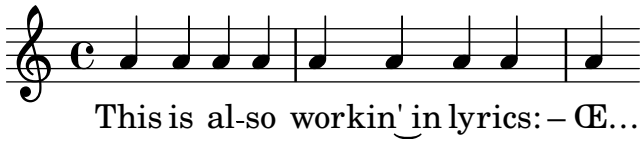
```
\paper {
  #(include-special-characters)
}

\markup "&flqq; &ndash; &OE;uvre incomplète&hellip; &frqq;"

\score {
  \new Staff { \repeat unfold 9 a'4 }
  \addlyrics {
    This is al -- so wor -- kin'~in ly -- rics: &ndash;_&OE;&hellip;
  }
}

\markup \column {
  "The replacement can be disabled:"
  "&ndash; &OE; &hellip;"
  \override #'(replacement-alist . ()) "&ndash; &OE; &hellip;"
}
```

« – Œuvre incomplète… »



The replacement can be disabled:
– Œ …
– &OE; …

L'extension de cette liste est possible aussi bien de manière globale :

```
\paper {
  #(add-text-replacements!
    '(("100" . "hundred")
      ("dpi" . "dots per inch")))
}
\markup "A 100 dpi."
```

A hundred dots per inch.

qu'en un point particulier de votre source :

```
\markup \replace #'(("100" . "hundred")
                    ("dpi" . "dots per inch")) "A 100 dpi."
```

A hundred dots per inch.

Le remplacement n'affectera pas nécessairement une chaîne ; il peut s'agir d'un *markup* quelconque. Au niveau de la syntaxe, ceci requiert d'utiliser la syntaxe de quasi-citation de Scheme, à savoir une apostrophe inversée ‘`’ au lieu d'une apostrophe normale ‘'’ pour écrire la liste associative.

```
\markup \replace
  #`(("2nd" . ,#{ \markup \concat { 2 \super nd } #})) "2nd time"
```

2nd time

Ces alias ne pourront plus, quant à eux, faire l'objet d'un remplacement.

Voir aussi

Manuel de notation : Section B.12 [Liste des caractères spéciaux], page 943.

Fichiers d'initialisation : `ly/text-replacements.ly`.

23 Contrôle des sorties

23.1 Extraction de fragments musicaux

LilyPond permet d'extraire des fragments d'une partition. La variable `clip-regions`, qui se place dans le bloc `\layout` ou `\paper`, permet de définir explicitement le ou les emplacements de la musique concernés ; ils seront extraits en lançant `lilypond` avec l'option `-dclip-systems`.

```
\layout {
  clip-regions
  = #(list (cons (make-rhythmic-location 5 1 2)
                 (make-rhythmic-location 7 3 4)))
}
```

L'exemple ci-dessus permet d'extraire un seul fragment *débutant* après une blanche dans la cinquième mesure (`5 1 2`) et *finissant* après trois noires dans la septième mesure (`7 3 4`).

D'autres fragments seront extraits dès lors que d'autres paires de `make-rhythmic-location` auront été ajoutées à la liste de `clip-regions`.

Lorsque des débuts ou fins de système sont inclus, les extensions à l'objet `System`, tels que les noms d'instrument, seront eux aussi inclus.

Les notes d'ornement en terminaison du fragment extrait ne seront pas incluses.

Chaque fragment fait l'objet d'un fichier particulier. La musique extraite est rendue comme si elle avait été littéralement « découpée » dans la partition. Par voie de conséquence, un fragment dépassant une ligne fera l'objet d'autant de fichiers séparés que de lignes de la partition complète. Partant de l'exemple ci-dessus couvre deux lignes du PDF résultant de `toto.ly`, les fichiers d'extraction s'appelleront `toto-from-5.1.2-to-7.3.4-clip.pdf` et `toto-from-5.1.2-to-7.3.4-clip-1.pdf`.

Voir aussi

Manuel de notation : Section 27.1 [Le bloc `\layout`], page 705.

Manuel d'utilisation : Voir Section “Utilisation en ligne de commande” dans *Utilisation des programmes*.

23.2 Ignorer des passages de la partition

Dans un travail de transcription ou de recopie de la musique, ce qui vous intéresse plus particulièrement se situe à la fin, là même où vous en êtes dans la notation. Dans le but de gagner du temps dans le processus de correction, vous pouvez « escamoter » le reste et ne générer que les dernières mesures en définissant une variable +particulière en début de fichier, comme ceci :
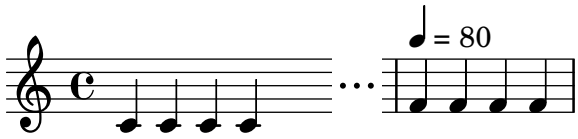
```
showLastLength = R1*5
\score { ... }
```

Ceci aura pour effet de ne générer que les cinq dernières mesures – si tant est que le morceau soit à 4/4 – de tous les `\score` de votre fichier. Dans le cas d'un œuvre conséquente, cette pratique s'avère fort utile puisqu'elle évite de tout générer. Vous pourriez aussi être amené à retravailler le début d'une œuvre, pour y ajouter une partie par exemple, auquel cas c'est la propriété `showFirstLength` que vous utiliserez.

Vous pouvez contrôler très finement les parties à escamoter, grâce au commutateur `Score.skipTypesetting` : lorsqu'il est activé, aucune gravure n'est réalisée. En tant que propriété du contexte `Score`, il affecte toutes les voix et portées – voir Section 33.1.2 [`Score` – le père de tous les contextes], page 760.

Ce commutateur agit aussi sur la sortie MIDI. Notez bien que tous les événements seront escamotés, y compris les changements de tempo ou d'instrument qui interviendraient avant que `skipTypesetting` ne soit désactivé.

```
\relative c' {
  c4 c c c
  \set Score.skipTypesetting = ##t
  d4 d d d
  \tempo 4 = 80
  e4 e e e
  \set Score.skipTypesetting = ##f
  f4 f f f
}
```



Commandes prédéfinies

`showLastLength`, `showFirstLength`.

Voir aussi

Manuel de notation : Chapitre 33 [Contextes d'interprétation], page 760, Section 33.1.2 [`Score` – le père de tous les contextes], page 760.

Référence des propriétés internes : Section "Tunable context properties" dans *Référence des propriétés internes*.

23.3 Formats de sortie alternatifs

En matière de partition imprimable, LilyPond génère par défaut des documents au format PostScript (PS) et Portable Document Format (PDF). Vous pouvez aussi obtenir des documents au format Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS) ou Portable Network Graphics (PNG) dès lors que vous aurez lancé LilyPond en ligne de commande avec l'option *ad hoc* – voir Voir Section "Utilisation en ligne de commande" dans *Utilisation des programmes* à ce sujet.

23.3.1 Sortie SVG

La sortie SVG peut accessoirement contenir des métadonnées pour les *grobs* (objets graphiques) tels que têtes de notes, silences, etc. Ces métadonnées peuvent correspondre aux attributs standards du format SVG comme `id` et `class`, ou bien à des attributs personnalisés. Les attributs et leur valeur se spécifient à l'aide d'une dérogation à la propriété `output-attributes` d'un *grob* par une liste associative (alist) en Scheme. Les valeurs peuvent être des nombres, chaînes ou symboles comme, par exemple :

```
{
  \once \override NoteHead.output-attributes =
  #'((id . 123)
     (class . "ceci cela")
     (data-quelconque . quelquechose))
  c
}
```

Le code ci-dessus produira la balise <g> (group) suivante dans le fichier SVG :

```
<g id="123" class="ceci cela" data-quelconque="quelquechose">
```

```
    ...NoteHead grob SVG elements...
  </g>
```

La balise <g> contient tous les éléments SVG d'un *grob* donné ; certains *grobs* génèrent de multiples éléments SVG. Dans la syntaxe SVG, le préfixe `data-` s'utilise pour les attributs de métadonnée personnalisée non-standard.

23.4 Intégration de fichiers à la sortie PDF

L'option en ligne de commande `-dembed-source-code` permet à LilyPond de « joindre » au fichier PDF final tous les fichiers sources nécessaires à la compilation (voir Voir Section "Options avancées de `lilypond`" dans *Utilisation des programmes*). Un lecteur de PDF permettra ensuite d'extraire ces annexes pour une utilisation future.

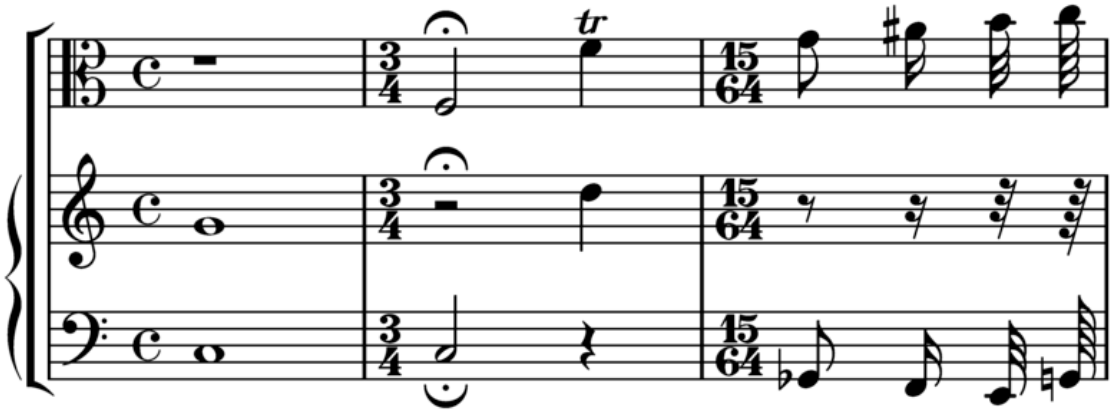
Dans le même ordre d'idée, il est possible d'adjoindre n'importe quel fichier dans le PDF résultant à l'aide de la fonction `ly:note-extra-source-file` (voir Section "Scheme functions" dans *Référence des propriétés internes*).

23.5 Changement des fontes musicales
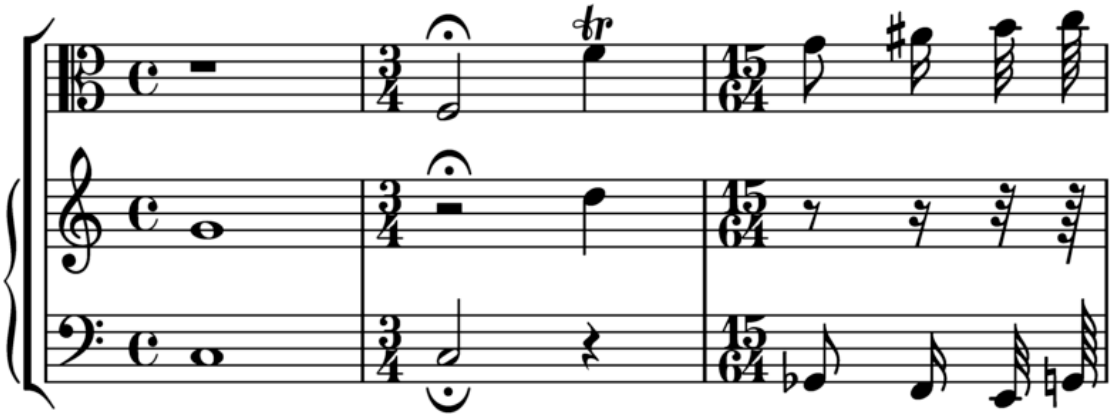
Gonville est une alternative au jeu de glyphes *Feta* inclus dans la fonte Emmentaler que LilyPond utilise par défaut. Vous pouvez la télécharger à partir de

`http://www.chiark.greenend.org.uk/~sgtatham/gonville/` (`https://www.chiark.greenend.org.uk/~sgtatham/gonville/`)

Voici quelques mesures utilisant la police Gonville :



Et ces même mesures avec les glyphes *Feta* de LilyPond :



Instructions d'installation

- Téléchargez puis extrayez les fichiers de fonte.
- Copiez[1] les fichiers

```
gonville-11.otf
gonville-13.otf
gonville-14.otf
gonville-16.otf
gonville-18.otf
gonville-20.otf
gonville-23.otf
gonville-26.otf
gonville-brace.otf
```

  dans le dossier `.../share/lilypond/X.Y.Z/fonts/otf`.
- Si vous disposez des fichiers `gonville-*.svg`, copiez les dans `.../share/lilypond/X.Y.Z/fonts/svg`.

Pour de plus amples informations, reportez-vous à Section "Autres sources de documentation" dans *Manuel d'initiation*.

Il est à noter que les fichiers `gonville-*.otf` sont destinés aux moteurs `ps` et `cairo` (pour obtenir des fichiers PDF ou PostScript, ainsi que tous les formats de sortie pour le moteur Cairo) ; les fichiers `gonville-*.svg` sont destinés au moteur `svg`. Pour de plus amples informations, consultez Voir Section "Options avancées de `lilypond`" dans *Utilisation des programmes*.

La syntaxe suivante substitue aux fontes musicales les fontes Gonville.

```
\paper {
  property-defaults.fonts.music = "gonville"
}
```

Pour plus d'informations, voir Section 8.3.5 [Choix des fontes], page 351.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section 8.3.5 [Choix des fontes], page 351, Section B.8 [La fonte Emmentaler], page 935.

Problèmes connus et avertissements

Gonville ne permet pas de générer de la notation ancienne, et certains glyphes ajoutés depuis lors aux jeux de caractères en sont absent. Consultez le site de l'auteur pour de plus amples informations ainsi qu'à propos des conditions d'utilisation.

Autres fontes musicales

Si vous disposez d'autres fontes musicales telles que *nomfonte*`-*.otf` ou *nomfonte*`-*.svg`, vous pouvez les utiliser de façon comparable à Gonville.

Autrement dit, copiez les fichiers *nomfonte*`-*.otf` dans `.../share/lilypond/X.Y.Z/fonts/otf`. Si vous disposez de fichiers *nomfonte*`-*.svg`, copiez les dans `.../share/lilypond/X.Y.Z/fonts/svg`.

Note : à ce jour, et pour fonctionner correctement, LilyPond requiert la présence des suffixes suivants dans les dossiers d'installation : `-11`, `-13`, `-14`, `-16`, `-18`, `-20`, `-23`, `-26` et `-brace`. Par exemple, `emmentaler-11.otf`, `emmentaler-20.svg`, etc.

[1] Il est pour l'instant nécessaire de répéter ces opération à chaque installation d'une nouvelle version de LilyPond. Si vous lancez le binaire `lilypond` directement du dossier de construction, consultez Section "Replacing the notation fonts in development versions" dans *Guide du contributeur* pour de plus amples informations.

La syntaxe suivante substitue aux fontes musicales les fontes *nomfonte*.

```
\paper {
  % fichier de fonte sans suffixe ni extension
  property-defaults.fonts.music = "nomfonte"
}
```

24 Génération de fichiers MIDI

LilyPond peut produire des fichiers conformes au standard MIDI (Musical Instrument Digital Interface), ce qui permet de vérifier le rendu à l'oreille grâce à un logiciel ou un périphérique sachant interpréter le MIDI. L'écoute du rendu en MIDI permet de contrôler aisément ce que vous avez saisi : octaves et altérations erronées heurteront votre oreille avertie !

Les fichiers MIDI, contrairement aux fichiers AAC, MP3 ou Vorbis, ne contiennent pas de son et nécessitent donc le recours à un logiciel supplémentaire pour les écouter.

24.1 Notation prise en compte dans le MIDI

LilyPond retranscrit par défaut dans un fichier MIDI les éléments de notation suivants :

- les marques de respiration,
- les accords nommés,
- les crescendos et decrescendos s'étendant sur plusieurs notes – le volume s'ajuste linéairement entre les deux extrêmes,
- les indications de nuance, de `ppppp` à `fffff`, y compris `mp`, `mf` et `sf`,
- les paroles,
- les marques : indications de repère, segnos, indications de coda et libellés de section,
- les microtonalités, mais *pas* sous forme d'accord ; leur rendu nécessite cependant un lecteur qui prenne en charge la modulation,
- les hauteurs,
- le rythme sous forme de durée de note, y compris les n-olets,
- les articulations « simples » comme staccato, staccatissimo, accent, marcato et portato,
- les changements de tempo indiqués par un `\tempo`, y compris des valeurs métronomique sous forme de fraction,
- les liaisons de tenue,
- les tremolos, excepté ceux utilisant la syntaxe « :[*nombre*] ».

Spatialisation, balance, expression, réverbération et chorus peuvent se contrôler à l'aide de propriétés de contexte – voir Section 24.8 [Propriétés de contextes et effets MIDI], page 682.

En combinaison avec le script `articulate`, d'autres éléments seront aussi reportés en MIDI :

- les appoggiatures – celles-ci prendront la moitié de la valeur, dépourvue de point, de la note qui les suit – par exemple,

  ```
  \appoggiatura c8 d2.
  ```

  le do (noté `c`) prendra la valeur d'une noire.
- les ornements (mordants, trilles et groupettos, etc.),
- rallentando, accelerando, ritardando et a tempo,
- les liaisons y compris de phrasé,
- les tenutos.

Voir Section 24.9 [Amélioration du rendu MIDI], page 683.

24.2 Notation non prise en compte dans le MIDI

Certains éléments de notation ne peuvent être retranscrits dans un fichier MIDI :

- les articulations autres que staccato, staccatissimo, accent, marcato et portato,
- les crescendos et decrescendos sur *une seule* note,
- les points d'orgue,
- la basse chiffrée,
- les glissandos,
- les chutes ou sauts,
- les accords en microtonalité,
- le rythme indiqué sous forme d'annotation, comme « swing »,
- les changements de tempo indiqués sous forme d'annotation (sans `\tempo`),
- les trémolos indiqués par la syntaxe « :[*nombre*] ».

24.3 Le bloc MIDI

LilyPond générera un fichier MIDI dès que vous ajouterez un bloc `\midi`, même vide, au sein du bloc `\score`[1] :

```
\score {
  ...musique...
  \layout { }
  \midi { }
}
```

> **Note :** Lorsque le bloc `\score` contient uniquement un bloc `\midi` (autrement dit pas de bloc `\layout`), LilyPond produira uniquement la sortie MIDI – aucun support visuel ne sera généré.
>
> Un bloc `\midi` en début de fichier permet d'effectuer des réglages MIDI de manière globale. Bien entendu, la génération d'un fichier MIDI ne sera effective qu'en présence d'une section `\midi` au sein d'un bloc `\score`.
>
> De façon similaire, un bloc `\layout` en début de fichier affecte la mise en forme de manière globale mais n'est d'aucune influence sur la production ou non d'un fichier imprimable.

L'extension par défaut des fichiers MIDI générés (`.midi`) peut se modifier en ligne de commande :

```
lilypond -dmidi-extension=mid MonFichier.ly
```

Une autre manière de procéder consiste à placer la ligne suivante au début de votre fichier source, avant l'ouverture de tout bloc `\book`, `\bookpart` ou `\score` – voir Section 20.5 [Structure de fichier], page 612 :

```
#(ly:set-option 'midi-extension "mid")
```

Voir aussi

Manuel de notation : Section 20.5 [Structure de fichier], page 612.

Fichiers d'initialisation : `scm/midi.scm`.

[1] Il existe aussi une commande de *markup* `\score`, qui ne produit pas de MIDI, même en présence d'un bloc `\midi` – voir [Scores within markup], page 888.

Problèmes connus et avertissements

Le standard MIDI dispose de 15 canaux plus un (le numéro 10) affecté aux percussions. Les portées sont assignées l'une après l'autre à un canal. Dans la mesure où une partition comporte plus de 15 portées, les portées au-delà de la quinzième partageront un même canal MIDI, sans toutefois l'écraser. Ceci peut entraîner des conflits au niveau des canaux en raison des propriétés MIDI, notamment l'instrument utilisé.

24.4 Gestion des nuances en MIDI

Le volume général de la sortie MIDI peut se définir, ainsi que ses modulations, en fonction des indications de nuance et les volumes relatifs entre les différents instruments.

Les indications de nuance se traduisent automatiquement en niveau de volume dans l'amplitude disponible en MIDI alors que crescendos et diminuendos auront une progression linéaire entre les extrêmes.

24.4.1 Indication des nuances en MIDI

Les indications de nuance, de ppppp à `fffff` – y compris `mp`, `mf` et `sf` – ont des valeurs prédéfinies. Ce coefficient est alors appliqué pour corriger le volume général de façon à obtenir le niveau sonore qui sera retranscrit dans le fichier de sortie pour la nuance considérée. Nous allons, par défaut, de 0,25 pour un *ppppp* à 0,95 pour un *fffff*. Les correspondances entre nuance et fraction de volume sont répertoriées dans le fichier `ly/midi-init.ly`.

Morceaux choisis

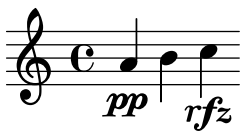
Création de nuance particulière pour la sortie MIDI

L'exemple suivant illustre la manière de créer une indication de nuance, absente de la liste par défaut, et de lui assigner une valeur spécifique utile à la sortie MIDI.

L'indication de nuance `\rfz` (*rinforzando*) se voit attribuer une valeur de `0.9`.

```
#(define (myDynamics dynamic)
    (if (equal? dynamic "rfz")
      0.9
      (default-dynamic-absolute-volume dynamic)))

\score {
  \new Staff {
    \set Staff.midiInstrument = "cello"
    \set Score.dynamicAbsoluteVolumeFunction = #myDynamics
    \new Voice {
      \relative {
        a'4\pp b c-\rfz
      }
    }
  }
  \layout {}
  \midi {}
}
```



Voir aussi

Fichiers d'initialisation : `ly/script-init.ly`, `scm/midi.scm`.

Morceaux choisis : Section "MIDI" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

24.4.2 Réglage du volume en MIDI

Les valeurs extrêmes du volume MIDI des nuances se contrôlent à l'aide des propriétés `midiMinimumVolume` et `midiMaximumVolume` qui agissent au niveau `Score`. Ces propriétés sont effectives dès lors qu'une nuance est indiquée ; une nuance de départ explicite est donc requise pour que le volume soit ajusté dès le début de la partition. Vous pouvez alors modifier la fraction correspondant à chaque nuance à l'aide de la formule

```
midiMinimumVolume + (midiMaximumVolume - midiMinimumVolume) * fraction
```

Voici comment ajuster les nuances tout en limitant l'amplitude du volume entre 0,2 et 0,5 :

```
\score {
  <<
    \new Staff {
      \set Staff.midiInstrument = "flute"
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinet"
      ... music ...
    }
  >>
  \midi {
    \context {
      \Score
      midiMinimumVolume = 0.2
      midiMaximumVolume = 0.5
    }
  }
}
```

La définition de l'amplitude du volume MIDI au niveau d'un contexte `Staff` – grâce aux propriétés `midiMinimumVolume` et `midiMaximumVolume` – permet en quelque sorte d'égaliser un instrument MIDI.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "flute"
    \set Staff.midiMinimumVolume = 0.7
    \set Staff.midiMaximumVolume = 0.9
    ... musique ...
  }
  \midi { }
}
```

Dans le cas d'une partition à plusieurs portées et différents instruments, les volumes relatifs entre les différents instruments se gèrent individuellement :

```
\score {
  <<
```

```
    \new Staff {
      \set Staff.midiInstrument = "flute"
      \set Staff.midiMinimumVolume = 0.7
      \set Staff.midiMaximumVolume = 0.9
      ... music ...
    }
    \new Staff {
      \set Staff.midiInstrument = "clarinet"
      \set Staff.midiMinimumVolume = 0.3
      \set Staff.midiMaximumVolume = 0.6
      ... music ...
    }
  >>
  \midi { }
}
```

La clarinette de cet exemple jouera relativement moins fort que la flûte.

En l'absence de tout réglage des propriétés de volume, LilyPond appliquera cependant un léger degré d'égalisation pour certains instruments – voir `scm/midi.scm`.

Morceaux choisis

Réglage de l'égalisation par défaut des instruments MIDI

L'égaliseur basique peut être modifié par la définition d'une nouvelle procédure Scheme `instrumentEqualizer` au sein du contexte `Score`. Cette procédure prend en unique argument le nom d'un instrument MIDI et renverra une paire de fractions correspondant aux minimum et maximum de volume alloué à cet instrument.

Dans l'exemple suivant sont réglés les volumes relatifs de la flûte et de la clarinette.
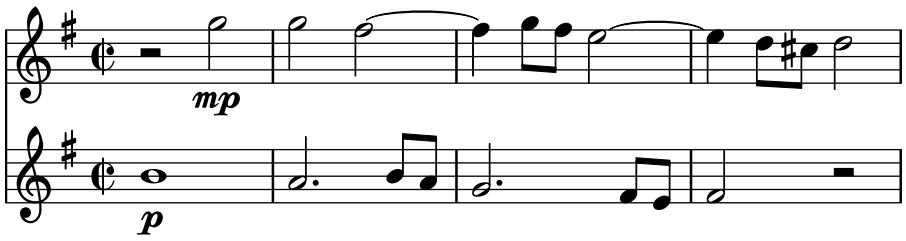
```
#(define my-instrument-equalizer-alist '())

#(set! my-instrument-equalizer-alist
  (append
    '(("flute" . (0.7 . 0.9))
      ("clarinet" . (0.3 . 0.6)))
    my-instrument-equalizer-alist))

#(define (my-instrument-equalizer s)
  (let ((entry (assoc s my-instrument-equalizer-alist)))
    (if entry
      (cdr entry))))

\score {
  <<
    \new Staff {
      \key g \major
      \time 2/2
      \set Score.instrumentEqualizer = #my-instrument-equalizer
      \set Staff.midiInstrument = "flute"
      \new Voice \relative {
        r2 g''\mp g fis~
        4 g8 fis e2~
        4 d8 cis d2
```

```
      }
    }
    \new Staff {
      \key g \major
      \set Staff.midiInstrument = "clarinet"
      \new Voice \relative {
        b'1\p a2. b8 a
        g2. fis8 e
        fis2 r
      }
    }
  >>
  \layout { }
  \midi {  }
}
```



Voir aussi

Fichiers d'initialisation : `scm/midi.scm`.

Manuel de notation : Chapitre 27 [Mise en forme de la partition], page 705.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les modifications apportées au volume MIDI n'interviennent que sur l'attaque d'une note, en conséquence de quoi crescendos et decrescendos n'affecteront pas le volume s'ils se produisent sur une même et unique note.

24.4.3 Réglage de propriétés dans le bloc MIDI

Le bloc `\midi` peut contenir des aménagements pour certains contextes, la définition de contextes particuliers ou du code permettant de déterminer la valeur de certaines propriétés.

```
\score {
  ... music ...
  \midi {
    \tempo 4 = 72
  }
}
```

Le tempo est ici réglé à 72 noires par minute. Une indication de tempo inscrite dans le bloc `\midi` ne sera pas reportée sur la partition imprimable. Cependant, tout `\tempo` mentionné dans le bloc `\score` sera répercuté dans la sortie MIDI.

Placée au sein d'un bloc `\midi`, la commande `\tempo` détermine des propriétés lors de la phase d'interprétation de la musique et dans le contexte de définition des sorties. Elle est alors considérée comme une modification de contexte.

La syntaxe permettant de définir un contexte pour le `\midi` est en tout point identique à celle que vous utilisez dans le bloc `\layout` :

```
\score {
  ... musique ...
  \midi {
    \context {
      \Voice
      \remove Dynamic_performer
    }
  }
}
```

Ces quelques lignes ont pour effet de supprimer l'application des nuances à la sortie MIDI. Vous aurez noté que les modules de traduction de LilyPond en matière de son s'appellent *performers* – des « interprètes ».

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Chapitre 27 [Mise en forme de la partition], page 705, Chapitre 3 [Signes d'interprétation], page 158.

Fichiers d'initialisation : `ly/performer-init.ly`.

Morceaux choisis : Section "MIDI" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Certains lecteurs MIDI ne rendent pas correctement les changements de tempo.

Les modifications de `midiInstrument` ou autres options MIDI en début de portée peuvent se retrouver dédoublées dans la sortie MIDI.

24.5 Gestion des instruments MIDI

L'instrument MIDI est déterminé par la propriété `midiInstrument`, au sein d'un contexte `Staff`.

```
\score {
  \new Staff {
    \set Staff.midiInstrument = "glockenspiel"
    ... music ...
  }
  \midi { }
}
```

ou

```
\score {
  \new Staff \with {midiInstrument = "cello"} {
    ... music ...
  }
  \midi { }
}
```

Lorsque l'instrument choisi ne correspond pas exactement à l'une des dénominations consacrées, LilyPond le remplacera par un piano de concert (`"acoustic grand"`) – voir Section B.6 [Instruments MIDI], page 929.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Section B.6 [Instruments MIDI], page 929, Chapitre 27 [Mise en forme de la partition], page 705.

Référence des propriétés internes : Section "Dynamic_performer" dans *Référence des propriétés internes*.

Fichiers d'initialisation : `scm/midi.scm`.

Problèmes connus et avertissements

Les percussions gérées par un contexte `DrumStaff` sont affectées directement au canal 10 qui leur est réservé. Certains instruments, tels le xylophone, le marimba, le vibraphone ou les timbales, se traitent cependant comme des instruments « classiques » puisqu'ils sont capables d'émettre des hauteurs différentes ; leur notation relève donc d'un contexte `Staff` standard, et non d'un `DrumStaff` pour pouvoir être rendus correctement en MIDI. Une liste complète des percussions affectées au canal 10 (`channel 10 drum-kits`) est disponible dans le fichier `scm/midi.scm` – voir Section "Autres sources de documentation" dans *Manuel d'initiation*.

24.6 Gestion des répétitions en MIDI

Les reprises de toutes sortes peuvent être rendues dans le fichier MIDI. Il suffit pour cela de recourir à la fonction `\unfoldRepeats`, qui développe toutes les reprises.

```
\score {
  \unfoldRepeats {
    \repeat tremolo 8 { c'32 e' }
    \repeat percent 2 { c''8 d'' }
    \repeat volta 2 { c'4 d' e' f' }
    \alternative {
      \volta 1 { g' a' a' g' }
      \volta 2 { f' e' d' c' }
    }
  }
  \midi { }
}
```

Lorsque l'on veut utiliser `\unfoldRepeats` seulement pour le rendu MIDI, il faut établir **deux** blocs `\score` : un pour le MIDI, avec des reprises explicites, et l'autre pour la partition, avec des reprises notées sous forme de barres de reprise, de trémolo ou de symboles de pourcentage. Par exemple

```
\score {
  ... musique ...
  \layout { }
}
\score {
  \unfoldRepeats ... musique ...
  \midi { }
}
```

Dans une partition comportant plusieurs voix, le développement des reprises ne sera effectif en MIDI qu'à la condition que ces reprises soient mentionnées correctement dans **toutes** les voix.

Voir aussi

Manuel de notation : Chapitre 4 [Répétitions et reprises], page 193.

24.7 Affectation des canaux MIDI

Lorsque LilyPond génère un fichier MIDI à partir d'une partition, chaque note contenue dans cette partition sera automatiquement assignée à un canal MIDI, celui sur lequel elle devrait être jouée quand elle est transmise à un périphérique MIDI. Chaque canal MIDI dispose d'un certain nombre de contrôles pour, par exemple, sélectionner l'instrument qui jouera les notes de ce canal ou bien demander au périphérique MIDI d'appliquer différents effets au son produit sur ce canal. En tout état de cause, chaque contrôle d'un canal MIDI ne peut se voir affecté que d'une seule valeur à la fois – celle-ci peut toutefois être modifiée pour, par exemple, changer d'instrument au milieu du morceau.

Le standard MIDI ne dispose que de 16 canaux par périphérique MIDI. Cette limite du nombre de canaux entraîne une limitation du nombre d'instruments pouvant jouer de concert.

LilyPond crée une piste MIDI séparée pour chaque portée (ou chaque instrument ou voix selon la valeur de `Score.midiChannelMapping`) ainsi que pour chaque contexte de paroles. Il n'y a pas de limite au nombre de pistes.

Afin de contourner la limitation du nombre de canaux MIDI, LilyPond dispose de différents modes d'allocation d'un canal MIDI grâce à la propriété de contexte `Score.midiChannelMapping`. Dans tous les cas, lorsque la limite au nombre de canaux est atteinte, LilyPond repart du canal 0, ce qui peut affecter des notes au mauvais instrument. Cette propriété de contexte peut prendre les valeurs suivantes :

`'staff`

Allocation d'un canal MIDI particulier à chacune des portées de la partition (option par défaut). Toutes les notes de toutes les voix d'une même portée partageront le canal MIDI affecté à la portée qui les englobe, et toutes seront encodées dans la même piste.

La limite des 16 canaux s'applique au nombre total de portées augmenté des contextes de paroles même si les paroles MIDI n'occupent pas de canal MIDI.

`'instrument`

Allocation d'un canal MIDI particulier à chaque instrument MIDI tel que spécifié dans la partition. En d'autre termes, des notes jouées par un même instrument MIDI partageront le même canal MIDI (et la même piste), même si elles proviennent de voix ou portées différentes.

Dans ce cas particulier, les contextes de paroles ne sont pas pris en compte dans la limite des 16 canaux, puisqu'ils ne sont pas assignés à un instrument MIDI, ce qui permet une meilleure allocation des canaux MIDI lorsque le nombre de portées et de contextes de paroles dépasse 16.

`'voice`

Allocation d'un canal MIDI particulier à chaque voix de la partition portant un nom unique parmi les voix de la portée considérée. Des voix appartenant à des portées différentes seront toujours affectées à des canaux MIDI différents, mais deux voix partageant une même portée partageront le même canal MIDI dès lors qu'elles porteront le même nom. Dans la mesure où `midiInstrument` et les différents contrôles d'effets MIDI sont des propriétés affectant le contexte de portée, ils ne peuvent se déterminer individuellement pour une voix. La première voix adoptera l'instrument et les effets spécifiés pour cette portée, et les voix dénommées différemment de la première se verront attribué l'instrument et le effets par défaut.

Note : l'affectation d'instruments ou d'effets différents aux différentes voix d'une même portée s'obtient dès lors que le `Staff_performer` est déplacé du contexte `Staff` au contexte `Voice` tout en maintenant le `midiChannelMapping` dans le contexte `'staff` ou en le réglant sur `'instrument`.

Par exemple, l'affectation par défaut des canaux MIDI d'une partition peut être réglée sur `'instrument` comme ceci :

```
\score {
  ...musique...
  \midi {
    \context {
      \Score
      midiChannelMapping = #'instrument
    }
  }
}
```

Morceaux choisis
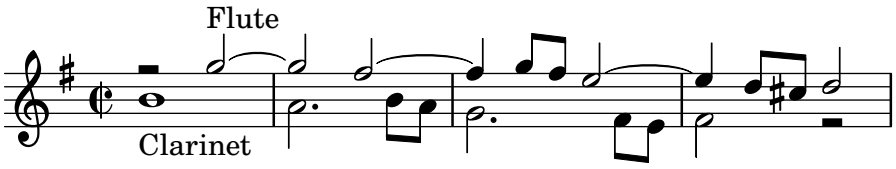
Affectation d'un canal MIDI par voix

Lorsque LilyPond génère un fichier MIDI, chaque portée sera par défaut affectée à un canal, quel que soit le nombre de voix qu'elle contient. Ceci permet d'éviter de se retrouver à court de canaux, sachant qu'il n'y en a que seize de disponibles par piste.

Le fait de déplacer le `Staff_performer` dans le contexte `Voice` permet d'affecter à chaque voix d'une même portée un canal MIDI spécifique. Dans l'exemple suivant, la même portée donnera lieu à deux canaux MIDI différents, chacun étant affecté de son propre `midiInstrument`.

```
\score {
  \new Staff <<
    \new Voice \relative c''' {
      \set midiInstrument = "flute"
      \voiceOne
      \key g \major
      \time 2/2
      r2 g-"Flute" ~
      g fis ~
      fis4 g8 fis e2 ~
      e4 d8 cis d2
    }
    \new Voice \relative c'' {
      \set midiInstrument = "clarinet"
      \voiceTwo
      b1-"Clarinet"
      a2. b8 a
      g2. fis8 e
      fis2 r
    }
  >>
  \layout { }
  \midi {
    \context {
      \Staff
      \remove "Staff_performer"
    }
    \context {
      \Voice
      \consists "Staff_performer"
```

```
      }
      \tempo 2 = 72
    }
  }
```



24.8 Propriétés de contextes et effets MIDI

Les différentes propriétés de contexte qui suivent permettent d'appliquer différents effets MIDI aux notes contenues dans le canal MIDI associé à la portée courante, à l'instrument ou à la voix, selon la valeur affectée à la propriété de contexte `Score.midiChannelMapping` et le contexte dans lequel le `Staff_performer` réside – voir Section 24.7 [Affectation des canaux MIDI], page 680.

Une adaptation de ces propriétés de contexte affectera toutes les notes jouées sur ce canal dès leur modification. Certains effets pourront même s'appliquer sur des notes déjà en cours, selon l'implémentation du périphérique de sortie MIDI.

LilyPond dispose des propriétés de contexte suivantes :

`Staff.midiPanPosition`
: La spatialisation (*pan position*) contrôle le positionnement d'un canal MIDI entre les sorties stéréo droite et gauche. Cette propriété de contexte prend en argument une valeur entre -1.0 (`#LEFT`) et 1.0 (`#RIGHT`). Une valeur de -1.0 enverra toute la puissance sonore sur le haut-parleur gauche (le droit sera silencieux), une valeur de 0.0 (`#CENTER`) distribuera équitablement le son entre les haut-parleurs de gauche et de droite, et une valeur de 1.0 enverra tout le son sur le haut-parleur de droite. Des valeurs entre -1.0 et 1.0 permettent d'obtenir une répartition du son entre les sorties gauche et droite d'un équipement stéréophonique.

`Staff.midiBalance`
: La balance stéréo d'un canal MIDI. Tout comme la spatialisation, cette propriété de contexte prend en argument une valeur comprise entre -1.0 (`#LEFT`) et 1.0 (`#RIGHT`). Elle permet de faire varier le volume relatif envoyé aux deux haut-parleurs stéréo sans pour autant affecter la distribution des signaux stéréo.

`Staff.midiExpression`
: Le niveau d'expression, en tant que fraction du niveau maximum de volume disponible, à appliquer à un canal MIDI. Un périphérique MIDI combine le niveau d'expression des canaux MIDI et le niveau de nuance de la voix en cours (tel que défini par `\p` ou `\ff`) afin d'obtenir le volume total de chacune des notes de la voix. Un contrôle de l'expression permet, par exemple, d'implémenter des effets de crescendo ou decrescendo sur une note tenue, ce que LilyPond ne sait pas faire automatiquement.

  Le niveau d'expression varie entre 0.0 (sans expression, autrement dit volume à zéro) et 1.0 (volume au maximum).

`Staff.midiReverbLevel`
: Le niveau de réverbération, en tant que fraction du niveau maximum disponible, à appliquer à un canal MIDI. Cette propriété prend en argument une valeur entre 0.0 (pas d'écho) et 1.0 (effet maximal).

`Staff.midiChorusLevel`
: Le niveau de chœur, en tant que fraction du niveau maximum disponible, à appliquer à un canal MIDI. Cette propriété prend en argument une valeur entre 0.0 (pas de chorus) et 1.0 (effet maximal).

Problèmes connus et avertissements

Dans la mesure où les fichiers MIDI ne comportent effectivement aucune donnée audio, les modifications des propriétés de contexte ne se traduisent qu'en requêtes de changement des contrôles du canal MIDI lorsque ces fichiers MIDI sont joués. La manière dont un périphérique MIDI particulier, tel un synthétiseur MIDI logiciel, gérera ces requêtes incluses dans un fichier MIDI dépend complètement de l'implémentation du périphérique : certains d'entre eux pourront simplement ignorer plusieurs, voire toutes ces requêtes. Par ailleurs, la manière dont un périphérique MIDI interprète les différentes valeurs de ces contrôles (en règle générale, le standard MIDI ne fixe le comportement qu'aux valeurs extrêmes de l'amplitude disponible pour chacun des contrôles) et leur modification alors qu'une note sur un canal est tenue, dépend de l'implémentation particulière à ce périphérique.

Lors de la génération d'un fichier MIDI, LilyPond transforme simplement les valeurs fractionnaires dans l'amplitude linéaire en valeurs entières correspondantes (de 0 à 127 et sur 7 bits, ou de 0 à 16383 et sur 14 bits pour les contrôles MIDI supportant une résolution fine). Ces valeurs entières converties sont stockées telles quelles dans le fichier MIDI généré. Pour plus d'information sur la manière dont un périphérique MIDI interprète ces valeurs, se reporter à sa documentation.

24.9 Amélioration du rendu MIDI

Le fichier MIDI généré par LilyPond est relativement brut. Il peut toutefois être amélioré en affectant des instruments MIDI ou en réglant certaines propriétés au sein du bloc `\midi`.

Des scripts additionnels permettent d'affiner la manière dont les nuances, articulations et rythme sont rendus en MIDI : le script `articulate` et le script `swing`.

24.9.1 Le script `articulate`

L'utilisation du script `articulate` se fait après avoir ajouté en tête de fichier la commande `\include` appropriée :

```
\include "articulate.ly"
```

Le script créera une sortie MIDI dont les notes seront échelonnées de sorte à tenir compte de toute articulation ou changement de tempo. La sortie imprimable sera toutefois modifiée en profondeur, pour refléter littéralement la sortie MIDI.

```
\score {
  \articulate <<
    ... musique ...
  >>
  \midi { }
}
```

Le script `\articulate` tient compte des abréviations telles que les trilles ou groupettos. L'intégralité des éléments traités est répertoriée dans le script lui-même – voir `ly/articulate.ly`.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*.

Manuel de notation : Chapitre 27 [Mise en forme de la partition], page 705.

Fichiers d'initialisation : `ly/articulate.ly`.

> **Note :** Dans la mesure où le script `articulate` tend à raccourcir les accords, certaines musiques, notamment pour l'orgue, paraîtront de moins bonne qualité. Les notes dépourvues d'articulation peuvent aussi se voir raccourcies ; pour pallier cet inconvénient, le recours à la fonction `\articulate` devrait ne concerner que de courts fragments, sauf à modifier les valeurs des variables contenues dans le script `articulate`.

24.9.2 Le script `swing`

Le script `swing` procure des fonctions additionnelles qui permettent de jouer des durées égales sur un rythme inégal. L'exemple le plus évident est l'interprétation *swinguée* que l'on trouve en jazz où des croches binaires devraient se jouer de façon ternaire. D'autres interprétations sont toutefois prises en charge.

Ce script doit faire l'objet d'une inclusion en début de fichier source :

```
\include "swing.ly"
```

Le script fournit trois commandes :

- `\tripletFeel` crée un *swing* sur une base de triolet. Elle prend deux arguments : les durées à affecter (typiquement `8` pour des croches) et l'expression musicale sur laquelle l'appliquer.
- `\applySwing` prend un argument supplémentaire avant l'expression musicale : une *liste de pondération* de n nombres de ratio exprimant la manière dont doivent être jouées les notes régulières. Par exemple, `#'(2 1)` indique que chaque note devrait se jouer deux fois plus longue que la suivante – en fait, `\tripletFeel` *durée* {*musique*} est un raccourci de `\applySwing` *durée* `#'(2 1)` {*musique*} Des croches chaloupées plus doucement s'obtiennent avec une *liste de pondération* de `#'(3 2)` ou toute autre valeur selon les goûts.

  Cette liste peut prendre plus de deux valeurs, ce qui permet d'adopter un schéma de *groove* plus long ou plus sophistiqué. Par exemple, une impression de samba sur des double-croches peut s'obtenir ainsi :

```
\score {
  \applySwing 16 #'(3 2 2 3) {
    ... music ...
  }
  \midi { }
}
```

- `\applySwingWithOffset` ajoute quant à elle un autre argument entre la *liste de pondération* et l'expression musicale : une durée de décalage. Cette commande devrait s'utiliser lorsque l'expression musicale démarre à contre temps, avec une portion de cycle.

> **Note :** Tout comme avec le script `articulate`, toutes les commandes seront rendues dans la partition imprimable, ce qui résulte en un espacement irrégulier. Ceci peut s'éviter en utilisant le script dans un bloc `\score` dédié à la sortie MIDI.

Une aide et des informations supplémentaires sont directement incluses dans le script – voir `ly/swing.ly`.

Voir aussi

Manuel d'initiation : Section “Autres sources de documentation” dans *Manuel d'initiation*.

Manuel de notation : Chapitre 2 [Rythme], page 53.

Fichiers d'initialisation : `ly/swing.ly`.

Problèmes connus et avertissements

- Les constructions `\repeat` et `\repeat unfold` ne sont pas prises en considération lors de la temporisation des notes. Ceci entraînera des problèmes à moins que les durées de tous les fragments répétés soient des multiples entiers du cycle de *swing*.
- Ces fonctions sont agnostiques en matière de métrique et de mesure, raison pour laquelle des décalages doivent être fournis à l'aide de `\applySwingWithOffset` lorsque la musique démarre sur une levée.
- Les appoggiatures sont ignorées et laissées telles quelles, tout comme les triolets.

25 Extraction d'informations musicales

En plus de générer du graphisme et du MIDI, LilyPond peut présenter l'information musicale sous forme textuelle.

25.1 Affichage de notation au format LilyPond

La fonction musicale `\displayLilyMusic` permet d'afficher en notation LilyPond une expression musicale. Le résultat défilera dans le terminal après avoir lancé LilyPond en ligne de commande. Par exemple,

```
{
  \displayLilyMusic \transpose c a, { c4 e g a bes }
}
```

affichera

```
{ a,4 cis4 e4 fis4 g4 }
```

LilyPond affichera le résultat sous forme de message en console, au milieu de toutes les informations de compilation. Afin d'isoler ces messages et enregistrer le résultat de la fonction `\displayLilyMusic`, pensez à rediriger la sortie vers un fichier.

```
lilypond fichier.ly > affichage.txt
```

Vous noterez que LilyPond ne se contente pas de simplement afficher l'expression musicale, mais procède aussi à son interprétation – du fait que `\displayLilyMusic` renvoie l'expression tout en l'affichant. La simple insertion d'un `\displayLilyMusic` dans une expression musicale permet d'obtenir l'information la concernant.

Si l'instruction `\displayLilyMusic` interprète et affiche des informations sur un fragment, la faire précéder d'un `\void` aura pour effet de l'exclure du fichier résultant.

```
{
  \void \displayLilyMusic \transpose c a, { c4 e g a bes }
  c1
}
```

25.2 Affichage de la musique sous forme d'expression Scheme

Voir Section "Affichage d'expressions musicales" dans *Extension de LilyPond*.

25.3 Enregistrement d'événements musicaux dans un fichier

LilyPond vous permet de sauvegarder dans un fichier séparé, sur la base de la portée, les événements musicaux. Vous devrez pour ce faire inclure dans votre fichier maître un fichier d'initialisation spécifique :

```
\include "event-listener.ly"
```

Pour chaque portée que comporte votre partition, vous obtiendrez un fichier `NOMFICHIER-PORTÉENOMMÉE.notes` ou `NOMFICHIER-unnamed-staff.notes`. Notez bien que si plusieurs portées ne sont pas explicitement nommées, tous leurs événements seront regroupés et mélangés dans le même fichier. Le résultat ressemblera à ceci :

```
0.000   note     57       4   p-c 2 12
0.000   dynamic  f
0.250   note     62       4   p-c 7 12
0.500   note     66       8   p-c 9 12
0.625   note     69       8   p-c 14 12
0.750   rest     4
```

```
0.750   breathe
```

Il s'agit d'un tableau dont les colonnes sont délimitées par une tabulation. Chaque ligne comporte deux champs fixes suivis d'un certain nombre de paramètres optionnels.

temps type ...paramètres...

Ces informations peuvent faire l'objet d'un retraitement par d'autres programmes, comme des scripts python, aux fins de recherche en analyse musicologique ou des expériences à partir du rendu de LilyPond.

Problèmes connus et avertissements

Tous les événements ne sont pas pris en charge par `event-listener.ly`. Il s'agit en premier lieu d'une démonstration, un « proof of concept » du potentiel de LilyPond. Si certains des éléments que vous cherchez à obtenir n'apparaissent pas, recopiez le fichier `event-listener.ly` dans votre répertoire et modifiez-le de telle sorte qu'il travaille selon vos attentes.

Gestion de l'espace

26 Mise en forme de la page

L'agencement général d'une partition dépend de trois facteurs interdépendants : la mise en page, les sauts de ligne et l'espacement. Les choix faits en matière d'espacement détermineront la densité de chacun des systèmes, ce qui influera sur le positionnement des sauts de ligne et, par voie de conséquence, sur le nombre de pages de la partition.

En pratique, cette procédure comporte quatre étapes. Dans un premier temps, des distances élastiques *(springs)* sont déterminées sur la base des durées. Sont alors calculées différentes possibilités de saut de ligne, chacune se voyant attribuer un « coefficient de laideur », puis est estimée la hauteur de chaque système. LilyPond opte enfin pour la combinaison entre sauts de page et de ligne qui offre la meilleure occupation de l'espace, tant horizontalement que verticalement.

Les réglages qui influencent la mise en forme se placent dans deux blocs différents. Le bloc `\paper {...}` étudié à la rubrique Chapitre 26 [Mise en forme de la page], page 691, contient les réglages applicables à toutes les partitions d'une partie ou de l'intégralité d'un ouvrage – tels que format du papier, impression ou non des numéros de page, etc. Quant au bloc `\layout {...}`, qui fait l'objet de la rubrique Chapitre 27 [Mise en forme de la partition], page 705, il détermine la mise en forme de la musique : le nombre de systèmes utilisés, l'espacement des regroupements de portées, etc.

> **Note :** Vous verrez au fil de ce chapitre apparaître certains termes dont la traduction vous semblera assurément erronée. Il n'en est cependant rien : certains termes techniques ont une histoire particulière selon leur langue d'origine. Ainsi le vocable anglais *Ragged* signifie en lambeau, en loques ; dans l'univers typographique, un maître français voit un alignement à gauche – il dira « au fer à gauche » – alors que son homologue anglophone constate un *ragged-right* – donc du vide à droite.

26.1 Le bloc \paper

Des blocs `\paper` peuvent apparaître à trois différents endroits et former ainsi une hiérarchie :

- En début de fichier source, avant même tout bloc `\book`, `\bookpart` ou `\score`.
- Au sein d'un bloc `\book` et indépendamment de tout bloc `\bookpart` ou `\score` qu'il pourrait contenir.
- Au sein d'un bloc `\bookpart`, mais en dehors de tout bloc `\score`.

Un bloc `\paper` ne doit donc en aucun cas prendre place au sein d'un bloc `\score`.

Les valeurs des différents champs seront filtrées en respectant cette hiérarchie ; les valeurs définies le plus haut persisteront à moins d'être remplacées à un niveau hiérarchique inférieur.

Plusieurs blocs `\paper` peuvent cohabiter à un même niveau, notamment en raison de la présence d'inclusion de fichiers. Dans une telle éventualité, les champs sont regroupés par niveau, la dernière valeur rencontrée ayant préséance en cas de doublon.

Peuvent apparaître dans un bloc `\paper` :

- la fonction Scheme `set-paper-size`,
- des variables propres au bloc `\paper` qui viendront adapter la mise en page,
- la définition des différents *markups* qui personnaliseront la mise en forme des en-têtes et pieds de page ainsi que des titrages.

La fonction `set-paper-size` fait l'objet de la rubrique qui suit – Section 26.2 [Format du papier et adaptation automatique], page 693. Les variables du bloc `\paper` chargées de la mise en page sont abordées plus loin dans ce chapitre. Quant aux définitions relatives aux *markups* des en-tête, pied de page et titrage, elles sont étudiées à la rubrique Section 21.2 [Titrages personnalisés], page 623.

La plupart des variables gérant le papier ne sont fonctionnelles que lorsque mentionnées dans un bloc `\paper`. Certaines, qui peuvent toutefois apparaître dans un bloc `\layout`, sont référencées à la rubrique Section 27.1 [Le bloc `\layout`], page 705.

Sauf mention contraire, toutes les variables du bloc `\paper` qui correspondent à des dimensions sont exprimées en millimètre – vous pouvez bien entendu spécifier un autre système de mesure. Voici comment, par exemple, définir la marge haute (`top-margin`) à dix millimètres :

```
\paper {
  top-margin = 10
}
```

Si vous préférez lui affecter une valeur de 0,5 pouce, vous devrez mentionner le suffixe d'unité `\in` :

```
\paper {
  top-margin = 0.5\in
}
```

LilyPond accepte les suffixes d'unité `\mm`, `\cm`, `\in` et `\pt` et `\bp`. Ces unités sont des conversions de millimètres, répertoriées dans le fichier `ly/paper-defaults-init.ly`. Pour plus de lisibilité, et bien que ce ne soit pas techniquement requis, nous vous conseillons d'ajouter `\mm` à votre code lorsque vous travaillez en millimètres.

Vous pouvez aussi définir les valeurs du bloc `\paper` à l'aide de fonctions Scheme ; voici l'équivalent de l'exemple précédent :

```
\paper {
  #(define top-margin (* 0.5 in))
}
```

Enfin, il est tout à fait possible de prédéfinir des variables concernant les sorties imprimables.

```
bigMargin = \paper { top-margin = 10\cm }

\paper {
  \bigMargin
  indent = 0\mm
}
```

Voir aussi

Manuel de notation : Section 26.2 [Format du papier et adaptation automatique], page 693, Section 27.1 [Le bloc `\layout`], page 705, Section 21.2 [Titrages personnalisés], page 623.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

26.2 Format du papier et adaptation automatique

26.2.1 Format du papier

LilyPond génère par défaut, et en l'absence de mention explicite d'un format de papier particulier, un fichier imprimable au format A4. Vous pouvez cependant utiliser un autre format à l'aide des deux fonctions :

```
set-default-paper-size
                #(set-default-paper-size "quarto")
```

qui se place en début de fichier, et

```
set-paper-size
                \paper {
                  #(set-paper-size "tabloid")
                }
```

qui s'inscrit au sein d'un bloc `\paper`.

La seule restriction à l'utilisation isolée de la fonction `set-default-paper-size` est qu'elle doit intervenir avant le premier bloc `\paper`. `set-default-paper-size` fixe le format pour toutes les pages, alors que `set-paper-size` détermine le format des feuilles rattachées à un bloc `\paper` particulier. Ainsi, lorsque le bloc `\paper` se trouve en tête de fichier, le format du papier s'appliquera à toutes les pages ; si `\paper` apparaît dans un bloc `\book`, la taille ne s'appliquera qu'au *book* en question.

À l'intérieur d'un bloc `\paper`, la fonction `set-paper-size` doit intervenir avant toute autre variable. Les raisons à ceci sont abordées à la rubrique Section 26.2.2 [Adaptation automatique au format], page 694.

Différents formats de papier sont définis dans le fichier `scm/paper.scm`. Bien que vous puissiez y ajouter votre propre format, sachez cependant que celui-ci est écrasé à chaque mise à jour de LilyPond. Les différents formats disponibles sont répertoriés à l'annexe Section B.5 [Formats de papier prédéfinis], page 927.

La commande suivante, inscrite dans votre fichier, vous permettra d'ajouter votre format personnalisé à ceux déjà connus, puis d'y faire appel à l'aide des fonctions `set-default-paper-size` et `set-paper-size`. Par exemple :

```
#(set-default-paper-size '(cons (* 100 mm) (* 50 mm)))
```

définit une feuille de largeur 100 mm et 50 mm de hauteur.

Les unités peuvent s'exprimer aussi bien en `in` (pouces), qu'en `cm` (centimètres), `mm` (millimètres), `pt` (point) ou `bp` (*big point*).

Le fait d'ajouter l'argument `'landscape` à l'instruction stipulant le format du papier permet d'obtenir une présentation à l'italienne – ou paysage si vous préférez – et donc des lignes plus longues.

```
#(set-default-paper-size "a6" 'landscape)
```

L'inversion des dimensions du papier sans pour autant basculer la présentation – comme pour imprimer sur une carte postale ou créer un graphique destiné à inclusion – s'obtient en ajoutant '`landscape`' au nom du format de papier :

```
#(set-default-paper-size "a6landscape")
```

Lorsque la taille du papier comporte explicitement '`landscape`' ou '`portrait`', la présence d'un argument `'landscape` aura pour seul effet de modifier l'orientation de l'image et non les dimensions de la feuille.

Voir aussi

Manuel de notation : Section 26.2.2 [Adaptation automatique au format], page 694, Section B.5 [Formats de papier prédéfinis], page 927.

Fichiers d'initialisation : `scm/paper.scm`.

26.2.2 Adaptation automatique au format

Toute modification du format de papier à l'aide des fonctions Scheme `set-default-paper-size` ou `set-paper-size`, que nous avons vues à la rubrique Section 26.2.1 [Format du papier], page 693, se traduira automatiquement par l'ajustement d'un certain nombre de variables attachées au bloc `\paper` afin qu'elles soient en concordance avec le format spécifié. Vous pouvez annuler l'ajustement automatique d'une variable particulière en redéfinissant sa valeur après avoir spécifié le format de papier utilisé. Notez bien que le simple fait d'affecter une valeur à `paper-height` ou `paper-width` ne déclenchera pas l'étalonnage automatique, bien que spécifier une largeur de papier *(*`paper-width`*)* peut influencer d'autres valeurs – mais c'est une autre histoire dont nous parlerons plus tard et qui n'a rien à voir avec la mise à l'échelle.

L'adaptation automatique affecte les dimensionnements verticaux `top-margin` et `bottom-margin` – voir Section 26.3 [Variables d'espacement vertical fixe], page 694, –, ainsi que les dimensionnements horizontaux `left-margin`, `right-margin`, `inner-margin`, `outer-margin`, `binding-offset`, `indent` et `short-indent` – voir Section 26.5 [Variables d'espacement horizontal], page 697.

Les valeurs par défaut de ces dimensionnements sont contenues dans le fichier `ly/paper-defaults-init.ly` et utilisent les variables internes `top-margin-default`, `bottom-margin-default`, etc. correspondant au format par défaut – papier A4 – pour lequel `paper-height` est à `297\mm` et `paper-width` à `210\mm`.

Voir aussi

Manuel de notation : Section 26.5 [Variables d'espacement horizontal], page 697, Section 26.3 [Variables d'espacement vertical fixe], page 694.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`, `scm/paper.scm`.

26.3 Variables d'espacement vertical fixe

> **Note :** Certains dimensionnements attachés au bloc `\paper` sont automatiquement ajustés selon le format du papier, ce qui peut conduire à un résultat inattendu – voir Section 26.2.2 [Adaptation automatique au format], page 694.

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`paper-height`
: La hauteur de la feuille. Il s'agit par défaut de la dimension du papier utilisé. Notez bien que cette variable n'affectera pas l'ajustement automatique d'un certain nombre de dimensionnements verticaux.

`top-margin`
: La marge entre le bord supérieur de la feuille et la surface imprimable. Elle est fixée par défaut à `5\mm` et s'ajustera selon le format de papier.

`bottom-margin`
: La marge entre la surface imprimable et le bord inférieur de la feuille. Elle est fixée par défaut à `6\mm` et s'ajustera selon le format de papier.

`ragged-bottom`
: L'activation de cette variable permet de ne pas répartir verticalement les systèmes sur les pages hormis la dernière. La valeur par défaut est `#f`. Lorsque la partition ne comporte que deux ou trois systèmes par page, comme pour un conducteur d'orchestre, nous vous conseillons d'activer cette variable.

`ragged-last-bottom`
: La désactivation de cette variable permet de répartir verticalement les systèmes de la dernière page d'une partition. La valeur par défaut est `#t`. Nous vous conseillons, lorsque des pièces couvrent deux pages ou plus, de désactiver cette variable.
Notez bien que la variable `ragged-last-bottom` affecte aussi la dernière page de chacune des parties – créées à l'aide d'un bloc `\bookpart` – d'un même ouvrage.

Voir aussi

Manuel de notation : Section 26.2.2 [Adaptation automatique au format], page 694.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Les titrages (contenus dans le bloc `\header{}`) sont considérés comme des systèmes à part entière ; ils seront donc affectés par `ragged-bottom` et `ragged-last-bottom`, qui éventuellement ajouteront de l'espace avant le premier système de la partition.

La définition explicite d'un format de papier annulera tout réglage des marges haute et basse.

26.4 Variables d'espacement vertical fluctuant

Il est souvent judicieux d'apporter un peu de flexibilité à l'espacement entre différents éléments (marges, titres, systèmes ou mouvements), en dilatation ou compression selon le cas. Un certain nombre de variables de type `\paper` répertoriées ci-dessous vous permettront d'affiner ces dimensionnements.

Gardez à l'esprit que les variables du bloc `\paper` dont nous parlons ici n'influencent en rien l'espacement des portées d'un même système. L'espacement au sein des systèmes est géré par des propriétés attachées à des objets graphiques (*grobs*) qui, elles, se définissent au niveau du bloc `\score` – voir à ce sujet Section 29.1 [Espacement vertical au sein d'un système], page 719.

26.4.1 Structure des variables d'espacement vertical fluctuant

Chacune de ces variables attachées au bloc `\paper` est constituée d'une liste associative *(alist)* à quatre *clés* :

- `basic-distance` *(distance de base)* – la grandeur d'espace par défaut, exprimée en hauteur de portée, séparant les *points de référence* de deux éléments, qui évite tout risque de collision en l'absence de dilatation ou compression. Le point de référence d'un titre ou d'un *markup* est son sommet, celui d'un système est le centre vertical du `StaffSymbol` le plus proche – même lorsqu'une ligne de « non-portée » viendrait à s'intercaler. Une `basic-distance` inférieure à `padding` ou `minimum-distance` sera sans effet, dans la mesure où l'espacement résultant ne saurait être inférieur à `padding` ou `minimum-distance`.
- `minimum-distance` *(distance minimale)* – l'espacement minimal, exprimé en hauteur de portée, entre les points de référence des deux éléments alors qu'il y a déjà un effet de compression. Une `minimum-distance` inférieure à la valeur du `padding` sera sans effet, dans la mesure où l'espacement résultant ne saurait être inférieur au `padding`.
- `padding` *(décalage)* – la grandeur minimale de « blanc » qui sépare deux éléments, exprimée en hauteur de portée. On peut le voir comme la hauteur minimale d'un rectangle vide qui devrait s'étendre sur toute la largeur des deux éléments.

- `stretchability` *(dilatation)* – le coefficient d'étirement de cet espace. Un coefficient nul permet de figer l'espacement, à moins qu'il n'en résulte des collisions. Un coefficient positif déterminera la propension d'un espacement à s'étirer, tout en tenant compte du coefficient affecté aux autres espacements. Par exemple, lorsque le coefficient de dilatation d'une dimension est double de celui d'une autre, elle pourra s'étirer deux fois plus que cette dernière. Il ne saurait être négatif. La valeur `+inf.0` provoque une `programming_error` (erreur de programmation) et est ignorée ; vous pouvez toutefois utiliser `1.0e7` pour obtenir une valeur proche de l'infini. Lorsque cette *clé* n'est pas définie, sa valeur est par défaut égale à `space`. Notez bien que l'utilisateur ne peut définir une propension à la compression ; elle est en fait égale à (`basic-distance` − `minimum-distance`).

Lorsque l'impression n'est pas en pleine page – elle est donc *ragged bottom* pour les anglophones – l'élément `space` n'est pas étiré. Les hauteurs sur une telle page correspondront donc au maximum de

- `basic-distance`, plus
- `minimum-distance` et
- `padding`, augmenté de ce qu'il faut pour éviter les chevauchements.

Cependant, lorsque la partition fait plusieurs pages, la dernière page reprendra dans la mesure du possible l'espacement de la page précédente.

Les manières de modifier des listes associatives font l'objet d'un Section 35.8 [Modification de listes associatives], page 798. L'exemple suivant indique deux façons de modifier une liste associative. La première déclaration intervient sur une seule clé, alors que la deuxième redéfinit complètement la variable.

```
\paper {
  system-system-spacing.basic-distance = 8

  score-system-spacing =
    #'((padding . 1)
       (basic-distance . 12)
       (minimum-distance . 6)
       (stretchability . 12))
}
```

26.4.2 Liste des variables d'espacement fluctuant

Le nom des dimensionnements à hauteur variable sont de la forme *avant*`-`*après*`-spacing`, où *avant* et *après* représentent les éléments qui doivent être espacés. La distance s'établit entre les points de référence des deux éléments concernés (voir la rubrique précédente pour plus de précision). Notez bien que, dans les règles de nommage des variables qui suivent, le terme `markup` fait référence aussi bien à un *markup de titrage* (`bookTitleMarkup` ou `scoreTitleMarkup`) qu'à un *markup de haut niveau* (voir Section 20.5 [Structure de fichier], page 612). Toutes les distances sont exprimées en espace de portée.

Leurs valeurs par défaut sont inscrites dans le fichier `ly/paper-defaults-init.ly`.

`markup-system-spacing`
: détermine l'espacement entre un titre ou un *markup* de premier niveau, et le système qui le suit.

`score-markup-spacing`
: détermine l'espacement entre le dernier système et le titre ou *markup* de haut niveau qui le suit.

`score-system-spacing`
: détermine l'espacement entre le dernier système d'une partition et le premier système de la partition suivante, en l'absence de titrage ou *markup* qui les sépare.

`system-system-spacing`
: détermine l'espacement entre deux systèmes d'un même mouvement.

`markup-markup-spacing`
: détermine l'espacement entre deux titres ou *markups* de premier niveau.

`last-bottom-spacing`
: détermine la distance entre le dernier système ou le dernier *markup* de haut niveau, et le bas de la surface imprimable – autrement dit le haut de la marge basse.

`top-system-spacing`
: détermine l'espace entre le haut de la surface imprimable (le bas de la marge haute) et le milieu du premier système. Cette variable n'est effective qu'en l'absence de titre ou *markup* de premier niveau en haut de page.

`top-markup-spacing`
: détermine l'espace entre le haut de la surface imprimable (le bas de la marge haute) et le premier titre ou *markup* de premier niveau. Cette variable n'est effective qu'en l'absence de système en haut de page.

Voir aussi

Manuel de notation : Section 29.1 [Espacement vertical au sein d'un système], page 719.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

26.5 Variables d'espacement horizontal

Note : Certains dimensionnements attachés au bloc `\paper` sont automatiquement ajustés selon le format du papier, ce qui peut conduire à un résultat inattendu – voir Section 26.2.2 [Adaptation automatique au format], page 694.

26.5.1 Variables de marge et de largeur

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`paper-width`
: La largeur de la page. Elle correspond par défaut à la largeur du format de papier utilisé. Si `paper-width` n'a aucun effet en matière d'ajustement automatique, cette variable influe sur la variable `line-width`. Lorsque vous définissez à la fois les valeurs de `paper-width` et `line-width`, les valeurs de `left-margin` et `right-margin` seront recalculées. Voir aussi `check-consistency`.

`line-width`
: La longueur d'une ligne. Lorsque spécifié dans un bloc `\paper`, ceci définit l'étendue horizontale dont disposeront les lignes de portée d'un système non indenté. La valeur par défaut est égale à `paper-width`, auquel sont retranchés `left-margin` et `right-margin`. Lorsque vous définissez `line-width` sans modifier les valeurs de `left-margin` et `right-margin`, les marges seront alors recalculées de telle sorte que les systèmes soient centrés. Voir aussi `check-consistency`.

La valeur de `line-width` peut aussi se spécifier individuellement au niveau de la partition, au sein d'un bloc `\layout`. Ceci permet de contrôler la longueur des lignes partition par partition. Si la longueur de ligne n'est pas spécifiée dans une partition particulière, elle sera valorisée à celle du `line-width` mentionné dans le bloc `\paper`. La détermination de `line-width` pour un `\score` particulier n'a aucun effet sur les marges. Les lignes d'une portée dont la longueur est déterminée par le `line-width` d'une partition seront alignées par la gauche sur la surface de papier telle que définie par le `line-width` du bloc `\paper`. Dès lors que les valeurs de `line-width` de la partition et du papier sont égales, les lignes s'étendront de la marge gauche à la marge droite. En cas de `line-width` de la partition supérieur au `line-width` du papier, les lignes de portée déborderont dans la marge de droite.

`left-margin`
: La marge entre le bord gauche de la feuille et le début de chaque système. La valeur par défaut est de `10\mm` ; elle sera ajustée selon le format du papier. Lorsque vous définissez `line-width` et `right-margin` sans modifier la valeur de `left-margin`, cette dernière sera alors égale à `(paper-width − line-width − right-margin)`. Lorsque seule `line-width` est définie, les deux marges correspondent à `((paper-width − line-width) / 2)`, ce qui a pour effet de centrer les systèmes sur la page. Voir aussi `check-consistency`.

`right-margin`
: La marge entre le bord droit de la page et la fin des systèmes en pleine largeur (non *ragged*). La valeur par défaut est de `10\mm` et s'ajustera selon le format du papier. Lorsque vous définissez `line-width` et `left-margin`, sans modifier la valeur de `right-margin`, cette dernière sera alors égale à `(paper-width − line-width − left-margin)`. Lorsque seule `line-width` est définie, les deux marges correspondent à `((paper-width − line-width) / 2)`, ce qui a pour effet de centrer les systèmes sur la page. Voir aussi `check-consistency`.

`check-consistency`
: Lorsqu'elle est activée, cette variable vérifie que `left-margin`, `line-width` et `right-margin` sont en cohérence, et que l'addition de ces trois éléments ne dépassera pas la largeur du papier (`paper-width`). La valeur par défaut est `#t`. Dans le cas d'une incohérence, un message d'avertissement est émis et les trois variables – marges et longueur de ligne – rétablies à leur valeur par défaut (ajustées selon le format du papier). En cas de désactivation de cette variable (valorisation à `#f`, toute incohérence sera ignorée, et les systèmes pourront déborder de la page.

`ragged-right`
: Lorsque cette variable est activée, les systèmes ne s'étendront pas sur la longueur de la ligne, mais s'arrêteront à leur longueur normale. La valeur par défaut est `#f` mais, si la partition ne comporte qu'un seul système, elle passe à `#t`. Cette variable peut aussi se gérer au sein d'un bloc `\layout`.

`ragged-last`
: Lorsqu'elle est activée, cette variable permet de ne pas étendre le dernier système de façon à occuper toute la longueur de la ligne. La valeur par défaut est `#f`. Cette variable peut aussi se gérer au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : Section 26.2.2 [Adaptation automatique au format], page 694.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Problèmes connus et avertissements

La définition explicite d'un format de papier annulera tout réglage des marges gauche et droite.

26.5.2 Variables spécifiques à l'impression recto-verso

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`two-sided`
: Cette variable permet de gérer efficacement les impressions recto-verso. Lorsqu'elle est activée, les réglages affectés à `inner-margin`, `outer-margin` ainsi que `binding-offset` détermineront les différentes marges selon qu'il s'agit d'une page paire ou impaire. Cette variable s'applique en lieu et place de `left-margin` et `right-margin`. La valeur par défaut est `#f`.

`inner-margin`
: La marge que toutes les pages d'une partie ou de tout un ouvrage devront avoir du côté intérieur. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à vrai. La valeur par défaut est de `10\mm` et s'ajustera selon le format du papier.

`outer-margin`
: la marge que toutes les pages d'une partie ou de tout un ouvrage devront avoir du côté extérieur – opposé à la reliure. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à vrai. La valeur par défaut est de `20\mm` et s'ajustera selon le format du papier.

`binding-offset`
: La gouttière, ou marge de reliure, permet d'augmenter en conséquence la valeur de la marge intérieure `inner-margin` de telle sorte que rien ne soit masqué par la reliure. Bien entendu, cette variable n'est effective que lorsque vous comptez générer un fichier imprimable en recto-verso – propriété `two-sided` définie à `#t`. La valeur par défaut est de `0` et s'ajustera selon le format du papier.

Voir aussi

Manuel de notation : Section 26.2.2 [Adaptation automatique au format], page 694.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

26.5.3 Variables d'indentation et de décalage

Les valeurs par défaut (avant étalonnage) sont définies dans le fichier `ly/paper-defaults-init.ly`.

`horizontal-shift`
: Tous les systèmes, ainsi que les titres et séparateurs de systèmes, seront poussés d'autant vers la droite. La valeur par défaut est de `0.0\mm`.

`indent`
: Le niveau d'indentation du premier système d'une partition. La valeur par défaut est de `15\mm` en A4 et s'ajustera selon le format du papier. L'espace correspondant à `line-width` est réduit d'autant pour le premier système. Cette variable peut aussi se gérer partition par partition au sein d'un bloc `\layout`.

`short-indent`
: Le niveau d'indentation de tous les systèmes hormis le premier. La valeur par défaut est de `0` pour du A4, et s'ajustera selon le format du papier dès lors que vous lui

aurez affecté une valeur. Bien entendu, l'espace spécifié par `line-width` sera réduit d'autant. Cette variable peut aussi se gérer partition par partition, au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : Section 26.2.2 [Adaptation automatique au format], page 694.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

26.6 Autres variables du bloc \paper

26.6.1 Variables de gestion des sauts de ligne

`max-systems-per-page`
: Le nombre maximal de systèmes qu'une page pourra comporter. Cette variable n'est prise en compte, à ce jour, que par l'option `ly:optimal-breaking`, et n'est pas définie.

`min-systems-per-page`
: Le nombre minimal de systèmes qu'une page pourra comporter. Attention cependant aux risques de débordement s'il est trop important. Cette variable n'est prise en compte, à ce jour, que par l'option `ly:optimal-breaking`, et n'est pas définie.

`systems-per-page`
: Le nombre de systèmes que devrait comporter chaque page. Cette variable n'est à ce jour prise en charge que par l'algorithme `ly:optimal-breaking` et n'est pas définie par défaut.

`system-count`
: Le nombre de systèmes requis par la partition. Cette variable n'est pas définie par défaut. Cette variable peut se gérer au sein d'un bloc `\layout`.

Voir aussi

Manuel de notation : Section 28.1 [Sauts de ligne], page 711.

26.6.2 Variables de gestion des sauts de page

Les valeurs par défaut sont définies dans le fichier `ly/paper-defaults-init.ly`.

`page-breaking`
: L'algorithme de calcul des sauts de page à utiliser. Vous avez le choix entre `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-page-breaking`, `ly:one-line-breaking`, `ly:one-line-auto-height-breaking`, et `ly:optimal-breaking`. La valeur par défaut est `ly:optimal-breaking`.

`page-breaking-system-system-spacing`
: Cette variable permet de « tromper » l'algorithme de gestion des sauts de page quant à la valeur de `system-system-spacing`. Ainsi, lorsque `page-breaking-system-system-spacing.padding` a une valeur nettement supérieure à `system-system-spacing.padding`, l'algorithme en question aura tendance à disposer moins de systèmes sur une même page. Cette variable est par défaut non définie.

`page-count`
: Le nombre de pages que devra comporter la partition. Cette variable est par défaut non définie.

`page-spacing-weight`
: Lorsqu'il utilise l'algorithme `ly:optimal-breaking` pour gérer les sauts de page, LilyPond doit faire des compromis entre étirement horizontal et vertical afin de présenter un espacement acceptable. Cette variable définit l'importance relative des espacements entre la page (verticalité) et la ligne (horizontalité). Une valeur élevée privilégiera l'espacement au niveau de la page. La valeur par défaut est de `10`.

Les variables qui suivent ne sont effectives que lorsque l'algorithme `page-breaking` adopte la fonction `ly:page-turn-breaking`. Les sauts de page sont alors positionnés de sorte à minimiser le nombre de tournes. Dans la mesure où il faut tourner la feuille pour passer d'une page impaire à une page paire, sera privilégiée une répartition qui présente une dernière page impaire. Les endroits où une tourne serait appropriée peuvent s'indiquer à l'aide d'un `\allowPageTurn` ou laissés à l'appréciation du `Page_turn_engraver` – voir Section 28.2.7 [Optimisation des tournes], page 717.

Lorsqu'aucune option n'est satisfaisante pour placer judicieusement les tournes, LilyPond peut décider d'insérer une page blanche au milieu d'une partition ou entre deux partitions successives, voire même finir par une page paire. La valeur des trois variables qui suivent peut se voir augmentée de façon à diminuer ces risques.

Il s'agit ici de pénalité ; autrement dit, au plus la valeur est élevée, au moins l'action associée sera favorisée en regard des autres choix.

`blank-page-penalty`
: Pénalité pour apparition d'une page blanche en cours de partition. L'attribution d'une valeur élevée à `blank-page-penalty` alors qu'a été activé `ly:page-turn-breaking` forcera LilyPond à éviter de placer une page blanche au milieu de la partition, quitte à espacer d'autant plus la musique pour remplir cette page blanche et la suivante. La valeur par défaut est de `5`.

`blank-last-page-penalty`
: Pénalité pour fin de partition intervenant sur une page paire. L'attribution d'une valeur élevée à `blank-last-page-penalty` alors qu'a été activé `ly:page-turn-breaking` forcera LilyPond à éviter de terminer la partition sur une page paire, quitte à ajuster les espacements jusqu'à obtenir une page de plus ou une de moins. La valeur par défaut est de `0`.

`blank-after-score-page-penalty`
: Pénalité pour apparition d'une page blanche entre deux partitions. Sa valeur est par défaut inférieure à celle de `blank-page-penalty` ; nous préférons qu'une page blanche s'insère après la fin de la partition plutôt qu'au milieu. La valeur par défaut est de `2`.

Voir aussi

Manuel de notation : Section 28.2.3 [Minimisation des sauts de page], page 716, Section 28.2.2 [Optimisation des sauts de page], page 716, Section 28.2.7 [Optimisation des tournes], page 717, Section 28.2.5 [Présentation en ligne continue], page 717, Section 28.2.4 [Présentation en page continue], page 717, Section 28.2.6 [Présentation en rouleau], page 717, Section 28.2 [Sauts de page], page 715.

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

26.6.3 Variables de gestion des numéros de page

Les valeurs par défaut sont définies dans le fichier `ly/paper-defaults-init.ly`.

`auto-first-page-number`
: L'algorithme qui gère les sauts de page prend en compte le fait que le premier numéro de page soit pair ou impair. Lorsque cette fonctionnalité est activée, l'algorithme

des sauts de page décidera de lui-même si le premier numéro sera pair ou impair, ce qui se traduira par un éventuel incrément de un. La valeur par défaut est `#f`.

`first-page-number`
: Le numéro de la première page. La valeur par défaut est de `#1`.

`print-first-page-number`
: Cette variable permet d'imprimer le numéro de page y compris sur la première. La valeur par défaut est `#f`.

`print-page-number`
: La désactivation de cette variable permet d'obtenir des pages non numérotées. La valeur par défaut est `#t`.

`page-number-type`
: Le type de chiffres à utiliser pour la numérotation : `'arabic` (arabes), `'roman-ij-lower` (romains minuscules avec ligature ij), `'roman-ij-upper` (romains majuscules avec ligature IJ), `'roman-lower` (romains minuscules) ou `'roman-upper` (romains majuscules) . La valeur par défaut est `'arabic`.

`bookpart-level-page-numbering`
: L'activation de cette variable permet d'obtenir une pagination indépendante pour chacune des parties, démarrant à `first-page-number` (1 par défaut).

  On peut aussi l'utiliser pour une unique partie. C'est typiquement le cas de propos liminaires faisant l'objet d'une partie indépendante et paginée en chiffres romains.

```
\book {
  \bookpart {
    \paper {
      bookpart-level-page-numbering = ##t
      page-number-type = #'roman-lower
    }
    \markuplist \wordwrap-lines {
      Lorem ipsum dolor sit amet.
    }
  }
  \bookpart {
    ...
  }
}
```

Voir aussi

Fichiers d'initialisation : `ly/paper-defaults-init.ly`.

Problèmes connus et avertissements

Les pages au numéro impair sont toujours à droite. Pour que la musique commence en page 1, le dos de la page de garde doit être vide de telle sorte que la page une se retrouve à droite.

26.6.4 Variables supplémentaires d'en-tête et *markup*

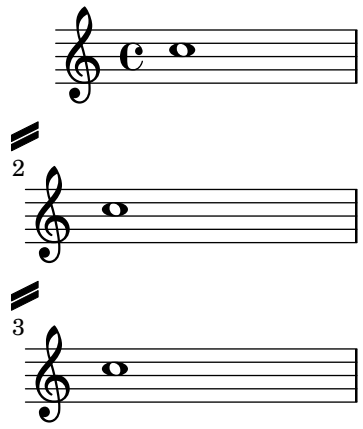
`print-all-headers`
: Lorsque cette variable est activée, l'intégralité des champs d'en-tête sera imprimée pour chaque bloc `\score`, plutôt que les seuls champs `piece` et `opus`. Voir Chapitre 21 [Titres et entêtes], page 615, pour les cas d'usage. La valeur par défaut est `#f`.

`reset-footnotes-on-new-page`
: Lorsque cette variable est activée, la numérotation des notes de bas de page est réinitialisée à chaque page. La désactivation de cette variable permet une numérotation sur l'intégralité de l'ouvrage. La valeur par défaut est `#t`.

`system-separator-markup`
: Il s'agit en l'occurrence d'insérer un objet de type *markup* entre chaque système, comme on le voit dans nombre de partitions orchestrales. Cette variable n'est pas définie par défaut. La commande `\slashSeparator` – définie dans le fichier `ly/titling-init.ly` – fournit un *markup* relativement courant :

```
#(set-default-paper-size "a8")

\book {
  \paper {
    system-separator-markup = \slashSeparator
  }
  \header {
    tagline = ##f
  }
  \score {
    \relative { c''1 \break c1 \break c1 }
  }
}
```



`footnote-separator-markup`
: Il s'agit d'un objet *markup* qui vient s'insérer au-dessus du texte de la note de bas de page. C'est, par défaut, une ligne horizontale centrée, définie dans `ly/paper-defaults-init.ly`.

Voir aussi

Fichiers d'initialisation : `ly/paper-defaults-init.ly`, `ly/titling-init.ly`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

L'en-tête par défaut, formé d'une seule ligne, est constitué du numéro de page et du champ `instrument` contenu dans le bloc `\header`.

26.6.5 Variables de débogage

Les variables `debug-beam-scoring`, `debug-slur-scoring` et `debug-tie-scoring` permettent d'afficher des informations de débogage en matière de ligature et liaisons de phrasé ou de tenue.

Voir Section “Débogage des algorithmes d’évaluation” dans *Guide du contributeur*, en anglais, pour des informations détaillées sur l’utilité de ces variables.

27 Mise en forme de la partition

Nous allons voir ici les options du bloc `\layout`. Elles sont plus particulièrement destinées à gérer la mise en forme de la partition.

27.1 Le bloc `\layout`

Alors que le bloc `\paper` définit le formatage des pages pour l'intégralité du document, le bloc `\layout` gère la mise en forme spécifique à la partition. La mise en forme de la musique peut concerner toutes les partitions d'un même ouvrage, auquel cas un bloc `\layout` indépendant se placera en tête de fichier. Dans le cas où la mise en forme concerne une partition en particulier, un bloc `\layout` se placera au sein du bloc `\score` en question. Sont susceptibles d'apparaître dans un bloc `\layout` :

- la fonction Scheme `layout-set-staff-size`,
- dans des blocs `\context`, les modifications apportées aux différents contextes, et
- les variables normalement attachées au bloc `\paper` qui affecteront la mise en forme de la partition.

La fonction `layout-set-staff-size` fait l'objet de la rubrique suivante, Section 27.2 [Définition de la taille de portée], page 707. La modification des contextes est abordée dans d'autres chapitres – voir Section 33.4 [Modification des greffons de contexte], page 769, et Section 33.5 [Modification des réglages par défaut d'un contexte], page 771.

Les variables du bloc `\paper` que l'on peut retrouver dans un bloc `\layout` sont :

- `line-width`, `ragged-right` et `ragged-last` (voir Section 26.5.1 [Variables de marge et de largeur], page 697)
- `indent` et `short-indent` (voir Section 26.5.3 [Variables d'indentation et de décalage], page 699)
- `system-count` (voir Section 26.6.1 [Variables de gestion des sauts de ligne], page 700)

Les valeurs par défaut des variables ci-dessus sont récupérées du bloc `\paper`.

Voici un exemple de bloc `\layout` :

```
\layout {
  indent = 2\cm
  \context {
    \StaffGroup
    \override StaffGrouper.staff-staff-spacing.basic-distance = 8
  }
  \context {
    \Voice
    \override TextScript.padding = 1
    \override Glissando.thickness = 3
  }
}
```

Il est tout à fait possible que plusieurs blocs `\layout` cohabitent en tant qu'expressions de niveau supérieur. Ceci se révèle particulièrement utile lorsque différents réglages sont stockés dans des fichiers séparés qui sont inclus au besoin. Lorsqu'un bloc `\layout` est évalué, une copie de la configuration du `\layout` actuel est réalisée en interne, augmentée des aménagements apportés. Bien qu'on puisse considérer que le contenu des différents blocs `\layout` se cumule, c'est la dernière adaptation qui aura préséance en cas de situation conflictuelle – cas typique d'une même propriété modifiée dans différents blocs.

Par exemple, placer le bloc suivant

```
\layout {
  \context {
    \Voice
    \override TextScript.color = #magenta
    \override Glissando.thickness = 1.5
  }
}
```

après celui de l'exemple précédent aura pour effet de cumuler les adaptations de `padding` et `color` pour l'objet `TextScript`, mais la dernière adaptation apportée à la propriété `thickness` de `Glissando` remplace, ou masque, celle précédemment établie.

Les blocs `\layout` peuvent faire l'objet de variables, aux fins de les utiliser ultérieurement. Ceci requiert toutefois une attention particulière dans la mesure où cette manière de procéder n'est pas équivalente à une définition complète et globale.

Lorsque nous définissons la variable suivante,

```
layoutVariable = \layout {
  \context {
    \Voice
    \override NoteHead.font-size = 4
  }
}
```

qui contient une configuration de `\layout` avec l'adaptation `NoteHead.font-size`, cette combinaison n'est pas enregistrée en tant que configuration courante. Notez bien que la « configuration courante » est lue lorsque la variable est définie, non lorsqu'elle est utilisée ; par voie de conséquence, le contenu de la variable dépend de l'endroit où elle se trouve dans le code source.

Notre variable peut alors être utilisée au sein d'un autre bloc `\layout`, comme par exemple :

```
\layout {
  \layoutVariable
  \context {
    \Voice
    \override NoteHead.color = #red
  }
}
```

Un bloc `\layout` qui contient une variable comme ci-dessus ne recopie pas la configuration actuelle ; il utilise en fait le contenu de `layoutVariable` en tant que configuration de base pour adaptations ultérieures, en conséquence de quoi toute modification intervenant entre la définition et l'utilisation de la variable sera perdue.

Si `layoutVariable` est définie, ou rapatriée par un `\indude`, juste avant d'être utilisée, son contenu devient la configuration actuelle augmentée des adaptations que la variable contient. Considérant l'exemple d'utilisation de `layoutVariable` ci-dessus, le bloc `\layout` final contiendra donc :

```
TextScript.padding = 1
TextScript.color = #magenta
Glissando.thickness = 1.5
NoteHead.font-size = 4
NoteHead.color = #red
```

ainsi que les adaptations de `indent` et `StaffGrouper`.

Cependant, si la variable avait été définie bien avant le premier bloc `\layout`, la configuration actuelle ne contiendrait que

```
NoteHead.font-size= 4 % écrit dans la définition de la variable
NoteHead.color = #red % ajouté après l'utilisation de la variable
```

Une gestion attentive des variables de `\layout` se révèle être un outil précieux dans la mise en forme des sources et le retour à une configuration donnée.

Voir aussi

Manuel de notation : Section 33.5 [Modification des réglages par défaut d'un contexte], page 771.

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.

27.2 Définition de la taille de portée

La **taille de portée** *(staff size)* est fixée par défaut à 20 points, ce qui correspond à 7,03 mm – 1 point équivaut à 100/7227 pouce, soit 2 540/7 227 mm. Il existe trois manières de la modifier :

1. La taille des portées peut se définir globalement pour toutes les partitions d'un même fichier à l'aide de `set-global-staff-size`.

   ```
   #(set-global-staff-size 14)
   ```

   Ceci définit donc la hauteur des portées à 14 points (4,92 mm) par défaut ; toutes les fontes seront ajustées en conséquence.

   Cette fonction permet aussi de définir différentes tailles de portée pour les différents blocs `\book` :

   ```
   #(set-global-staff-size 30)
   \book {
     { c' }
   }

   #(set-global-staff-size 10)
   \book {
     { c' }
   }
   ```

2. La taille d'une partition particulière au sein d'un ouvrage se définit à l'aide d'un `layout-set-staff-size` placé dans le bloc `\layout` approprié :

   ```
   \score{
     ...
     \layout{
       #(layout-set-staff-size 14)
     }
   }
   ```

3. Pour l'affectation d'une taille particulière à l'une des portées d'un système, LilyPond dispose de la commande `\magnifyStaff`. Par exemple, les partitions traditionnelles de musique de chambre avec piano présentaient souvent des portées de piano de 7 mm alors que les autres portées étaient gravées à une hauteur de cinq septièmes (s'il y avait assez de place) ou trois cinquièmes (en cas de présentation resserrée) de cette hauteur. Une proportion de 5/7 se libelle ainsi :

   ```
   \score {
     <<
       \new Staff \with {
         \magnifyStaff #5/7
       } { ... }
       \new PianoStaff { ... }
   ```

```
  >>
}
```

Si la valeur de `fontSize` à utiliser est connue, la forme suivante peut s'employer :

```
\score {
  <<
    \new Staff \with {
      \magnifyStaff #(magstep -3)
    } { ... }
    \new PianoStaff { ... }
  >>
}
```

Mieux vaut éviter de réduire l'épaisseur des lignes si l'on veut que la partition s'approche au plus près des canons de la gravure traditionnelle.

Relation automatique entre fonte et taille

La fonte Ementaler fournit le jeu de symboles musicaux *Feta* dans huit tailles différentes. Chaque fonte correspond à une hauteur particulière de portée ; les petites tailles comportent des symboles plus épais pour être cohérent avec l'épaisseur relativement plus importante des lignes de la portée. Le tableau suivant répertorie les différentes tailles de police.

nom de la fonte	**hauteur de portée (pt)**	**hauteur de portée (mm)**	**utilisation**
feta11	11,22	3,9	format de poche
feta13	12,60	4,4	
feta14	14,14	5,0	
feta16	15,87	5,6	
feta18	17,82	6,3	carnet de chant
feta20	20	7,0	partition standard
feta23	22,45	7,9	
feta26	25,2	8,9	

Voir aussi
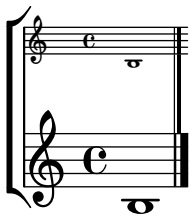
Manuel de notation : Section 7.1.1 [Indication de la taille de fonte musicale], page 286, Section B.8 [La fonte Emmentaler], page 935.

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.

Problèmes connus et avertissements

Lorsque `\magnifyStaff` est utilisé au sein d'un `StaffGroup`, les objets `BarLine` ne s'alignent plus, en raison des modifications apportées à leurs propriétés `thick-thickness`, `hair-thickness` et `kern`.

```
\new StaffGroup
  <<
    \new Staff \with { \magnifyStaff #1/2 } { b1 \bar "|." }
    \new Staff { b }
  >>
```



Vous pouvez opter pour annuler le redimensionnement des objets BarLine, l'imiter pour les autres portées, ou bien encore appliquer une valeur intermédiaire à toutes les portées.

```
#(define bar-line-props
  '((BarLine thick-thickness)
    (BarLine hair-thickness)
    (BarLine kern)))

mus = { b1 \bar "|."}
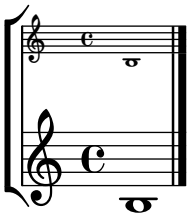
\markup "Annulation de \\magnifyStaff pour les barres de mesure :"
\new StaffGroup
  <<
    \new Staff
      \with {
        \magnifyStaff #1/2
        #(revert-props 'magnifyStaff 0 bar-line-props)
      }
      \mus
    \new Staff
      \mus
  >>

\markup "Imitation de \\magnifyStaff dans les autres portées :"
\new StaffGroup
  <<
    \new Staff
      \with { \magnifyStaff #1/2 }
      \mus
    \new Staff
      \with {
        #(scale-props 'magnifyStaff 1/2 #t bar-line-props)
      }
      \mus
  >>
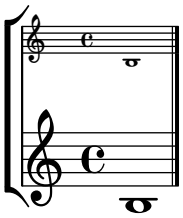
\markup "Application à tous de valeurs intermédiaires :"
\new StaffGroup
  <<
    \new Staff
```

```
    \with {
      \magnifyStaff #1/2
      #(scale-props 'magnifyStaff 3/2 #t bar-line-props)
    }
    \mus
  \new Staff
    \with {
      #(scale-props 'magnifyStaff 3/4 #t bar-line-props)
    }
    \mus
>>
```

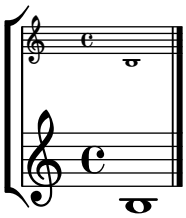
Annulation de \magnifyStaff pour les barres de mesure :



Imitation de \magnifyStaff dans les autres portées :



Application à tous de valeurs intermédiaires :



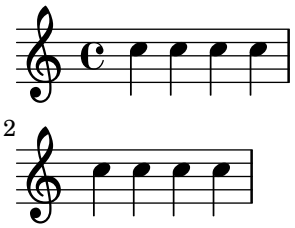
28 Sauts

28.1 Sauts de ligne

Les sauts de ligne sont normalement gérés de façon automatique. Ils interviennent de telle sorte qu'une ligne ne soit ni trop resserrée, ni trop aérée, et que des lignes consécutives aient à peu près la même densité.

Vous pouvez cependant insérer l'instruction `\break` à l'endroit où vous le jugez utile pour « forcer » le passage à la ligne suivante :

```
\relative c'' {
  c4 c c c | \break
  c4 c c c |
}
```



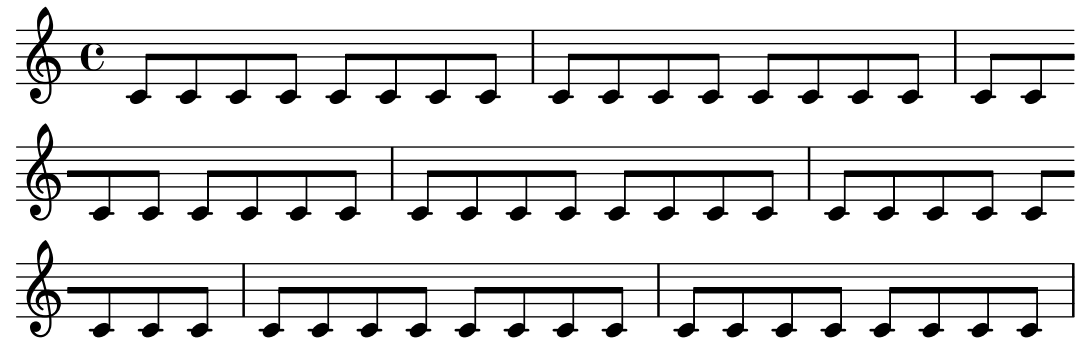
Par défaut, un saut de ligne ne peut se produire qu'en présence d'une barre de mesure. D'autres facteurs peuvent empêcher la survenance d'un saut de ligne :

- une note ou un silence se prolonge sur la mesure suivante,
- la présence d'un bandeau non ruptible, telle une ligature ou un glissando à cheval sur les mesures.

La commande `\break` force le saut de ligne dans tous les cas, sans tenir compte de la présence d'une barre de mesure ou des autres facteurs. On peut aussi s'affranchir de tous ces facteurs à l'aide de la commande `\allowBreak`. Dans l'exemple suivant, les ruptures sont autorisées partout, y compris au milieu des mesures et malgré la présence de ligature.

```
\*56 { c'8 \allowBreak }

\paper {
  indent = 0
  line-width = 140\mm
}
```



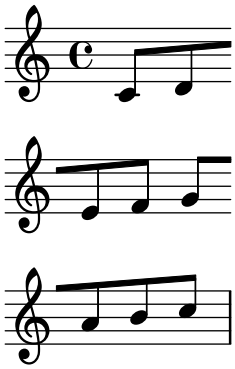
Il existe différents moyens d'éviter l'abus de `\allowBreak` afin de contourner les facteurs ci-dessus qui empêchent les ruptures.

- Le `Bar_engraver` interdit les ruptures entre les barres de mesure lorsqu'est activée `forbidBreakBetweenBarLines`. Pour s'en affranchir, il suffit de désactiver cette propriété.

```
\fixed c' {
  c8 d e f g a b c'
}

\layout {
  \context {
    \Score
    forbidBreakBetweenBarLines = ##f
  }
}
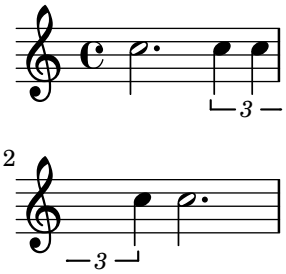
\paper {
  indent = 0
  line-width = 30\mm
}
```



- Notes et silences en suspend en fin de mesure n'interdiront plus les ruptures dès lors que le `Forbid_line_break_engraver` aura été supprimé du contexte `Voice` :

```
\new Voice \with {
  \remove Forbid_line_break_engraver
} \relative {
  c''2. \tuplet 3/2 { c4 c c } c2.
}
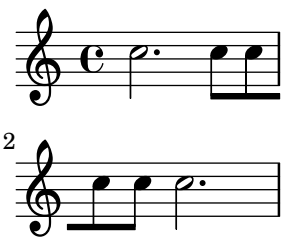
\paper {
  indent = 0
  line-width = 35\mm
}
```



- Une ligature ou autre bandeau non ruptible s'étendant sur deux mesures consécutives seront ignorés dès lors que leur propriété `breakable` est activée.

```
\relative c'' {
  \override Beam.breakable = ##t
  c2. c8[ c |
  c8 c] c2. |
}

\paper {
  indent = 0
  line-width = 35\mm
}
```



L'instruction opposée, \noBreak, interdira toute tentative de saut de ligne à la fin de la mesure où elle est explicitée.

Au sein même d'une pièce, les sauts de lignes automatiques sont inhibés dans la musique encadrée par les commandes \autoLineBreaksOff et \autoLineBreaksOn. Dans le cas où les sauts de page automatiques devraient eux aussi être inhibés, ce sont les commandes \autoBreaksOff et \autoBreaksOn qu'il faudrait utiliser. Les sauts de ligne ou de page manuels ne sont pas affectés par ces commandes. Notez bien que la désactivation du positionnement automatique des sauts des ligne peut avoir pour effet un débordement de la musique dans la marge de droite si tout ne peut être contenu sur une ligne.

Des sauts de ligne automatiques peuvent cependant être autorisés en un point particulier à l'aide d'un \once \autoLineBreaksOn sur une barre de mesure. Ceci ne concerne pas les sauts de page. Ceci indique qu'un saut de ligne peut intervenir à cet endroit précis, mais ne le force en aucun cas.

LilyPond dispose de deux variables de base pour influencer l'espacement au niveau des lignes. Toutes deux se définissent dans un bloc \layout, indent réglant l'indentation de la première ligne, et line-width la longueur des lignes.

L'activation du commutateur ragged-right au sein du bloc \layout aura pour effet de terminer les systèmes là où ils prendraient fin normalement plutôt que de les étirer sur toute la longueur de la ligne. Ceci est particulièrement utile pour de petits fragments ou pour vérifier la densité induite par l'espacement naturel.

Le commutateur ragged-last est équivalent à ragged-right, à ceci près qu'il n'affecte que la dernière ligne de la pièce.

```
\layout {
  indent = 0\mm
  line-width = 150\mm
  ragged-last = ##t
}
```

L'utilisation conjointe de \break et de blancs dans une section \repeat vous permettra de positionner des sauts de ligne à intervalle régulier. Par exemple, les 28 mesures de ce qui suit, si l'on est à 4/4, seront coupées toutes les quatre mesures, pas ailleurs :

```
<<
  \repeat unfold 7 {
```

```
    s1 \noBreak s1 \noBreak
    s1 \noBreak s1 \break
  }
  { et ici la musique... }
>>
```

Commandes prédéfinies

\break, \allowBreak, \noBreak, \autoBreaksOff, \autoBreaksOn, \autoLineBreaksOff, \autoLineBreaksOn.

Morceaux choisis

Recours à une voix supplémentaire pour gérer les sauts

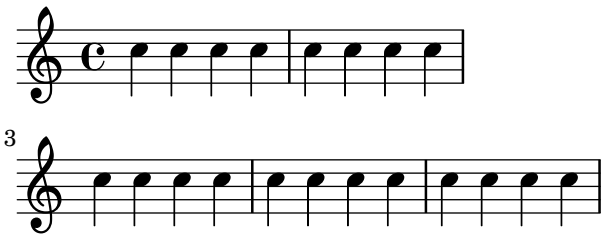
Il est souvent plus pratique de séparer ce qui est purement musical et les informations concernant les sauts de ligne ou de page, en créant une voix supplémentaire dédiée. Cette voix spécifique ne contiendra que des blancs – des silences invisibles \skip –, des \break, des \pageBreak et autres informations concernant les ruptures.

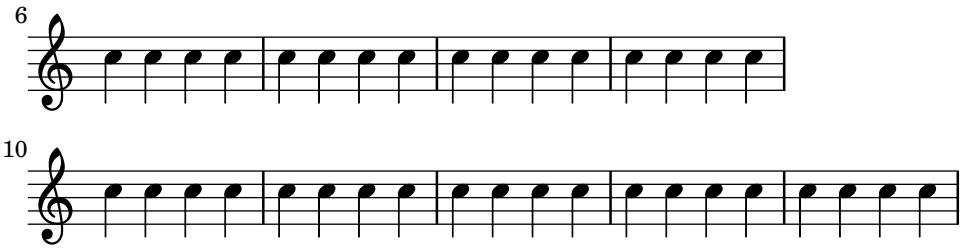
Cette manière de procéder est tout à fait indiquée lorsque vous ajustez les line-break-system-details et autres propriétés fort intéressantes de l'objet NonMusicalPaperColumn.

```
music = \relative c'' { c4 c c c }

\score {
  \new Staff <<
    \new Voice {
      s1*2 \break
      s1*3 \break
      s1*4 \break
      s1*5 \break
    }
    \new Voice {
      \*2 \music
      \*3 \music
      \*4 \music
      \*5 \music
    }
  >>
}

\paper {
  indent = 0
  line-width = 140\mm
  ragged-right = ##t
}
```





Voir aussi

Manuel de notation : Section 27.1 [Le bloc `\layout`], page 705, Section 26.6.1 [Variables de gestion des sauts de ligne], page 700.

Référence des propriétés internes : Section “LineBreakEvent” dans *Référence des propriétés internes*.

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.

Problèmes connus et avertissements

Les commandes `\autoLineBreaksOff` et `\autoBreaksOff` doivent impérativement se placer après le début de la musique pour éviter tout message d'erreur.

28.2 Sauts de page

Cette section présente les différentes méthodes de gestion des sauts de page, ainsi que les moyens de les modifier.

28.2.1 Saut de page manuel

La gestion automatique des sauts de page se contrôle à l'aide des commandes `\pageBreak` et `\noPageBreak`. Ces commandes fonctionnent de manière analogue à `\break` et `\noBreak` pour les sauts de ligne et se placent donc au moment d'une barre de mesure. Elles permettent de forcer, ou d'interdire, un saut de page à la prochaine barre de mesure. Comme on peut s'y attendre, `\pageBreak` force aussi le saut de ligne.

Les commandes `\pageBreak` et `\noPageBreak` peuvent se trouver à des niveaux supérieurs, entre deux partitions ou *markups* de premier rang.

Au sein même d'une pièce, les sauts de page automatiques sont inhibés dans la musique encadrée par les commandes `\autoPageBreaksOff` et `\autoPageBreaksOn`. Les sauts de page manuels ne sont pas affectés par ces commandes.

Tout comme `ragged-right` et `ragged-last` qui permettent de gérer la répartition horizontale, LilyPond dispose de commutateurs équivalents au niveau de la verticalité. `ragged-bottom`, une fois activé, empêchera les systèmes de se répartir sur la page. Quant à `ragged-last-bottom` (valorisé à `#t` par défaut), il laissera un espace vide en dernière page, y compris pour chaque `\bookpart`. Pour de plus amples détails, reportez-vous à Section 26.3 [Variables d'espacement vertical fixe], page 694.

Les sauts de page sont générés par la fonction `page-breaking`. LilyPond dispose de plusieurs algorithmes en la matière : `ly:optimal-breaking`, `ly:page-turn-breaking` et `ly:minimal-breaking`. C'est `ly:optimal-breaking` qui est activé par défaut, mais rien ne vous empêche d'en changer, par l'intermédiaire du bloc `\paper` :

```
\paper {
  page-breaking = #ly:page-turn-breaking
}
```

Lorsqu'un ouvrage contient plusieurs partitions et un certain nombre de pages, la gestion des sauts de page finit par devenir très gourmande, tant au niveau du processeur que de la mémoire.

Vous pouvez cependant alléger la charge en recourant à des blocs `\bookpart` afin de sectionner l'ouvrage que vous traitez ; les sauts de page seront alors gérés individuellement au niveau de chacune des parties. Par ailleurs, cela vous autorisera une gestion différente selon les sections.

```
\bookpart {
  \header {
    subtitle = "Préface"
  }
  \paper {
     %% Pour une partie constituée principalement de texte
     %% ly:minimal-breaking est plus judicieux.
     page-breaking = #ly:minimal-breaking
  }
  \markup { ... }
  ...
}
\bookpart {
  %% Cette partie étant purement musicale,
  %% retour au style par défaut (optimal-breaking).
  \header {
    subtitle = "Premier mouvement"
  }
  \score { ... }
  ...
}
```

Commandes prédéfinies

`\pageBreak`, `\noPageBreak`, `\autoPageBreaksOn`, `\autoPageBreaksOff`.

Voir aussi

Manuel de notation : Section 26.6.2 [Variables de gestion des sauts de page], page 700.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Problèmes connus et avertissements

Le préfixe `\once` est ineffectif en ce qui concerne les commandes `\autoPageBreaksOn` et `\autoPageBreaksOff`. Si le positionnement automatique des sauts de page est désactivé et qu'il est réactivé pour permettre un saut de page, il doit le rester pendant quelques mesures (le nombre précis de mesures dépendant de la pièce) avant d'être à nouveau désactivé, autrement l'opportunité de passer à la page suivante ne sera pas saisie.

28.2.2 Optimisation des sauts de page

LilyPond, pour déterminer où placer un saut de page, utilise par défaut la fonction `ly:optimal-breaking`. Celle-ci tend à trouver une rupture qui évite d'obtenir à la fois une page trop dense ou exagérément aérée. Contrairement à la fonction `ly:page-turn-breaking`, elle n'a aucune notion de ce qu'est une « tourne ».

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

28.2.3 Minimisation des sauts de page

La fonction `ly:minimal-breaking` est celle qui réalise le moins de calculs pour positionner les sauts de page. Elle mettra le plus de systèmes possible sur une page avant de passer à la

suivante. On peut donc la préférer lorsque la partition s'étend sur beaucoup de pages ou lorsque les autres fonctions de gestion des sauts de page ralentissent nettement le traitement, sont trop gourmandes en mémoire ou qu'il y a beaucoup de texte. Il suffit de la mentionner au sein du bloc `\paper` :

```
\paper {
  page-breaking = #ly:minimal-breaking
}
```

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

28.2.4 Présentation en page continue

La fonction `ly:one-page-breaking` constitue un algorithme de calcul des sauts de page particulier en ceci que la hauteur de page sera automatiquement ajustée à la longueur de la partition, de telle sorte que toute la musique tienne sur une seule page. La variable `paper-height` du bloc `\paper` est ignorée, mais les autres réglages restent disponibles. En particulier, l'espacement entre le dernier système (ou un *markup* de premier niveau) et le pied de page est réglable à l'aide de la variable `last-bottom-spacing` du bloc `\paper`. La largeur de page n'est, par défaut, pas modifiée ; ceci peut s'ajuster à l'aide de la variable `paper-width` du bloc `\paper`.

Problèmes connus et avertissements

`ly:one-page-breaking` est à ce jour incompatible avec l'utilisation de `\bookpart`.

28.2.5 Présentation en ligne continue

La fonction `ly:one-line-breaking` constitue un algorithme de calcul des sauts de page particulier en ceci que chaque partition fait l'objet d'une page unique, d'une seule ligne. Cette fonctionnalité s'affranchit de l'impression des titres et marges ; seule la partition est affichée.

La largeur de page est ajustée de telle sorte que la pièce la plus longue tienne sur une seule ligne. En particulier, les variables `paper-width`, `line-width` et `indent` du bloc `\paper` seront ignorées ; les `left-margin` et `right-margin` seront honorées. La hauteur de page ne sera pas modifiée.

28.2.6 Présentation en rouleau

La fonction `ly:one-line-auto-height-breaking` opère comme `ly:one-line-breaking`, à ceci près que la hauteur de page s'adapte automatiquement à la hauteur de la musique. Dans les faits, la variable `paper-height` du bloc `\paper` s'ajuste de façon à englober la hauteur de la partition la plus étendue, plus les `top-margin` et `bottom-margin`.

L'affectation d'une valeur à `top-system-spacing` influencera le positionnement vertical de la musique. Sa désactivation – valorisation à `#f` au sein du bloc `\paper` – aura pour effet de simplement placer la musique entre les marges supérieure et inférieure.

28.2.7 Optimisation des tournes

Aboutir à une configuration des sauts de page de telle sorte que les pages de droite se terminent toujours par un silence devient souvent une nécessité. En effet, l'exécutant pourra alors tourner la page sans risquer de manquer des notes. La fonction `ly:page-turn-breaking` tend à trouver une rupture qui évite d'obtenir à la fois une page trop dense ou exagérément aérée, tout en tenant compte du fait qu'une tourne ne saurait intervenir qu'à certains endroits.

L'utilisation de cette fonction se fait en deux étapes. Il vous faut tout d'abord l'activer au sein du bloc `\paper` comme indiqué à la rubrique Section 28.2 [Sauts de page], page 715. Vous devrez, dans un deuxième temps, informer la fonction des endroits où les sauts de page sont permis.

Cette deuxième étape se réalise de deux manières différentes. Vous pouvez spécifier manuellement chaque tourne potentielle en insérant un `\allowPageTurn` à l'endroit approprié de votre fichier source.

Toutefois, cette option peut vite se révéler fastidieuse selon l'ampleur de l'œuvre. Vous pouvez alors recourir au `Page_turn_engraver` que vous mentionnerez dans un contexte de voix ou de portée. Ce graveur de tournes recherchera dans le contexte en question les passages sans note. Notez bien qu'il ne recherche pas des silences, mais l'absence de notes ; autrement dit, il ne restera pas inactif dans le cadre d'une portée polyphonique dont l'une des parties contiendrait des silences. Lorsqu'il rencontre un fragment suffisamment long ne contenant aucune note, il insère un `\allowPageTurn` à la barre terminant ce fragment, à moins qu'il ne rencontre en chemin une « barre spéciale » – telle une double barre – auquel cas il y déposera le `\allowPageTurn`.

Le `Page_turn_engraver` examine la propriété de contexte `pageTurnMinimumRestLength` pour déterminer quelle doit être la longueur d'un fragment sans note avant une tourne. La valeur par défaut de `pageTurnMinimumRestLength` est `1`, soit une ronde, et s'ajuste de la manière suivante :

```
\new Staff \with { \consists Page_turn_engraver }
{
  a4 b c d |
  R1 | % une tourne peut se placer ici
  a4 b c d |
  \set Staff.pageTurnMinimumRestLength = #5/2
  R1 | % il ne peut pas y avoir de tourne ici
  a4 b r2 |
  R1*2 | % une tourne peut se placer ici
  a1
}
```

Le `Page_turn_engraver` tient compte des reprises. C'est pourquoi il ne permettra une tourne que dans le cas où il y aura suffisamment de temps au début et à la fin de la reprise pour que l'exécutant puisse aisément revenir à la page précédente. Le `Page_turn_engraver` est même capable d'interdire une tourne dans le cas d'une reprise de courte durée, ajustable au travers de la propriété de contexte `pageTurnMinimumRepeatLength`.

Les commandes de tourne – `\pageTurn`, `\noPageTurn` et `\allowPageTurn` – peuvent s'utiliser à des niveaux supérieurs, entre des blocs `\score` ou des *markups* de haut niveau.

Commandes prédéfinies

`\pageTurn`, `\noPageTurn`, `\allowPageTurn`.

Voir aussi

Manuel de notation : Section 26.6.1 [Variables de gestion des sauts de ligne], page 700.

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.

Problèmes connus et avertissements

Une partition ne devrait contenir qu'une seule instance du `Page_turn_engraver`, au risque de les voir se contredire.

29 Espacement vertical

L'espacement vertical dépend de trois éléments : la surface disponible – format de papier et marges –, l'espace qui doit séparer les systèmes, et l'espace qui sépare les portées d'un même système.

29.1 Espacement vertical au sein d'un système

LilyPond dispose de trois différents mécanismes permettant de contrôler l'espacement au sein d'un système selon trois catégories :

- *portées isolées*,
- *portées regroupées* (portées d'un même groupe, telles celles d'un `ChoirStaff`, etc.), et
- *lignes de non-portée* (`Lyrics`, `ChordNames`, etc.).

La hauteur de chaque système se détermine en deux phases. Les portées sont tout d'abord espacées selon la surface disponible. Puis les lignes autres que des portées, comme les paroles ou les accords, sont réparties entre les portées.

Les paragraphes qui suivent traitent exclusivement de la manière de gérer l'espacement entre les lignes d'un système – portée musicale ou non. Pour ce qui a trait aux espacements entre les systèmes, mouvements, annotations et marge, ils sont contrôlés par des variables attachées au bloc `\paper` et font l'objet du chapitre Section 26.4 [Variables d'espacement vertical fluctuant], page 695.

29.1.1 Propriétés d'espacement au sein d'un système

L'espacement entre les portées est géré par deux jeux de propriétés d'objet graphique (*grob*). Le premier, associé à l'objet graphique `VerticalAxisGroup`, est créé pour toute ligne de portée ou de non-portée. Le second, associé à l'objet graphique `StaffGrouper`, doit être explicitement créé pour un regroupement de portées particulier. Les propriétés qui leur sont attachées sont abordées en fin de section.

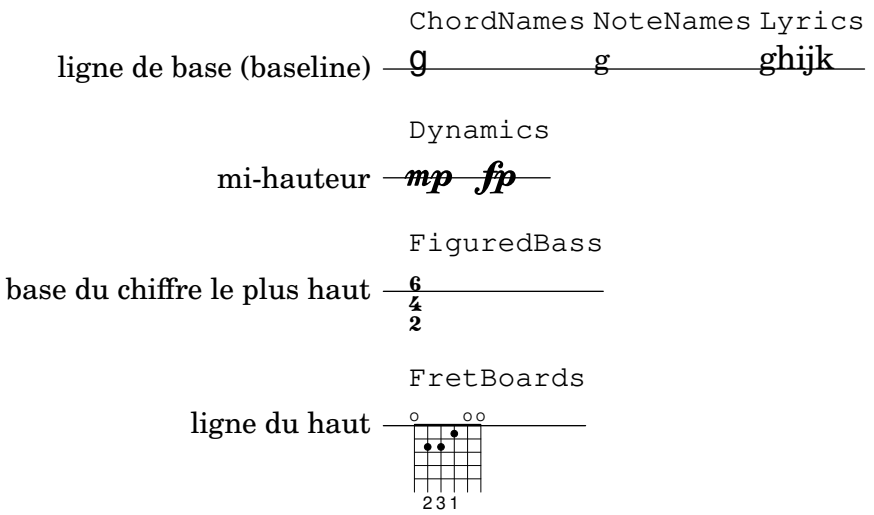
Le nom de ces propriétés, sauf `staff-affinity`, suit le schéma *item1*`-`*item2*`-spacing` – *item1* et *item2* étant les éléments à espacer. Notez bien que *item2* n'est pas forcément placé au-dessous : c'est le cas pour la propriété `nonstaff-relatedstaff-spacing` qui spécifie l'espacement d'une ligne de non-portée alors que sa `staff-affinity` a été déterminée à `UP`.

Toutes ces distances sont mesurées entre les *points de référence*[1] respectifs des éléments considérés. Le point de référence d'une portée est le centre vertical du `StaffSymbol` – la ligne médiane si `line-count` est impair, l'interligne médian si `line-count` est pair. Quant aux lignes rattachées à des portées – lignes de non-portée – le tableau suivant présente le point de référence pour chacune d'elles :

Ligne de non-portée	**Point de référence**
`ChordNames`	ligne de base
`NoteNames`	ligne de base
`Lyrics`	ligne de base
`Dynamics`	mi-hauteur du « m »
`FiguredBass`	ligne de base de l'élément le plus haut
`FretBoards`	ligne supérieure

En voici une représentation graphique :

[1] C'est une formulation hasardeuse. Pour les contextes, le positionnement des objets sur l'axe horizontal est calculé par l'algorithme de LilyPond (à partir d'objets `PaperColumn` entre autres), ce qui signifie que la seule information alors pertinente est la position verticale. En d'autres termes, le « point » de référence d'un contexte est le positionnement vertical sur lequel les autres *grobs* s'alignent.



Hormis `staff-affinity` – propriété attachée au *grob* `VerticalAxisGroup` –, chacune de ces propriétés est enregistrée sous la forme d'une liste associative dont la structure est identique à celle des variables du bloc `\paper` que nous avons examinées au chapitre Section 26.4 [Variables d'espacement vertical fluctuant], page 695. Les particularités en matière de modification d'une liste associative font l'objet d'un Section 35.8 [Modification de listes associatives], page 798. Les propriétés des objets graphiques se règlent avec un `\override` mentionné dans un bloc `\score` ou `\layout`, pas dans le bloc `\paper`.

L'exemple suivant illustre deux façons de modifier une liste associative. La première déclaration n'agit que sur une seule clé, alors que la seconde redéfinit la propriété dans son intégralité.

```
\new Staff \with {
  \override VerticalAxisGroup
              .default-staff-staff-spacing.basic-distance = 10
} { ... }

\new Staff \with {
  \override VerticalAxisGroup.default-staff-staff-spacing =
    #'(('basic-distance  . 10)
       (minimum-distance . 9)
       (padding . 1)
       (stretchability . 10))
} { ... }
```

La modification d'un espacement au niveau global se mentionne au sein du bloc `\layout` :

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup
      .default-staff-staff-spacing.basic-distance = 10
  }
}
```

Les réglages concernant les propriétés d'espacement vertical des objets graphiques sont répertoriées aux chapitres Section "VerticalAxisGroup" dans *Référence des propriétés internes* et Section "StaffGrouper" dans *Référence des propriétés internes*. Les propriétés relatives aux lignes de non-portée sont répertoriées selon la définition de leur contexte dans la Section "Contexts" dans *Référence des propriétés internes*.

Propriétés de l'objet `VerticalAxisGroup`

Les propriétés de l'objet `VerticalAxisGroup` s'ajustent à l'aide d'un `\override` au niveau d'un contexte `Staff` (ou son équivalent).

`staff-staff-spacing`
: Il s'agit de la distance entre la portée en cours et la portée qui suit au sein du même regroupement, qu'il y ait ou des lignes de non-portée (`Lyrics` ou autre entre les deux. Cette propriété ne s'applique pas à la dernière portée d'un système.

  En tout état de cause, la fonction Scheme `staff-staff-spacing` d'un `VerticalAxisGroup` affectera les propriétés du `StaffGrouper` si la portée est incluse dans un regroupement ; elle s'appliquera au `default-staff-staff-spacing` en l'absence de regroupement. Les portées peuvent donc s'aligner différemment selon qu'elles sont ou non regroupées. Pour obtenir le même espacement sans tenir compte des éventuels regroupements, cette fonction peut faire place à une complète redéfinition des espacements fluctuants à l'aide de règles dérogatoires comme vu précédemment. Au cas où seulement certaines +valeurs font l'objet d'une dérogation, les valeurs non mentionnées +seront ajustées sur celles de `default-staff-staff-spacing` (si +tant est qu'elles y soient définies).

`default-staff-staff-spacing`
: Il s'agit de la distance qui s'appliquera par défaut aux portées isolées, à moins que `staff-staff-spacing` n'ait été redéfini explicitement par un `\override`.

`staff-affinity`
: Il s'agit de la direction – `UP`, `DOWN` ou `CENTER` – que prendra une ligne de non-portée pour aller s'accoler aux portées adjacentes. Si vous lui attribuez `CENTER`, cette ligne de non-portée ira se placer à équidistance entre les portées qui l'encadrent, tout en tenant compte des éventuels risques de collision et autres contraintes d'espacement. Des lignes de non-portée adjacentes devraient avoir une `staff-affinity` allant de haut en bas – autrement dit, pas de `UP` après un `DOWN`. Une ligne de non-portée en dessous d'un système devrait avoir sa `staff-affinity` définie à `UP`. De la même manière, lorsque cette ligne surplombe un système, sa `staff-affinity` devrait être définie à `DOWN`. Prenez garde à la valeur que vous affectez à `staff-affinity` : si vous affectez la valeur `#f` à une ligne de non-portée, cette ligne sera considérée comme étant une portée ; à l'inverse, utiliser la propriété `staff-affinity` pour une portée lui fera perdre cette qualité.

`nonstaff-relatedstaff-spacing`
: Il s'agit de la distance entre la ligne de non-portée en cours et la portée la plus proche selon la `staff-affinity`, à la double condition qu'il n'y ait pas déjà une autre ligne de non-portée et que la valeur de `staff-affinity` soit `UP` ou `DOWN`. Lorsque la valeur de `staff-affinity` est égale à `CENTER`, la valeur de `nonstaff-relatedstaff-spacing` servira à centrer la ligne de non-portée entre les deux portées adjacentes même si une autre non-portée est présente (quelque soit le côté). Le positionnement d'une ligne de non-portée dépend donc à la fois des portées qui l'entourent tout comme des autres lignes de non-portée adjacentes. L'affectation d'une faible valeur à la propriété `stretchability` de l'un de ces types d'espacement l'avantagera ; lui affecter une valeur élevée aura pour conséquence de diminuer l'influence de l'espacement considéré.

`nonstaff-nonstaff-spacing`
: Il s'agit de la distance entre deux lignes de non-portée selon l'orientation définie par `staff-affinity` et dès lors qu'elles ont la même orientation. Bien entendu, ceci ne peut concerner que les valeurs `UP` et `DOWN` de `staff-affinity`.

`nonstaff-unrelatedstaff-spacing`
: Il s'agit de la distance entre une ligne de non-portée et la portée à l'opposé de l'orientation adoptée, à la double condition qu'il n'y ait pas déjà une autre ligne de non-portée et que la valeur de `staff-affinity` soit `UP` ou `DOWN`. Cette propriété trouve toute sa légitimité pour décaler une ligne de `Lyrics` de la portée à laquelle elle ne correspond pas.

Propriétés de l'objet `StaffGrouper`

Les propriétés de l'objet `StaffGrouper` s'ajustent à l'aide d'un `\override` au niveau d'un contexte `StaffGroup` (ou son équivalent).

`staff-staff-spacing`
: Il s'agit de la distance entre deux portées consécutives d'un même système. La propriété `staff-staff-spacing` de l'objet `VerticalAxisGroup` d'une portée en particulier peut se redéfinir à l'aide de règles dérogatoires.

`staffgroup-staff-spacing`
: Il s'agit de la distance entre la dernière portée d'un regroupement et la portée suivante, au sein d'un même système, y compris lorsqu'une ou plusieurs lignes de non-portée (tel `Lyrics`) s'insèrent entre les deux. Cette propriété ne concerne pas la dernière portée d'un système. Dans le cas où la propriété `staff-staff-spacing` d'une portée du regroupement a été ajustée au niveau de son propre `VerticalAxisGroup`, cette dernière aura préséance.

Voir aussi

Manuel de notation : Section 35.8 [Modification de listes associatives], page 798, Section 26.4 [Variables d'espacement vertical fluctuant], page 695.

Fichiers d'initialisation : `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

29.1.2 Espacement de portées isolées

Les `Staff`, `DrumStaff`, `TabStaff` entre autres sont des contextes de « portée » pouvant contenir plusieurs voix, mais pas une portée.

L'espacement de ces *portées isolées* est géré par les propriétés suivantes :

- Propriétés du `VerticalAxisGroup` :
  - `default-staff-staff-spacing`
  - `staff-staff-spacing`

Ces propriétés d'objet graphique sont expliquées une à une au chapitre Section 29.1.1 [Propriétés d'espacement au sein d'un système], page 719.

Certaines propriétés supplémentaires s'appliqueront dès lors que ces portées sont regroupées – voir Section 29.1.3 [Espacement de portées regroupées], page 723.

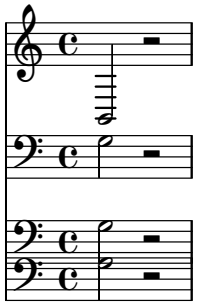
L'exemple suivant illustre la manière de gérer l'espacement de portées isolées à l'aide de la propriété `default-staff-staff-spacing`. Les mêmes règles appliquées de manière dérogatoire au `staff-staff-spacing` produiront les mêmes effets, y compris au sein de regroupements.

```
\layout {
  \context {
    \Staff
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 8)
         (minimum-distance . 7)
         (padding . 1))
  }
}

<<
  % The very low note here needs more room than 'basic-distance
  % can provide, so the distance between this staff and the next
  % is determined by 'padding.
  \new Staff { b,2 r | }

  % Here, 'basic-distance provides enough room, and there is no
  % need to compress the space (towards 'minimum-distance) to make
  % room for anything else on the page, so the distance between
  % this staff and the next is determined by 'basic-distance.
  \new Staff { \clef bass g2 r | }

  % By setting 'padding to a negative value, staves can be made to
  % collide.  The lowest acceptable value for 'basic-distance is 0.
  \new Staff \with {
    \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 3.5)
         (padding . -10))
  } { \clef bass g2 r | }
  \new Staff { \clef bass g2 r | }
>>
```



Voir aussi

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

29.1.3 Espacement de portées regroupées

Dans les partitions orchestrales ou de grande ampleur, il arrive souvent que des portées soient regroupées. L'espacement est alors plus important entre deux regroupements qu'entre les portées d'un même groupe.

Les *regroupements de portées* tels le StaffGroup ou le ChoirStaff sont des contextes qui peuvent contenir simultanément une ou plusieurs portées.

L'espacement entre les portées d'un même regroupement est géré par les propriétés suivantes :

- Propriétés du VerticalAxisGroup :
  - staff-staff-spacing
- Propriétés du StaffGrouper :
  - staff-staff-spacing
  - staffgroup-staff-spacing

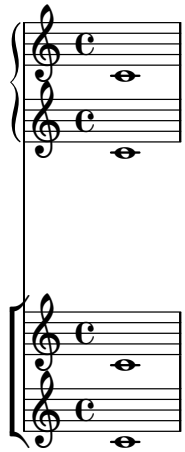
Ces propriétés d'objet graphique sont expliquées une à une au chapitre Section 29.1.1 [Propriétés d'espacement au sein d'un système], page 719.

L'exemple suivant illustre la manière de gérer l'espacement de portées regroupées, à l'aide des propriétés de l'objet graphique StaffGrouper :

```
\layout {
  \context {
    \Score
    \override StaffGrouper.staff-staff-spacing.padding = 0
    \override StaffGrouper.staff-staff-spacing.basic-distance = 1
  }
}

<<
  \new PianoStaff \with {
    \override StaffGrouper
                .staffgroup-staff-spacing
                .basic-distance = 20
  } <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>

  \new StaffGroup <<
    \new Staff { c'1 }
    \new Staff { c'1 }
  >>
>>
```



Voir aussi

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "VerticalAxisGroup" dans *Référence des propriétés internes*, Section "StaffGrouper" dans *Référence des propriétés internes*.

29.1.4 Espacement des lignes rattachées à des portées

Les *lignes de non-portée*, comme les `Lyrics` ou les `ChordNames`, sont des contextes dont les objets de rendu sont gravés à l'instar des portées – une ligne horizontale dans un système. En fait, les lignes de non-portée sont des contextes qui vont créer un objet de rendu `VerticalAxisGroup` auquel est attaché le Section "Axis_group_engraver" dans *Référence des propriétés internes*.

L'espacement des lignes de non-portée est géré par les propriétés suivantes :

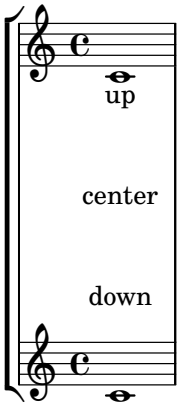
- Propriétés du `VerticalAxisGroup` :
  - `staff-affinity`
  - `nonstaff-relatedstaff-spacing`
  - `nonstaff-nonstaff-spacing`
  - `nonstaff-unrelatedstaff-spacing`

Ces propriétés d'objet graphique sont expliquées une à une au chapitre Section 29.1.1 [Propriétés d'espacement au sein d'un système], page 719.

L'exemple suivant utilise la propriété `nonstaff-nonstaff-spacing` pour gérer l'espacement entre des lignes consécutives de non-portée. Vous noterez que la valeur élevée attribuée à la clé `stretchability` permet aux paroles de s'étirer plus que de raison.

```
\layout {
  \context {
    \Lyrics
    \override VerticalAxisGroup
                .nonstaff-nonstaff-spacing
                .stretchability = 1000
  }
}

\new StaffGroup
<<
  \new Staff \with {
    \override VerticalAxisGroup.staff-staff-spacing =
      #'((basic-distance . 30))
  } { c'1 }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #UP
  } \lyricmode { up }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #CENTER
  } \lyricmode { center }
  \new Lyrics \with {
    \override VerticalAxisGroup.staff-affinity = #DOWN
  } \lyricmode { down }
  \new Staff { c'1 }
>>
```



Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`, `scm/define-grobs.scm`.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

29.2 Positionnement explicite des portées et systèmes
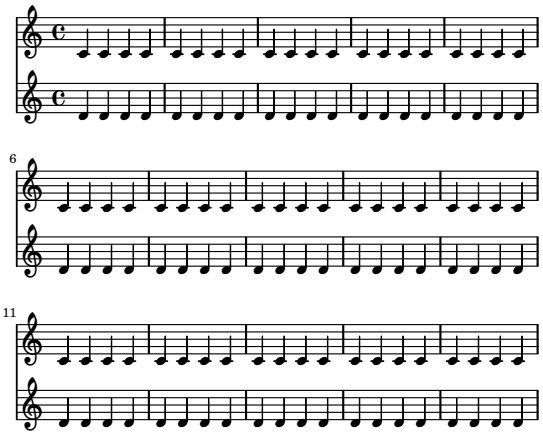
Il est possible d'aborder l'espacement vertical d'une autre manière que ce qui est indiqué dans la rubrique précédente, en utilisant la propriété `line-break-system-details` de l'objet graphique `NonMusicalPaperColumn`. Alors que les mécanismes d'espacement vertical flexible spécifient un décalage vertical, `line-break-system-details` peut déterminer les positionnements exacts sur la page.

Cette propriété contient cinq différentes sous-propriétés : `X-offset`, `Y-offset`, `extra-offset`, `alignment-distances` et`bottom-padding`.

Pour bien comprendre comment fonctionnent ces différentes sous-propriétés, commençons par examiner un exemple dépourvu de toute mise au point. Cette partition isole les informations de saut de ligne ou de page dans une voix spécifique. La mise en forme est ainsi séparée des événements musicaux ; ceci nous permettra d'y voir plus clair au fur et à mesure que nous avancerons. Pour plus de précisions, relisez Chapitre 28 [Sauts], page 711.
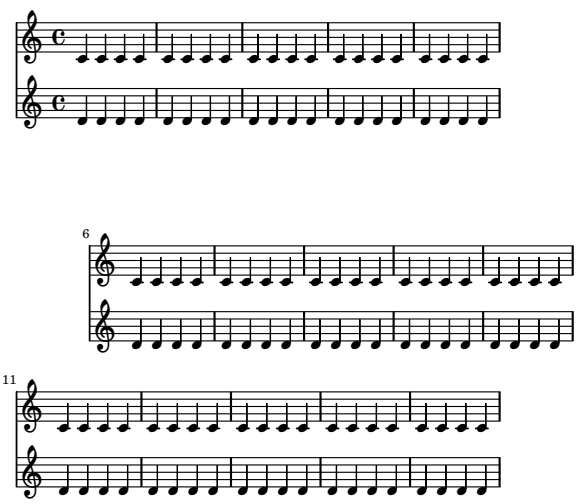
```
\header { tagline = ##f }
\paper { left-margin = 0\mm }
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break
        }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new Staff {
        \*15 { d'4 d' d' d' }
      }
    >>
  }
```
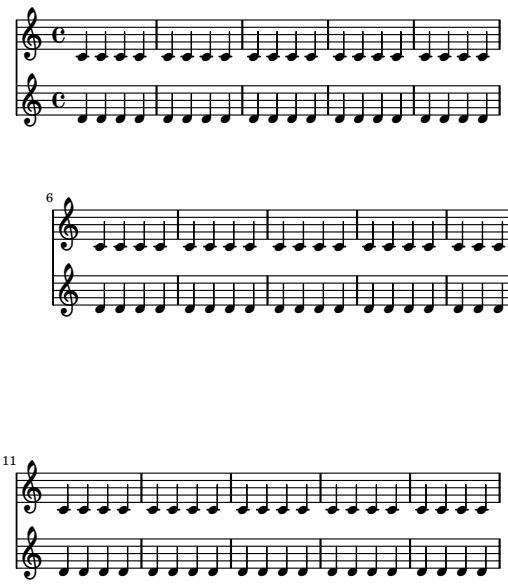
```
}
```



Déterminons maintenant les points de départ vertical et horizontal de chacun des systèmes à l'aide des sous-propriétés `Y-offset` et `X-offset` de `line-break-system-details`.

```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \once \override
            Score.NonMusicalPaperColumn
                 .line-break-system-details.Y-offset = 0
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn
                 .line-break-system-details.Y-offset = 30
          \once \override
            Score.NonMusicalPaperColumn
                 .line-break-system-details.X-offset = 10
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn
                 .line-break-system-details.Y-offset = 50
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new Staff { \*15 { d'4 d' d' d' } }
    >>
  }
}
```
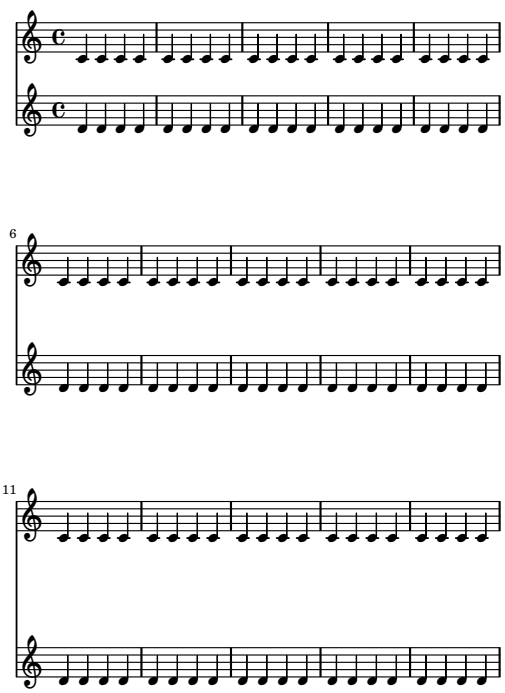


Contrairement au positionnement absolu accessible par `Y-offset` et `X-offset`, il est possible d'opter pour un positionnement relatif à l'aide de la sous-propriété `extra-offset`. Le placement sera relatif à la mise en forme par défaut ou au positionnement absolu géré par `X-offset` et `Y-offset`. La propriété `extra-offset` prend en argument une `paire` constituée des déplacements sur les axes horizontal et vertical.

```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn
              .line-break-system-details.extra-offset = #'(5 . 5)
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn
              .line-break-system-details.extra-offset = #'(0 . 20)
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new Staff { \*15 { d'4 d' d' d' } }
    >>
  }
}
```



Maintenant que chaque système est explicitement positionné, nous pouvons jouer sur la distance séparant les portées de chacun des systèmes, grâce à la sous-propriété `alignment-distances`. Comme on peut le constater, est utilisée une liste associative pour définir en même temps deux sous-propriétés différentes de `line-break-system-details`.

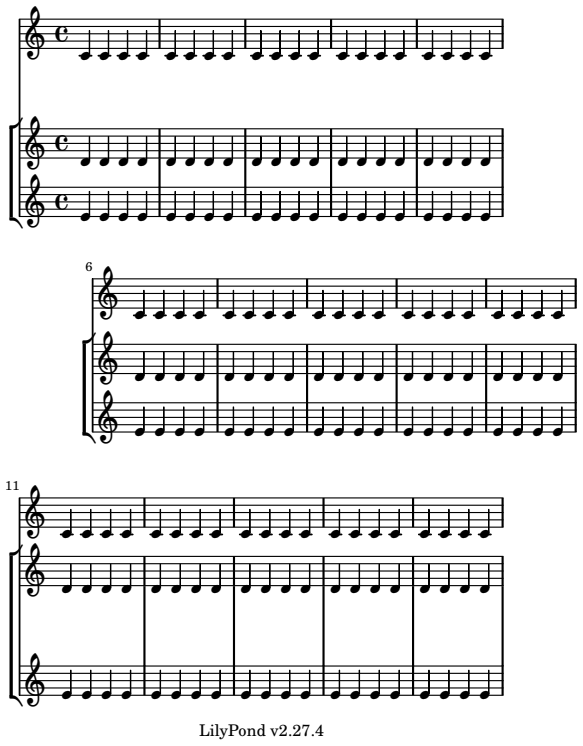
```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 0)
                 (alignment-distances . (10)))
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 30)
                 (alignment-distances . (15)))
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 65)
                 (alignment-distances . (20)))
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new Staff { \*15 { d'4 d' d' d' } }
    >>
  }
}
```



Voici maintenant comment positionner des portées à l'intérieur d'un système. Il est à noter que `alignment-distances` s'applique directement aux portées sans tenir compte d'un regroupement par `StaffGroup` ou autre. Une valeur à `#f` indique qu'il faudra prendre la valeur par défaut.
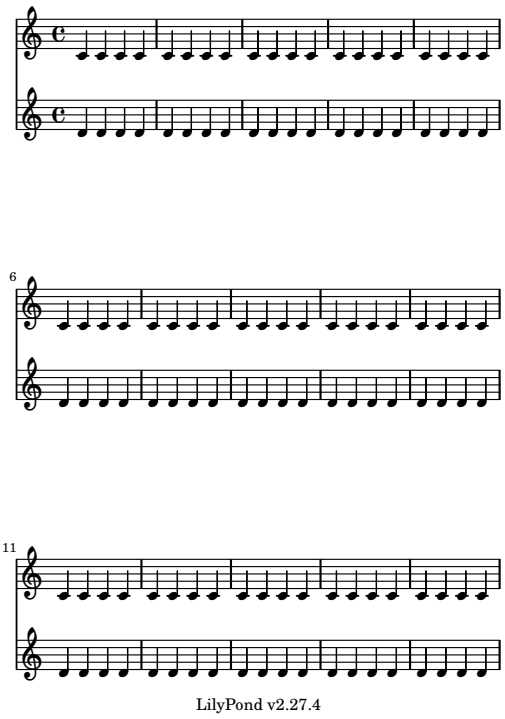
```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 0)
                 (alignment-distances . (15 8)))
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 35)
                 (X-offset . 10)
                 (alignment-distances . (#f 8)))
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((Y-offset . 65)
                 (alignment-distances . (8 15)))
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new StaffGroup <<
        \new Staff { \*15 { d'4 d' d' d' } }
        \new Staff { \*15 { e'4 e' e' e' } }
      >>
```

```
      >>
    }
}
```



La dernière sous-propriété de `line-break-system-details`, `bottom-padding`, agit de manière analogue à `last-bottom-spacing`. Il est important de garder à l'esprit qu'elle ne définit pas un espace compressible ou extensible, mais une valeur de décalage fixe et rigide entre le bas de la page et le dernier système. Voici notre exemple de base sans `ragged-bottom`, autrement dit réparti sur la hauteur de page, pour indiquer les distances par défaut.
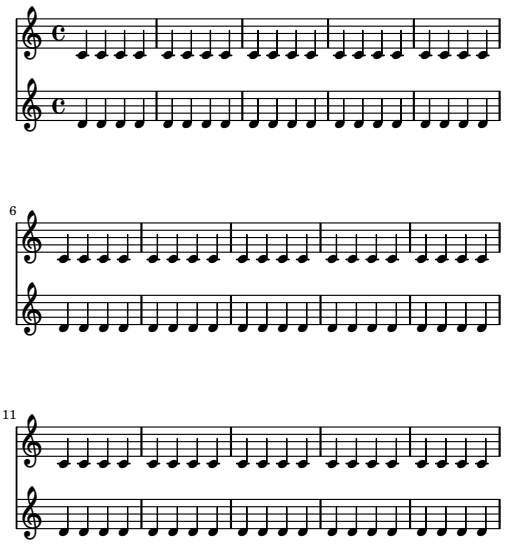
```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
      \new Staff { \*15 { d'4 d' d' d' }
      }
    >>
  }
  \paper { ragged-last-bottom = ##f }
}
```



Définir bottom-padding mène à ce résultat :

```
\book {
  \score {
    <<
      \new Staff <<
        \new Voice {
          s1*5 \break
          s1*5 \break
          \once \override
            Score.NonMusicalPaperColumn.line-break-system-details =
              #'((bottom-padding . 20))
          s1*5 \break }
        \new Voice { \*15 { c'4 c' c' c' } }
      >>
```

```
      \new Staff { \*15 { d'4 d' d' d' } }
    >>
  }
  \paper { ragged-last-bottom = ##f }
}
```



LilyPond v2.27.4

Quelques points à prendre en considération :

- Lorsque vous utilisez `alignment-distances`, les paroles et autres lignes de non-portée ne comptent pas pour une portée.
- Les nombres fournis aux sous-propriétés `X-offset`, `Y-offset`, `extra-offset` et `alignment-distances` sont considérés comme des multiples de la distance entre des portées adjacentes. Des valeurs positives remontent les portées et paroles, des valeurs négatives les descendent.
- Dans la mesure où `NonMusicalPaperColumn.line-break-system-details` permet de positionner systèmes et portées n'importe où sur une page, vous pourriez être en contradiction avec les dimensionnements de la feuille ou bien aboutir à des surimpressions. Soyez donc raisonnable quant aux différentes valeurs que vous affectez à ces sous-propriétés.

Voir aussi

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.

29.3 Résolution des collisions verticales

Vous savez de manière intuitive qu'un certain nombre d'objets en matière de notation musicale appartiennent à la portée, et que d'autres se placent en dehors de la portée. Entre autres objets externes, nous avons les marques repères, les textes et les nuances ; nous les appellerons « objets extérieurs à la portée ». La règle adoptée par LilyPond pour positionner verticalement ces objets extérieurs consiste à les placer au plus près de la portée tout en prenant garde d'éviter qu'il y ait chevauchement.

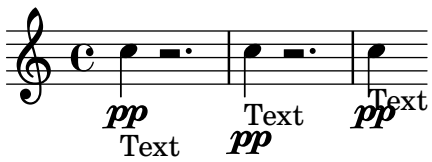
LilyPond utilise la propriété `outside-staff-priority` afin de déterminer si un objet est ou non un objet extérieur à la portée : lorsque la valeur de `outside-staff-priority` est numérique,

il s'agit d'un objet extérieur à la portée. De plus, la propriété `outside-staff-priority` indique à LilyPond l'ordre dans lequel ces objets doivent être disposés.

Tout d'abord, LilyPond dispose tous les objets qui ne sont pas externes. Les objets extérieurs à la portée sont alors triés selon l'ordre croissant de leur `outside-staff-priority`. Enfin, LilyPond prend chacun des ces objets et les positionne de telle sorte qu'il n'entrent pas en collision avec ceux qui on déjà été placés. Autrement dit, lorsque deux objets devraient se placer au même endroit, celui dont la `outside-staff-priority` est la plus faible sera disposé au plus près de la portée.

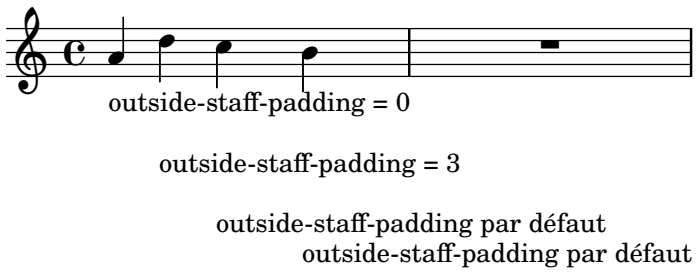
Une liste des valeur par défaut pour `outside-staff-priority` fait l'objet d'une Section B.17 [Valeurs par défaut de outside-staff-priority], page 952.

```
\relative c'' {
  c4_"Text"\pp
  r2.
  \once \override TextScript.outside-staff-priority = 1
  c4_"Text"\pp % this time the text will be closer to the staff
  r2.
  % by setting outside-staff-priority to a non-number,
  % we disable the automatic collision avoidance
  \once \override TextScript.outside-staff-priority = ##f
  \once \override DynamicLineSpanner.outside-staff-priority = ##f
  c4_"Text"\pp % now they will collide
}
```



Le décalage vertical entre des objets extérieurs à la portée se contrôle par la propriété `outside-staff-padding`.

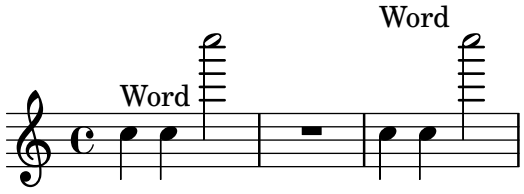
```
\relative {
  \once \override TextScript.outside-staff-padding = 0
  a'4-"outside-staff-padding = 0"
  \once \override TextScript.outside-staff-padding = 3
  d-"outside-staff-padding = 3"
  c-"outside-staff-padding par défaut"
  b-"outside-staff-padding par défaut"
  R1
}
```



Par défaut, les objets extérieurs à la portée sont positionnés en évitant les collisions horizontales avec des objets précédemment positionnés. Ceci peut cependant générer des situations où des objets se trouvent horizontalement trop proches. Comme l'illustre l'exemple suivant, la

propriété `outside-staff-horizontal-padding` permet d'accroître l'espace horizontal requis et repoussera verticalement un objet pour éviter qu'il ne soit trop proche d'éventuelles lignes supplémentaires.

```
\relative {
  c''4^"Word" c c''2
  R1
  \once \override TextScript.outside-staff-horizontal-padding = 1
  c,,4^"Word" c c''2
}
```



Voir aussi

Morceaux choisis : Section “Espacements” dans *Morceaux choisis*.
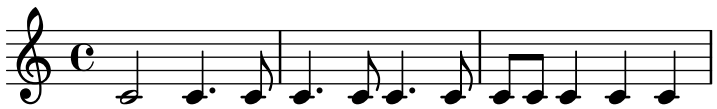
30 Espacement horizontal

30.1 Généralités sur l'espacement horizontal

Le moteur d'espacement traduit les différences de durée en distances étirables (*springs* pour ressorts) de différentes longueurs. Des durées importantes prennent ainsi plus de place que des durées moins longues. Les durées les plus courtes se verront attribuer un espace fixe, contrôlé par la propriété `shortest-duration-space` de l'objet Section "SpacingSpanner" dans *Référence des propriétés internes*. Au plus la durée s'allonge, au plus elle prendra d'espace : le doublement d'une durée attribuera à la note un espace fixé d'après la propriété `spacing-increment`.

L'exemple suivant comporte des blanches, des noires et un certain nombre de croches. La croche est suivie d'un espace de la largeur d'une tête de note ; pour la noire, cet espace est de deux têtes ; il est de trois pour la blanche.

```
\relative c' {
  c2 c4. c8
  c4. c8 c4. c8
  c8 c c4 c c
}
```



`spacing-increment` est normalement défini à 1,2 espace de portée – ce qui correspond à peu près à la largeur d'une tête de note – et `shortest-duration-space` à 2,0. La note la plus courte s'étendra donc sur l'équivalent de 2,4 espaces de portée (deux fois le `spacing-increment`). Le point de départ de cet espace se situe à l'extrémité gauche du symbole ; la note la plus courte est donc suivie en général d'un espace égal à la largeur d'une tête de note.
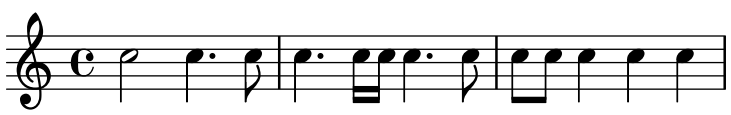
Si l'on suit à la lettre ce qui précède, ajouter une simple triple croche à une partition qui comporte déjà des croches et des doubles augmentera considérablement son volume : la durée la plus courte n'est plus la double mais la triple croche, ce qui aura pour conséquence d'ajouter une largeur de tête à chacune des notes. Pour s'affranchir de cet effet quelque peu pervers, la durée la plus courte prise en considération au niveau de l'espacement n'est pas la note la plus brève de la partition, mais celle qui apparaît le plus souvent.

La courte durée la plus fréquente est déterminée à partir de la note la plus courte de chaque mesure. C'est elle qui servira de base pour l'espacement, à cette nuance près que la plus courte durée ne saurait être strictement supérieure à la croche.

Ces durées peuvent être adaptées. Vous pouvez définir la durée de base pour les espacements grâce à la propriété `common-shortest-duration` de l'objet Section "SpacingSpanner" dans *Référence des propriétés internes*. La durée maximale de cet étalon, normalement la croche, est gérée par la propriété `base-shortest-duration`.

Les notes plus courtes que la note témoin sont suivies d'un espace proportionnel à la durée témoin. Si donc nous ajoutions quelques doubles croches à l'exemple précédent, elles seraient suivies d'une demie largeur de tête :

```
\relative { c''2 c4. c8 | c4. c16[ c] c4. c8 | c8 c c4 c c }
```
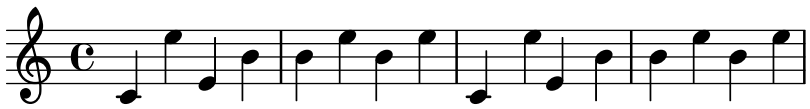


Comme le met en évidence notre *Essai sur la gravure musicale automatisée*, la direction des hampes peut influencer l'espacement – voir Section "Espacement" dans *Essai* et s'ajuster à l'aide

de la propriété `stem-spacing-correction` de l'objet Section "NoteSpacing" dans *Référence des propriétés internes* créé pour chaque contexte Section "Voice" dans *Référence des propriétés internes*.

L'objet `StaffSpacing`, généré au niveau d'un contexte Section "Staff" dans *Référence des propriétés internes*, possède une même propriété qui contrôlera l'espacement entre hampe et barre de mesure.

L'exemple suivant montre ces adaptations, tout d'abord selon les réglages par défaut, puis avec des corrections forcées.



L'espacement spécifique à la notation proportionnelle fait l'objet d'une Section 30.6 [Notation proportionnelle], page 746.

Voir aussi

Essai sur la gravure musicale automatisée : Section "Espacement" dans *Essai*.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SpacingSpanner" dans *Référence des propriétés internes*, Section "NoteSpacing" dans *Référence des propriétés internes*, Section "StaffSpacing" dans *Référence des propriétés internes*, Section "NonMusicalPaperColumn" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Il n'existe pas de mécanisme simple et efficace qui permette de forcer manuellement l'espacement. La solution ci-dessous permet cependant « d'aérer » artificiellement une partition ; il vous suffit d'ajuster la valeur du décalage *(padding)* autant que de besoin.

```
\override Score.NonMusicalPaperColumn.padding = 10
```
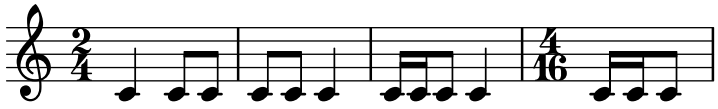
Il n'y a aucun moyen de diminuer l'espacement.

30.2 Changement d'espacement en cours de partition

Il arrive, au cours d'un même mouvement, qu'une nouvelle partie modifie substantiellement la notion de valeur brève et valeur longue. La commande `newSpacingSection` permet alors de réinitialiser les paramètres d'espacement.

Dans l'exemple qui suit, le changement de métrique marque le début d'une nouvelle partie ; remarquez comme les doubles-croches sont alors automatiquement un peu plus espacées :
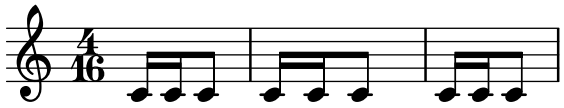
```
\time 2/4
c4 c8 c
c8 c c4 c16[ c c8] c4
\newSpacingSection
\time 4/16
c16[ c c8]
```



La commande `\newSpacingSection` crée un nouvel objet `SpacingSpanner` à cet instant musical. Si toutefois les ajustements apportés à l'espacement automatique ne se révèlent pas

satisfaisants, ils peuvent s'adapter à l'aide d'`\overrides`. Ces amendements doivent intervenir au même moment que la commande `\newSpacingSection` ; ils produiront leurs effets jusqu'à ce qu'ils soient à nouveau modifiés par une nouvelle section, comme ici :

```
\relative c' {
  \time 4/16
  c16[ c c8]
  \newSpacingSection
  \override Score.SpacingSpanner.spacing-increment = #2
  c16[ c c8]
  \newSpacingSection
  \revert Score.SpacingSpanner.spacing-increment
  c16[ c c8]
}
```



Voir aussi
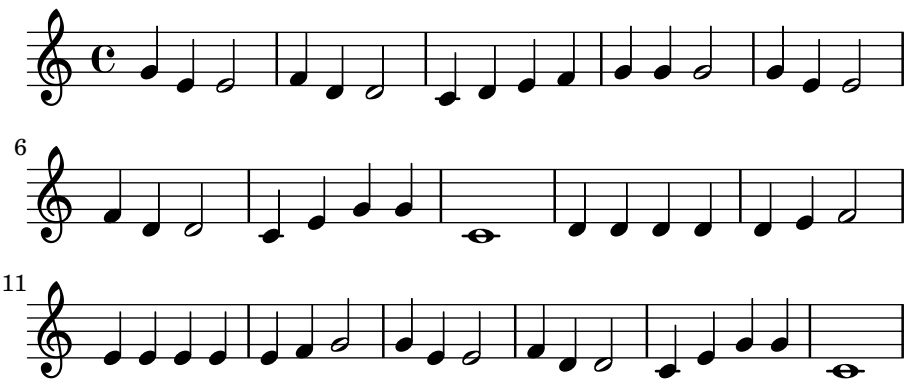
Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Référence des propriétés internes : Section "SpacingSpanner" dans *Référence des propriétés internes*.
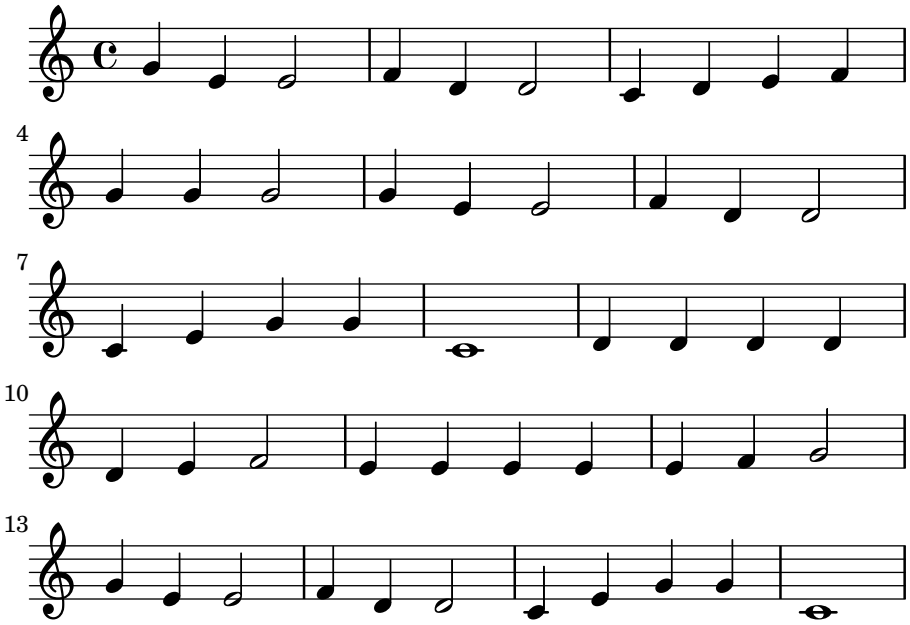
30.3 Modification de l'espacement horizontal global

Vous pouvez influencer l'espacement horizontal à l'aide de la propriété `base-shortest-duration`. Comparons les deux partitions qui suivent, toutes deux montrant la même musique. La première partition applique les réglages par défaut, alors que la seconde bénéficie d'un ajustement de la propriété `base-shortest-duration`. Au plus la valeur fournie en argument est grande, au plus la musique sera resserrée.

```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
}
```



```
\score {
  \relative {
    g'4 e e2 | f4 d d2 | c4 d e f | g4 g g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
    d4 d d d | d4 e f2 | e4 e e e | e4 f g2 |
    g4 e e2 | f4 d d2 | c4 e g g | c,1 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.base-shortest-duration =
        \musicLength 16
    }
  }
}
```

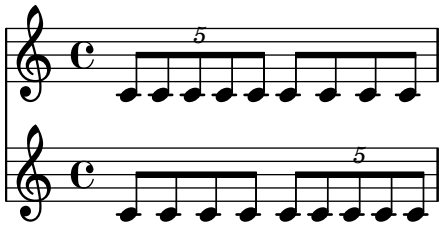


30.3.1 Étirement uniforme des n-olets

L'espacement au sein d'un n-olet dépend par défaut d'un certain nombre de facteurs qui ne sont pas liés à la durée (altération, changement de clef, etc.). `Score.SpacingSpanner.uniform-stretching` permet d'ignorer ces symboles et, par voie de conséquence, de forcer l'espacement sur la simple durée. Notez bien que cette propriété s'appliquera à toute la partition, puisque mentionnée au sein d'un bloc `\layout`.

```
\score {
  <<
    \new Staff \relative c' {
      \tuplet 5/4 { c8 c c c c } c8 c c c
    }
    \new Staff \relative c' {
      c8 c c c \tuplet 5/4 { c8 c c c c }
    }
  >>
  \layout {
    \context {
```

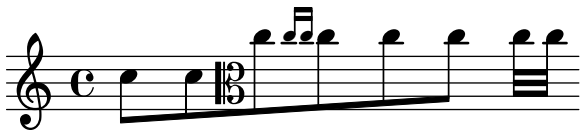
```
        \Score
        \override SpacingSpanner.uniform-stretching = ##t
      }
    }
  }
```



30.3.2 Espacement strict des notes

L'activation du commutateur `strict-note-spacing` permet d'espacer les notes sans tenir compte des clefs, barres de mesure ou notes d'ornement qui pourraient apparaître :

```
\override Score.SpacingSpanner.strict-note-spacing = ##t
\new Staff \relative {
  c''8[ c \clef alto c \grace { c16 c } c8 c c]  c32[ c] }
```



Voir aussi

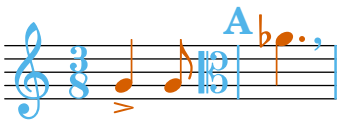
Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

30.4 Ajustement de l'espacement horizontal de certains objets de rendu

Au-delà des paramètres d'ordre général attachés à l'algorithme d'espacement par défaut et s'appliquant à tous les éléments de la partition ou d'une section particulière, plusieurs propriétés permettent d'opérer des ajustements sur la base du type d'objet. On peut citer par exemple la gestion de l'espace entre clef et métrique sans pour autant modifier quoi que ce soit entre une clef et les notes en l'absence de métrique, ou bien l'étalement des notes en présence d'un texte de sorte à éviter tout chevauchement. Les retouches en pareil cas nécessitent d'avoir tout d'abord identifié le type d'espace dont il est question.

30.4.1 Généralités sur l'ajustement de l'espacement de certains objets

Les objets de rendu qui ont une position horizontale fixe sont appelés des « éléments » (des *items*), par opposition aux « bandeaux » (les *spanners*). Pour les besoins de l'espacement horizontal, ils sont regroupés en empilements, également appelés colonnes. Têtes de note et silences, ainsi que les objets qui leur sont directement liés – altérations, articulations, hampes, points, etc. – font partie des « empilements musicaux » (représentés par des objets `NoteColumn`). Les éléments préfabriqués, tels que clefs, métriques et barres de mesure, sont regroupés en « empilements non musicaux » (représentés par des objets `NonMusicalPaperColumn`). Dans l'exemple ci-dessous, les éléments musicaux sont présentés en rouge, alors que les éléments non musicaux apparaissent en bleu.
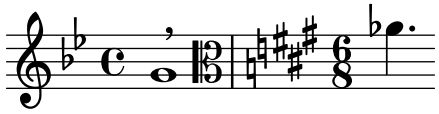


Cet exemple met en évidence l'alternance entre empilements musicaux et non musicaux. Le premier empilement non musical contient une clef et une métrique. Le premier empilement musical contient une tête de note avec sa hampe et son articulation. Le deuxième empilement non musical est vide ; il sera donc supprimé lors de la phase de mise en page. Le deuxième empilement musical contient à nouveau une note. Le troisième empilement non musical contient une clef, une barre de mesure et un repère, etc.

Au sein d'un même empilement, l'espacement est fixe. Par contre, le volume d'espace entre deux empilements consécutifs est flexible. Comme nous le verrons, les méthodes d'ajustement varient selon que l'on s'intéresse au contenu même d'un empilement ou à ce qui se trouve entre des empilements.

30.4.2 Espacement entre éléments adjacents non musicaux

Au sein d'un empilement non musical, les éléments sont disposés dans un ordre spécifique. Par exemple, si l'on considère le jeu d'éléments de l'image ci-dessous, l'ordonnancement par défaut placera d'abord le signe de respiration, puis la clef, la barre de mesure, les altérations d'annulation et l'armure, et enfin la métrique – ceci est contrôlé par la propriété `BreakAlignment .break-align-orders` qui peut être amendée à l'aide de `\breakAlignInsert`.

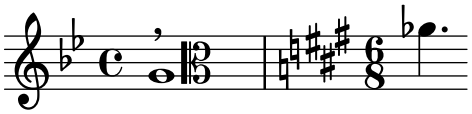
```
\relative {
  \key g \minor
  g'1
  \breathe
  \clef alto
  \time 6/8
  \key a \major
  aes4.
}
```



La distance entre deux éléments successifs d'un même empilement non musical est contrôlée par la valeur de la propriété `space-alist` de celui de gauche. `space-alist` prend la forme d'une liste associative mettant en correspondance les éléments non musicaux autorisant la survenue d'un saut avec des paires (*`style-d-espacement . valeur`*). Le type d'élément non musical est caractérisé par sa propriété `break-align-symbol`. Les possibilités standard sont listées dans Section "break-alignment-interface" dans *Référence des propriétés internes*, et les styles d'espacement dans Section "break-aligned-interface" dans *Référence des propriétés internes*. Parmi les options disponibles, seules `extra-space` et `minimum-space` sont pertinentes en matière de retouche de l'espacement entre des éléments non musicaux. Elles se distinguent par le fait que `extra-space` mesure l'espacement en partant de la droite du premier objet, alors que `minimum-space` compte dès la gauche du premier objet. L'un des moyens d'éloigner la barre de mesure de la clef est donc le suivant :

```
\relative {
  \key g \minor
  g'1
  \override Staff.Clef.space-alist.staff-bar = #'(extra-space . 4)
  \breathe
  \clef alto
  \time 6/8
  \key a \major
  aes4.
```

```
}
```



Les réglages de space-alist, qui ne se limitent pas aux styles d'espacement que nous venons de voir, permettent aussi d'apporter des dérogations à l'espacement entre différents empilements. Cette sorte d'espacement est néanmoins flexible et ne dépend pas uniquement des types d'objet concernés mais aussi de leur galbe. Des méthodes spécifiques à ce cas de figure sont abordées dans la section suivante.

Morceaux choisis

Séparation entre altérations annulées et nouvelle armure

Les altérations supprimées lors d'un changement de tonalité sont par défaut accolées à la nouvelle armure. Ce comportement peut s'adapter grâce à la propriété break-align-orders de l'objet BreakAlignment.

Si l'on se réfère à la référence des propriétés internes à propos de l'objet Section "break-alignment-interface" dans *Référence des propriétés internes*, l'ordre du deuxième élément est :

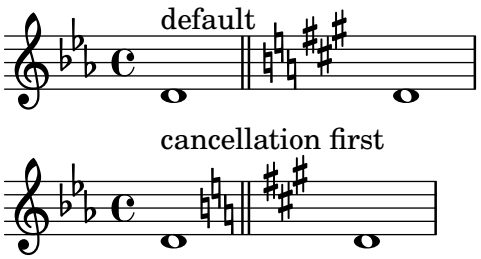
```
ambitus
breathing-sign
clef
cue-clef
cue-end-clef
custos
key-cancellation
key-signature
left-edge
signum-repetitionis
staff-bar
staff-ellipsis
time-signature
```

La consultation de cette liste nous apprend qu'il faut déplacer key-cancellation pour le mettre avant staff-bar. Ceci se réalise à l'aide de la fonction \breakAlignInsert.

```
music = { \key es \major d'1 \bar "||"
          \key a \major d'1 }

{ <>^\markup "default"
  \music }

{ <>^\markup "cancellation first"
  \breakAlignInsert key-cancellation before staff-bar
  \music }
```



Voir aussi

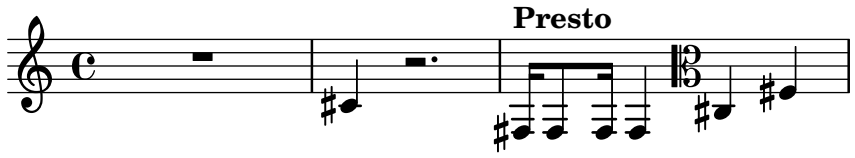
Manuel de notation : Section 36.9.4 [Utilisation de `break-aligned-interface`], page 826.

Manuel d'extension : Section “Listes associatives (alists)” dans *Extension de LilyPond*.

Référence des propriétés internes : Section “Break_align_engraver” dans *Référence des propriétés internes*, Section “BreakAlignGroup” dans *Référence des propriétés internes*, Section “BreakAlignment” dans *Référence des propriétés internes*, Section “break-alignable-interface” dans *Référence des propriétés internes*, Section “break-aligned-interface” dans *Référence des propriétés internes*, Section “break-alignment-interface” dans *Référence des propriétés internes*.

30.4.3 Espacement entre empilements adjacents

Contrairement à l'espacement au sein d'un empilement, l'espacement entre des empilements adjacents est flexible ; il s'étire ou se comprime selon la densité de la musique présente sur la ligne. Par défaut, il arrive même que, dans certaines situations, des empilements se chevauchent. L'exemple suivant illustre trois cas. La deuxième altération se déporte sous la barre de mesure, et la troisième déborde sous la clef. Par ailleurs, l'indication de tempo *Presto* s'étend sur plusieurs empilements. Observez comment la première altération qui, en raison de son positionnement vertical, viendrait chevaucher la barre de mesure si elle était placée aussi près que la seconde altération, en est écartée.

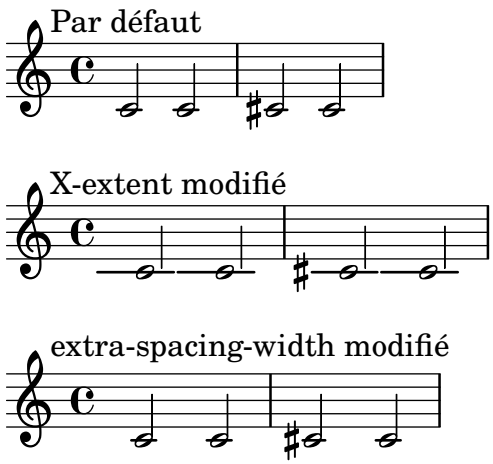


Ces règles d'espacement peuvent se modifier. Il faut à la fois modifier la largeur et la hauteur qu'un objet occupe dans l'espacement horizontal. Les propriétés utiles en la matière sont `extra-spacing-width` et `extra-spacing-height`. Lorsqu'elles ne sont pas activées, un objet occupera autant de place dans l'espacement horizontal que ses propriétés `X-extent` et `Y-extent` le permettent. Ce sont les valeurs précises de ses dimensions. Les propriétés `extra-spacing-width` et `extra-spacing-height` rendent l'objet plus ou moins imposant dans le calcul de l'espacement horizontal uniquement entre les empilements, mais préservent ses dimensions pour les autres types d'espacement.

```
{
  \textMark "Par défaut"
  c'2 2 cis'2 2
}

{
  \textMark "X-extent modifié"
  \override NoteHead.X-extent = #'(-2 . 2)
  c'2 2 cis'2 2
}

{
  \textMark "extra-spacing-width modifié"
  \override NoteHead.extra-spacing-width = #'(-2 . 2)
  c'2 2 cis'2 2
}
```
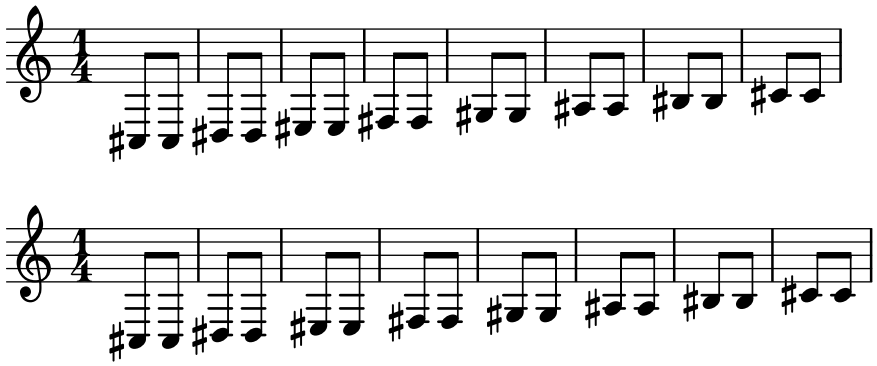


Les valeurs de `extra-spacing-width` et `extra-spacing-height` sont des paires de nombres qui seront ajoutés aux dimensions dans chacun des axes. Par exemple, régler `extra-spacing-height` à `'(-2 . 3)` agrandira l'objet de trois unités vers le haut et de deux unités vers le bas (limite abaissée de 2). L'exemple suivant illustre comment utiliser `extra-spacing-height` pour changer la limite à partir de laquelle les altérations ne chevaucheront plus les barres de mesure.

```
music = \relative {
  \time 1/4
  cis8 8 | dis8 8 | eis8 8 | fis8 8 |
  gis8 8 | ais8 8 | bis8 8 | cis8 8 |
}

{
  \music
}
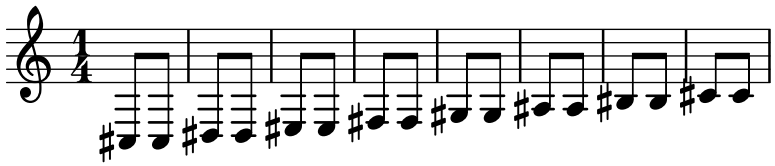
{
  \override Accidental.extra-spacing-height = #'(0 . 1.0)
  \music
}
```



Affecter la valeur `'(+inf.0 . -inf.0)` à `extra-spacing-width` ou `extra-spacing-height` revient à supprimer la présence de l'objet.
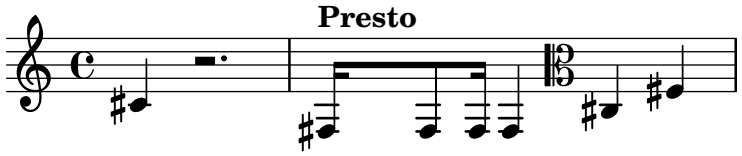
```
music = \relative {
  \time 1/4
  cis8 8 | dis8 8 | eis8 8 | fis8 8 |
  gis8 8 | ais8 8 | bis8 8 | cis8 8 |
}

{
  \override Accidental.extra-spacing-height = #'(+inf.0 . -inf.0)
  \music
}
```



Par contre, un `extra-spacing-height` de valeur `'(-inf.0 . +inf.0)` rend l'objet infiniment haut, ce qui lui évitera dans tous les cas de chevaucher un autre empilement. L'exemple suivant illustre cette technique pour `Accidental` et `MetronomeMark`. En ce qui concerne `MetronomeMark`, il est tout d'abord nécessaire de régler `extra-spacing-width` à `'(0 . 0)` dans la mesure où elle est par défaut à `'(+inf.0 . -inf.0)`, et même un objet infiniment haut ne prend pas de place s'il n'a pas de largeur.

```
{
  \override Score.MetronomeMark.extra-spacing-width =
    #'(0 . 0)
  \override Score.MetronomeMark.extra-spacing-height =
    #'(-inf.0 . +inf.0)
  \override Accidental.extra-spacing-height =
    #'(-inf.0 . +inf.0)
  cis'4 r2.
  \tempo Presto
  fis16 8 16 4 \clef alto cis4 fis4
}
```



Voir aussi

Référence des propriétés internes : Section “item-interface” dans *Référence des propriétés internes*, Section “separation-item-interface” dans *Référence des propriétés internes*.

30.5 Largeur de ligne

Deux réglages de base ont une influence considérable sur l'espacement : `line-width` et `indent`. Tous deux se placent dans le bloc `\layout`. Ils contrôleront la longueur des lignes et l'indentation de la première.

L'activation du commutateur `ragged-right` au sein du bloc `\layout` permet de terminer les systèmes naturellement plutôt que de les voir s'étirer sur toute la largeur de la page. Cette option est particulièrement utile lorsque vous traitez de courts fragments, ou bien pour vérifier

ce que donnerait l'espacement naturel. Bien qu'il soit désactivé par défaut, il sera activé si la partition ne comporte qu'un seul système.

Le fonctionnement de l'option `ragged-last` est en tout point identique à celui de `ragged-right`, à ceci près qu'il ne concerne que la dernière ligne de la partition. Il n'y a pas de restriction quant à cette ligne. Il en va de même que pour le formatage d'un paragraphe de texte, la dernière ligne s'arrête au dernier caractère.

```
\layout {
  indent = 0
  line-width = 150
  ragged-last = ##t
}
```

Voir aussi

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

30.6 Notation proportionnelle

LilyPond prend en charge la notation proportionnelle. Il s'agit dans ce cas de représenter la notation selon un espacement strictement relatif aux durées. Ce type d'espacement pourrait se comparer à l'utilisation de papier millimétré pour positionner les notes au fil de la portée. Certaines œuvres de la fin du XXe siècle et à l'aube du XXIe utilisent cette proportionnalité dans le but de clarifier des structures rythmiques complexes, d'aider au positionnement d'indications temporelles ou autres éléments graphiques directement dans la partition.
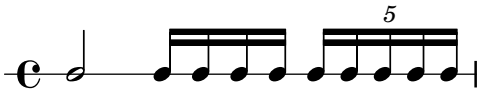
LilyPond met à votre disposition différents réglages, qui peuvent s'utiliser conjointement ou individuellement, aux fins de mettre au point cette notation proportionnelle.

- `proportionalNotationDuration`
- `uniform-stretching`
- `strict-note-spacing`
- `\remove Separating_line_group_engraver`

Nous allons examiner, dans les différents exemples qui suivent, les effets de ces réglages et comment ils interagissent.

Commençons par cette mesure toute simple qui utilise l'espacement classique et justifiée à gauche.

```
\new RhythmicStaff {
  c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
}
```

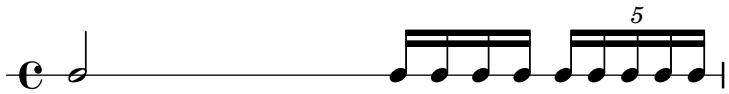


Vous constatez que la blanche qui entame la mesure prend moins de la moitié de l'espace. De même, les doubles croches et le quintolet de doubles (donc des vingtièmes de ronde) qui terminent cette mesure n'en occupent pas la moitié de l'espace horizontal.

En matière de gravure traditionnelle, cet espacement correspond tout à fait à nos attentes, puisque nous pouvons rogner de l'espace sur la blanche et ainsi gagner en largeur sur toute la mesure qui fait une ronde.

Par contre, si nous avons besoin d'insérer une indication temporelle ou un autre graphisme en surplomb ou en dessous de notre partition, nous aurons besoin de la notation proportionnelle. Celle-ci s'active en définissant la propriété `proportionalNotationDuration`.

```
\new RhythmicStaff {
  c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
}
\layout {
  \context {
    \Score
    proportionalNotationDuration = #1/20
  }
}
```



La blanche du début et les notes plus rapides de la deuxième moitié de la mesure occupent maintenant exactement le même espace horizontal. Nous pourrions donc y insérer, au-dessus ou au-dessous, une indication temporelle ou autre graphisme.

`proportionalNotationDuration` est une propriété attachée au contexte `Score`. Rappelez-vous que vous pouvez régler les propriétés d'un contexte à trois différents endroits de votre fichier : dans un bloc `\with`, dans un bloc `\context` ou au beau milieu de la musique à l'aide de la commande `\set`. Vous pouvez donc définir `proportionalNotationDuration` selon l'une de ces trois façons, à l'instar de n'importe quelle définition de contexte.

La propriété `proportionalNotationDuration` prend en unique argument la durée de référence qui servira de base pour espacer toute la musique. Une valeur de `#1/20` produit donc une durée de référence égale à un vingtième de ronde.

Se pose alors le problème de fournir la juste durée de référence à `proportionalNotationDuration`. Il faut en l'occurrence procéder par tâtonnement, en commençant par une valeur proche de la note la plus rapide (la durée la plus courte) du morceau. Au plus la durée de référence est petite, au plus la musique sera étalée ; à l'inverse, une durée de référence élevée produira une musique resserrée.

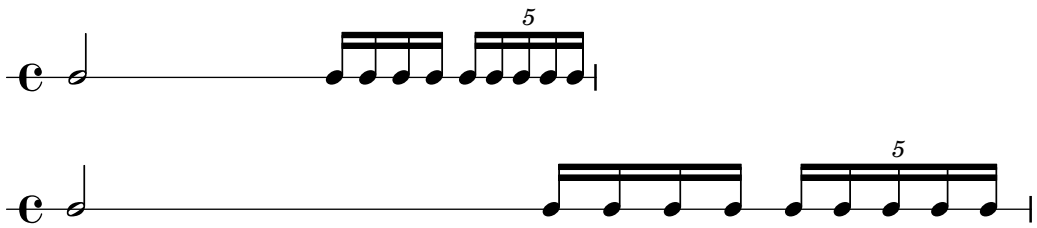
```
rhythm = { c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 } }

\new RhythmicStaff {
  \set Score.proportionalNotationDuration = #1/8
  % Allow overlapping of note heads.
  \override NoteHead.extra-spacing-width = #'(+inf.0 . -inf.0)
  \rhythm
}

\new RhythmicStaff {
  \set Score.proportionalNotationDuration = #1/16
  \rhythm
}

\new RhythmicStaff {
  \set Score.proportionalNotationDuration = #1/32
  \rhythm
}
```

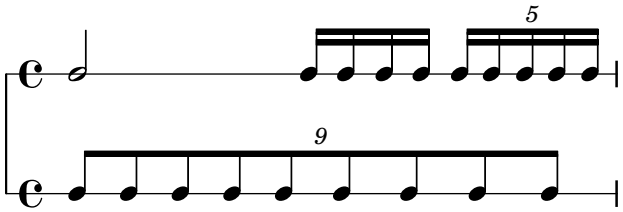




Vous ne manquerez pas de noter qu'une durée de référence trop grande, comme la croche pour la première ligne, a pour conséquence de resserrer la musique, ce qui peut aboutir à des chevauchements de têtes. Vous remarquez aussi que, par principe, la notation proportionnelle occupe beaucoup plus d'espace horizontal que l'espacement traditionnel. La notation proportionnelle met en évidence le rythme au détriment de l'espacement horizontal.

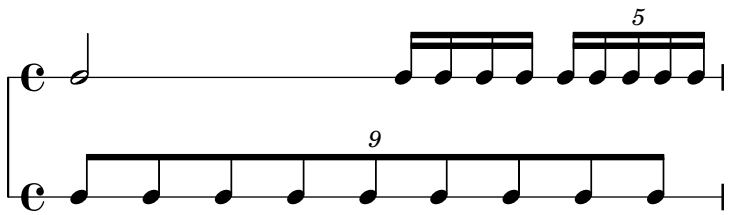
Examinons à présent le moyen d'optimiser l'espacement de n-olets en tuilage. Reprenons notre exemple de départ, avec son espacement traditionnel, et ajoutons lui une portée incluant un autre type de n-olet.

```
<<
  \new RhythmicStaff {
    c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
  \new RhythmicStaff {
    \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
  }
>>
```



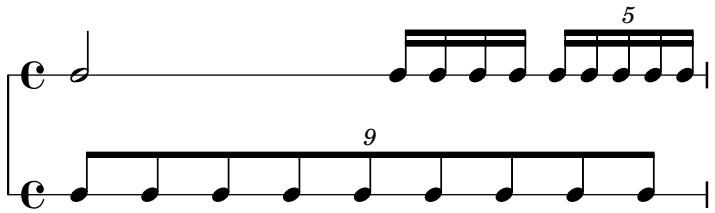
L'espacement est loin d'être idéal, pour la simple raison que l'espacement régulier des notes de la portée inférieure ne s'étire pas uniformément. Il est vrai que de telles constructions complexes en n-olets sont assez rares en gravure traditionnelle, ce qui explique que les règles qu'elle applique peuvent amener à ce résultat. Le recours à `proportionalNotationDuration` permet d'arranger les choses.

```
<<
  \new RhythmicStaff {
    c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
  \new RhythmicStaff {
    \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
  }
>>
\layout {
  \context {
    \Score
    proportionalNotationDuration = #1/20
  }
}
```



Cependant, si l'on observe de près, il est évident que les notes de la deuxième moitié du ennaolet ont tendance à s'espacer légèrement plus que celles de la première moitié. Afin d'uniformiser cet étalement, nous allons activer le `uniform-stretching`, propriété attachée au `SpacingSpanner`.

```
<<
  \new RhythmicStaff {
    c2 16 16 16 16 \tuplet 5/4 { 16 16 16 16 16 }
  }
  \new RhythmicStaff {
    \tuplet 9/8 { c8 8 8 8 8 8 8 8 8 }
  }
>>
\layout {
  \context {
    \Score
    proportionalNotationDuration = #1/20
    \override SpacingSpanner.uniform-stretching = ##t
  }
}
```



L'espacement sur les deux portées est maintenant correct, les relations rythmiques sont clairement perceptibles, et nous pourrions y insérer une indication temporelle ou autre graphisme selon notre envie.

Il est préférable de toujours activer la propriété `uniform-stretching` du `SpacingSpanner` lorsqu'on fait appel à `proportionalNotationDuration`. Dans le cas contraire, un espacement erroné pourrait se produire en présence de silences invisibles *skip*.

Le `SpacingSpanner` est en fait un objet graphique abstrait attaché au contexte `Score`. Tout comme pour la propriété `proportionalNotationDuration`, les réglages du `SpacingSpanner` peuvent se faire à trois différents endroits de votre fichier : dans un bloc `\with` attaché à `\Score`, dans un bloc `\context` ou au beau milieu de la musique à l'aide de la commande `\set`.

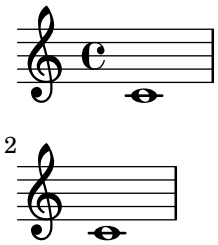
Gardez bien à l'esprit qu'il n'y a par défaut qu'un seul `SpacingSpanner` par `Score`. Il s'ensuit que `uniform-stretching` est soit activé, soit désactivé, et dans tous les cas pour l'intégralité de la partition. Vous pourriez toutefois avoir besoin de modifier ce comportement en cours de partition, et recourir alors à l'instruction `\newSpacingSection` – pour de plus amples détails, voir la rubrique Section 30.2 [Changement d'espacement en cours de partition], page 737.

Intéressons-nous maintenant au `Separating_line_group_engraver`, qui est désactivé pour la plupart des partitions en notation proportionnelle. Voici ce qui apparaît dans une partition

traditionnelle : il y a toujours un « espace préservé » juste avant la première note de chaque portée.

```
\paper {
  indent = 0
}

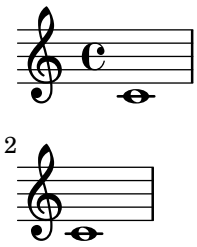
\new Staff {
  c'1 \break
  c'1
}
```



Cet espace, géré par le `Separating_line_group_engraver`, est aussi présent lorsqu'intervient un changement de métrique, d'armure ou de clef. Désactiver le `Separating_line_group_engraver` revient à réduire cet espace à zéro.

```
\paper {
  indent = 0
}

\new Staff \with {
  \remove Separating_line_group_engraver
} {
  c'1 \break
  c'1
}
```



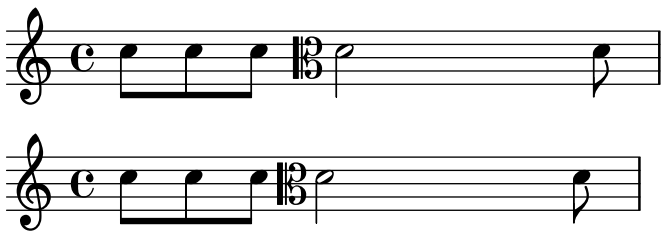
Les éléments non musicaux tels que métrique, armure, clef et altérations, posent problème lorsqu'on travaille en notation proportionnelle. Bien qu'aucune notion de durée ne leur soit attachée, ces éléments « consomment » de l'espace. Différentes approches permettent cependant de gérer ce problème.

Éviter les problèmes d'espacement avec l'armure est chose aisée : il suffit qu'il n'y en ait pas ! C'est bien souvent le cas en musique contemporaine, où l'on trouve le plus d'ouvrages en notation proportionnelle. Il en va de même pour la métrique, et tout particulièrement lorsque la partition comporte un quadrillage temporel ou autres graphismes. L'absence de métrique reste cependant exceptionnelle et la plupart des partitions en notation proportionnelle laissent apparaître quelques métriques. Il est par contre pratiquement impossible de se passer de clef et d'altération.

L'une des options permettant de s'affranchir de l'espacement dû aux éléments non musicaux consiste en l'activation de la propriété `strict-note-spacing` attachée au `SpacingSpanner`. Observons les deux portées suivantes :

```
{
  \set Score.proportionalNotationDuration = #1/16
  c''8 8 8 \clef alto d'2 d'8
}

{
  \set Score.proportionalNotationDuration = #1/16
  \override Score.SpacingSpanner.strict-note-spacing = ##t
  c''8 8 8 \clef alto d'2 d'8
}
```



Toutes deux affichent un espacement proportionnel. Cependant, la première ligne laisse apparaître un espacement plus lâche en raison de la présence d'un changement de clef. En ce qui concerne la deuxième ligne, l'espacement est strictement observé dès lors que la propriété `strict-note-spacing` a préalablement été activée. Comme vous pouvez le constater, l'activation de `strict-note-spacing` a pour conséquence que l'algorithme d'espacement ignore tout bonnement la largeur des métriques, armures, clefs et altérations.

En plus de ceux que nous venons de voir, vous trouverez d'autres réglages en usage dans la notation proportionnelle.

`\override SpacingSpanner.strict-grace-spacing = ##t`
: espace les notes d'ornement de manière stricte (voir [Positionnement des notes d'ornement avec espace flottant], page 154)

`\set tupletFullLength = ##t`
: étend le crochet de n-olet pour indiquer rythmiquement ses points de début et de fin

`\override Beam.breakable = ##t`
: permet d'interrompre des ligatures (voir [Ligature au moment d'un saut de ligne], page 105)

`\override Glissando.breakable = ##t`
: permet d'interrompre des glissandos (voir [Saut de ligne et glissando], page 183)

`\remove Forbid_line_break_engraver`
: autorise les sauts de lignes même lorsqu'un élément musical est encore actif (voir Section 28.1 [Sauts de ligne], page 711)

Voir aussi

Manuel de notation : Section 30.2 [Changement d'espacement en cours de partition], page 737.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

31 Réduction du nombre de pages de la partition

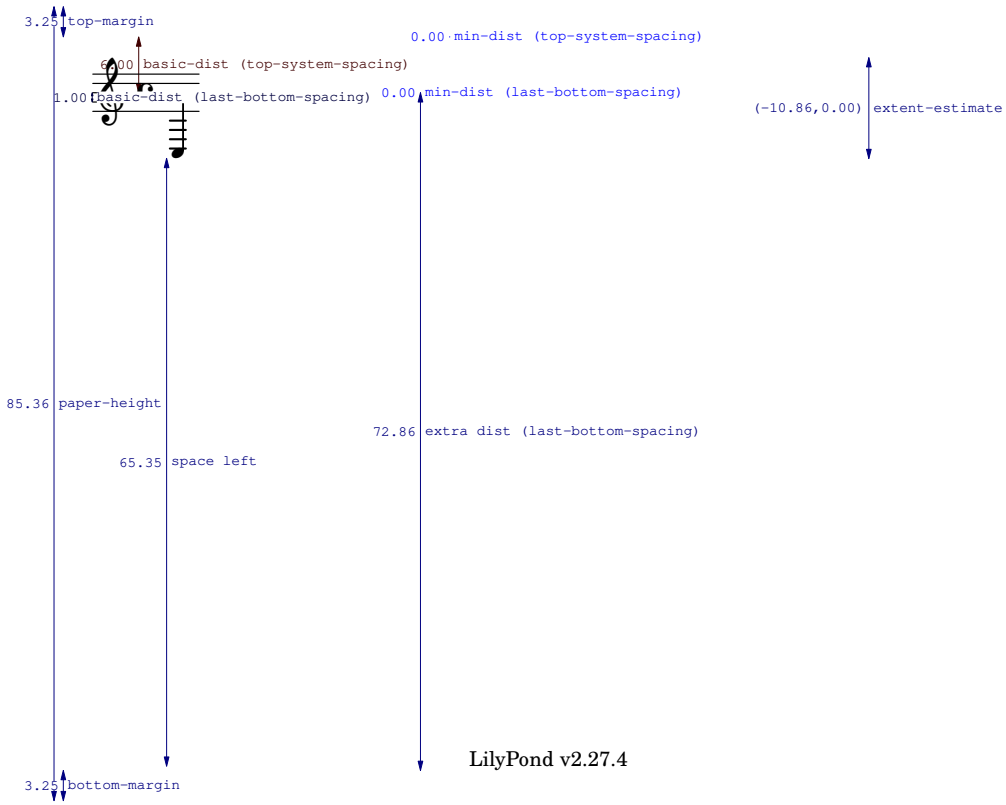
Vous pourriez un jour être confronté au problème suivant : l'une des pages de votre partition ne comporte que deux portées alors que, ce qui est d'autant plus frustrant, l'espace libre sur les autres pages aurait permis une distribution différente.

L'instruction `annotate-spacing` se révèle être un outil indispensable pour l'analyse des problèmes de mise en forme. Cette commande met en surimpression la valeur des différentes variables d'espacement et de mise en forme, comme nous allons le voir dans la rubrique Section 31.1 [Mise en évidence de l'espacement], page 752.

31.1 Mise en évidence de l'espacement

Le meilleur moyen d'appréhender les différentes variables de dimensionnement vertical sur lesquelles vous pouvez jouer au niveau de la mise en page consiste à activer, au sein du bloc `\paper`, la fonction `annotate-spacing` :

```
\book {
  \score { { c4 } }
  \paper { annotate-spacing = ##t }
}
```



Toutes les dimensions sont exprimées en espace de portée *(staff space)* quelle que soit l'unité mentionnée dans les blocs `\paper` ou `\layout`. Dans cet exemple, la hauteur de la feuille (`paper-height`) est de 59,75 espaces de portée et la taille de portée (`staff-size`) de 20 points – sa valeur par défaut. Notez que :

1 point = (25,4/72,27) mm

1 staff-space = (`staff-size`)/4 pts

= (staff-size)/4 * (25,4/72,27) mm

Dans le cas qui nous occupe, un espace de porté égale environ 1,757 millimètres. Les 59,75 espaces de portée de paper-height correspondent donc à 105 millimètres, soit la hauteur d'une feuille au format A6 à l'italienne. Les paires (a,b) sont des intervalles, *a* en étant l'extrémité inférieure et *b* l'extrémité supérieure.

Voir aussi

Manuel de notation : Section 27.2 [Définition de la taille de portée], page 707.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

31.2 Modification de l'espacement

Les informations fournies par annotate-spacing en matière de dimensionnement vertical sont incomparables. Pour plus de détails sur les manières de modifier marges et autres variables connexes, consultez la rubrique Chapitre 26 [Mise en forme de la page], page 691.

En dehors des marges, vous disposez de quelques moyens supplémentaires pour gagner de l'espace :

- Rapprocher les systèmes le plus possible les uns des autres, de telle sorte qu'il en tienne un maximum sur une même page, tout en les espaçant suffisamment pour éviter le blanc en bas de page.

```
\paper {
  system-system-spacing = #'((basic-distance . 0.1) (padding . 0))
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

- Forcer le nombre de systèmes par page. Ceci peut se révéler judicieux à deux titres. D'une part, le fait de définir un nombre de systèmes – même s'il est égal à la valeur par défaut – peut aboutir à plus de systèmes par page dans la mesure où l'une des étapes d'estimation des hauteurs est tout simplement sautée. D'autre part, réduire le nombre de systèmes par page permet d'en disposer plus sur les suivantes. Par exemple, avec un nombre par défaut de 11 systèmes par page, l'instruction suivante le force à 10.
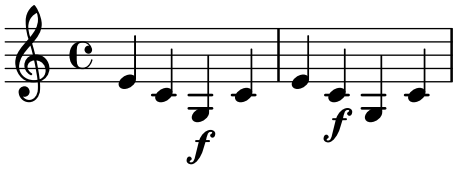
```
\paper {
  system-count = 10
}
```

- Forcer le nombre de pages. L'instruction suivante forcera la musique à se répartir sur deux pages.

```
\paper {
  page-count = 2
}
```
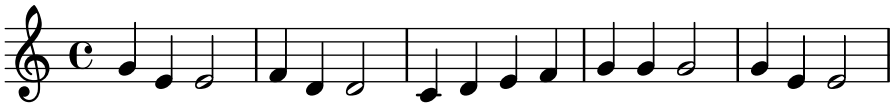
- Éviter ou réduire les objets qui augmentent la hauteur des systèmes. Par exemple, un crochet d'alternative en fin de reprise consomme de l'espace. Dans le cas où il s'étend sur deux systèmes, il occupera plus d'espace que si seul le premier comportait l'indication. Autre exemple, les indications de nuance qui se « détachent » d'un système peuvent être rapprochées de la portée :

```
\relative e' {
  e4 c g\f c
  e4 c g-\tweak X-offset -2.7 \f c
}
```
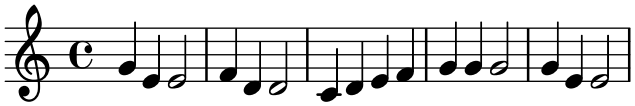


- Modifier l'espacement horizontal à l'aide du SpacingSpanner, comme indiqué à la rubrique Section 30.3 [Modification de l'espacement horizontal global], page 738. Voici ce que donne l'espacement par défaut :

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
}
```



Par contre, le fait de modifier la valeur de la propriété common-shortest-duration en passant de 1/4 à 1/2 – bien que la noire soit la durée la plus courante, nous prenons une valeur plus longue – donnera un effet « resserré » à la musique :

```
\score {
  \relative {
    g'4 e e2 |
    f4 d d2 |
    c4 d e f |
    g4 g g2 |
    g4 e e2 |
  }
  \layout {
    \context {
      \Score
      \override SpacingSpanner.common-shortest-duration =
        \musicLength 2
    }
  }
}
```



La propriété common-shortest-duration ne peut être modifiée dynamiquement. Elle se place toujours dans un bloc \context et s'applique à l'intégralité de la partition.

Voir aussi

Manuel de notation : Chapitre 26 [Mise en forme de la page], page 691, Section 30.3 [Modification de l'espacement horizontal global], page 738.

Morceaux choisis : Section "Espacements" dans *Morceaux choisis*.

Modification des réglages prédéfinis

32 Personnalisation des sorties

LilyPond est conçu pour générer, par défaut, des partitions de la plus haute qualité. Cependant, on peut parfois avoir à modifier cette mise en forme par défaut. Celle-ci est réglée par tout un ensemble de « leviers et manettes » plus connus sous le terme de « propriétés », dont ce chapitre ne cherche pas à faire l'inventaire exhaustif – le chapitre Section “Retouche de partition” dans *Manuel d'initiation* du manuel d'initiation vous en propose un aperçu. Le propos est plutôt ici de mettre en évidence les différents groupes auxquels s'apparentent ces contrôles et d'expliquer comment trouver le bon levier pour obtenir tel ou tel effet en particulier.

Les moyens de contrôle des différents réglages sont décrits dans un document séparé, *Référence des propriétés internes*. Ce guide répertorie toutes les variables, fonctions et autres options que LilyPond met à votre disposition.

En sous-main, LilyPond se sert du langage Scheme (un dérivé du Lisp) comme infrastructure. Modifier les choix de mise en page revient à pénétrer dans les entrailles du programme, et de ce fait requiert l'emploi du Scheme. Les fragments de Scheme, dans un fichier `.ly`, sont introduits par le caractère *hash* (#), improprement surnommé « dièse ».[1]

[1] Le Section “Tutoriel Scheme” dans *Extension de LilyPond* fournit quelques notions de base pour saisir des nombres, des listes, des chaînes de caractères ou des symboles, en Scheme.

33 Contextes d'interprétation

Nous allons voir ici ce que sont les contextes et comment les modifier.

Voir aussi

Manuel d'initiation : Section "Contextes et graveurs" dans *Manuel d'initiation*.

Fichiers d'initialisation : `ly/engraver-init.ly`, `ly/performer-init.ly`.

Morceaux choisis : Section "Contextes et graveurs" dans *Morceaux choisis*.

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "Engravers and Performers" dans *Référence des propriétés internes*.

33.1 Tout savoir sur les contextes

Les contextes sont hiérarchisés.

33.1.1 Définitions de la sortie – hiérarchie des contextes

Les lignes qui suivent traitent de l'intérêt des définitions de sorties lorsque l'on travaille avec les *contextes*. Des exemples de définitions seront présentés plus avant – voir Section 33.5.1 [Modification de tous les contextes d'un même type], page 771.

Alors que la musique écrite dans un fichier fait référence à des types ou noms de contexte, les contextes ne sont effectivement créés que lorsque la musique est interprétée. LilyPond interprète la musique sous le contrôle d'une *définition de sortie*, voire différemment selon le cas, et génère ainsi différents résultats. La définition de sortie appropriée pour une sortie imprimable est spécifiée à l'aide d'un `\layout`.

Une définition de sortie beaucoup plus simple sera utilisée pour produire une sortie MIDI, spécifiée à l'aide d'un `\midi`. LilyPond utilise en interne plusieurs autres définitions de sortie, notamment dans le cadre du combinateur automatique de parties (voir Section 5.2.5 [Regroupement automatique de parties], page 238) ou la reproduction d'extraits (voir Section 6.3.2 [Citation d'autres voix], page 274).

Les définitions de sortie ont pour objet non seulement de définir la relation entre les contextes, mais aussi leurs réglages par défaut. Si la plupart des adaptations prennent habituellement place au sein d'un bloc `\layout`, les réglages affectant le MIDI ne seront effectifs que s'ils interviennent au sein d'un bloc `\midi`.

Certains réglages affectent plusieurs sorties : par exemple, lorsque `autoBeaming` est désactivé dans un contexte, les ligatures sont considérées comme marquant un mélisme dans le but de faire correspondre la musique aux paroles comme indiqué dans Section 9.1.4 [Durée automatique des syllabes], page 359. Cette correspondance est respectée autant à l'écrit qu'à l'oral. Des modifications apportées à `autoBeaming` par une définition de contexte au sein d'un bloc `\layout` ne seront pas reportées dans le bloc `\midi` correspondant ; paroles et musique ne seront alors plus synchrones dans le fichier MIDI.

Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`, `ly/performer-init.ly`.

33.1.2 `Score` – le père de tous les contextes

Il s'agit en l'occurrence du contexte le plus élevé, autrement dit le plus important, en matière de notation.[1] En effet, c'est au niveau de la partition – *score* en anglais – que se gèrent le temps et

[1] Il existe bien un contexte nommé `Global` au-dessus de `Score`, codé en dur et point d'entrée pour LilyPond. Néanmoins, il n'est pas destiné à être modifié, et sera utile uniquement pour la construction d'un graveur en Scheme. On peut donc l'ignorer en règle générale.

la tonalité ; c'est donc là qu'il faut s'assurer que les différents éléments, tels les clefs, métriques et armures sont bien répercutés sur toutes les portées.

On ne peut créer explicitement de contexte `Score` dans la mesure où il n'est contenu dans aucun autre contexte. Dès lors que LilyPond rencontre un bloc `\score {...}` se crée implicitement un contexte `Score`.

Un alias nommé `Timing` est établi par le `Timing_translator` quel que soit le contexte qui l'initialise, et les variables de temps sont alors recopiées de là où `Timing` a précédemment été établi. L'alias au niveau de `Score` fournit une cible pour l'initialisation des variables de `Timing` dans les définitions de mise en forme avant même qu'un quelconque `Timing_translator` ne soit lancé.

33.1.3 Contextes de haut niveau – les systèmes

`StaffGroup`
: Le groupe de portées est attaché par un crochet et les barres de mesure sont d'un seul tenant, de la première à la dernière portée.

`ChoirStaff`
: Ce regroupement est identique au `StaffGroup`, à ceci près que les barres de mesure ne traversent pas l'espace inter-portées.

`GrandStaff`
: Le groupe de portées est attaché par une accolade sur la gauche et les barres de mesure sont d'un seul tenant.

`PianoStaff`
: Ce regroupement est identique au `GrandStaff`, à ceci près que ses portées ne pourront être masquées que conjointement, jamais individuellement.

`OneStaff` Ce regroupement fournit un axe commun pour les portées qu'il englobe, les faisant ainsi apparaître dans un même espace vertical. Ceci peut s'avérer utile dans le cas d'un succession de portées d'aspect différent ou pour temporairement changer l'aspect d'une portée ou la recouvrir d'une autre. On l'utilise souvent en combinaison avec `\stopStaff` et `\startStaff` pour de meilleurs résultats.

`VaticanaScore`
: Contexte de plus haut niveau remplaçant `Score` pour noter du grégorien selon le style vatican. Par rapport à `Score`, les lignes apparaissent en rouge, l'espacement est plus resserré et les mesures ne sont pas numérotées.

`ChordGridScore`
: Contexte de plus haut niveau remplaçant `Score` pour noter des grilles harmoniques. Par rapport à `Score`, il utilise la notation proportionnelle et dispose de quelques réglages particuliers tels que la suppression des numéros de mesure.

`StandaloneRhythmScore`
: Contexte de niveau `Score` dévolu à `\markup \rhythm`.

33.1.4 Contextes de niveau intermédiaire – les portées

`Staff` La portée prend en charge les clefs, barres de mesure, armures et les altérations accidentelles. Un contexte `Staff` peut contenir plusieurs contextes `Voice`.

`RhythmicStaff`
: De même nature qu'un `Staff`, ce contexte est destiné à n'imprimer que du rythme. Quelle que soit la hauteur, les notes seront imprimées sur une même et unique ligne ; la sortie MIDI rendra les hauteurs saisies.

`TabStaff` Ce contexte permet de générer des tablatures. La mise en forme par défaut correspond à une tablature pour guitare, sur six lignes. Un contexte `TabStaff` ne peut contenir que des contextes `TabVoice`.

`DrumStaff`
Contexte dévolu tout spécialement aux parties de percussion ; il peut contenir plusieurs `DrumVoice`.

`VaticanaStaff`
Identique au contexte `Staff`, à ceci près qu'il est tout particulièrement adapté au grégorien.

`MensuralStaff`
Identique au contexte `Staff`, à ceci près qu'il est tout particulièrement adapté au style mensural de musique ancienne.

`PetrucciStaff`
Un type de `Staff` cherchant à reproduire les préceptes édictés par Ottaviano Petrucci dans *Harmonices Musices Odhecaton* (Venise, 1501).

`KievanStaff`
Identique au contexte `Staff`, à ceci près qu'il est tout particulièrement adapté au style de notation kiévienne.

`GregorianTranscriptionStaff`
Portée destinée à noter du grégorien dans un style moderne.

`ChordGrid`
Crée une notation en grilles harmoniques. Ce contexte est toujours attaché à un contexte `ChordGridScore`.

`StandaloneRhythmStaff`
Contexte de niveau `Staff` dévolu à `\markup \rhythm`.

`FretBoards`
Contexte permettant d'afficher des diagrammes de fret.

`Devnull` Passe sous silence toutes les informations musicales contenues dans ce contexte.

33.1.5 Contextes de bas niveau – les voix

Les contextes de niveau « voix » initialisent un certain nombre de propriétés et activent les graveurs appropriés. Un contexte de bas niveau est un contexte n'ayant aucun contexte enfant – ou `\defaultchild`. Bien qu'ils puissent accepter ou contenir des sous-contextes, ceux-ci devront être libellés et créés explicitement.

`Voice` Correspond à une voix positionnée sur une portée. Le contexte `Voice` s'occupe des indications de nuance, des hampes, des ligatures, des scripts placés au-dessus ou au-dessous de la portée, des différentes liaisons et des silences.

Lorsque plusieurs voix doivent cohabiter sur la même portée, il est indispensable de les instancier explicitement.

`VaticanaVoice`
Fonctionnant comme le contexte `Voice`, il est tout particulièrement destiné à gérer le grégorien.

`MensuralVoice`
Fonctionnant comme le contexte `Voice`, il est tout particulièrement adapté aux musiques anciennes.

`PetrucciVoice`
: Un type de `Voice` cherchant à reproduire les préceptes édictés par Ottaviano Petrucci dans *Harmonices Musices Odhecaton* (Venise, 1501).

`KievanVoice`
: Identique au contexte `Voice`, à ceci près qu'il est tout particulièrement adapté au style de notation kiévienne.

`GregorianTranscriptionVoice`
: Voix destinée à noter du grégorien dans un style moderne.

`Lyrics`
: Correspond à une voix contenant des paroles. Le contexte `Lyrics` gère l'impression d'une ligne de paroles.

`VaticanaLyrics`
: Identique au contexte `Lyrics`, à ceci près qu'il fournit un style d'hyphénation (trait d'union simple aligné sur la syllabe de gauche) tels qu'utilisé en *Editio Vaticana*.

`GregorianTranscriptionLyrics`
: Contexte de paroles pour noter du grégorien dans un style moderne.

`DrumVoice`
: Contexte de voix dévolu à une portée de percussions.

`FiguredBass`
: Contexte prenant en charge les objets `BassFigure` – la basse chiffrée – créés à partir de ce qui a été saisi en mode `\figuremode`.

`TabVoice`
: Contexte de voix dévolu au contexte `TabStaff`, il est habituellement créé implicitement.

`CueVoice`
: Contexte de voix utilisé essentiellement dans le cadre de citations ajoutées à une portée – voir Section 6.3.3 [Mise en forme d'une citation], page 277, –, il est habituellement créé implicitement.

`ChordNames`
: Permet d'imprimer des noms d'accord.

`NoteNames`
: Permet d'imprimer le nom des notes.

`NullVoice`
: Permet d'aligner des paroles sans afficher les notes.

`Devnull`
: Passe sous silence toutes les informations musicales contenues dans ce contexte.

`Dynamics`
: Gère les nuances sur une même ligne, centrée entre les portées qui entourent ce contexte.

`StandaloneRhythmVoice`
: Contexte de niveau `Voice` dévolu à `\markup \rhythm`.

`Bottom`
: Il s'agit d'un contexte du plus bas niveau, accepté par tous les contextes intermédiaires. Il s'avère utile dans des situations où la même musique est utilisée, par exemple, à la fois dans des contextes `Staff` et `TabStaff`.

33.2 Création et référencement d'un contexte

LilyPond crée automatiquement des contextes de bas niveau lorsque l'expression musicale intervient avant qu'un contexte adéquat n'existe, ce qui peut être pratique dans le cadre d'une partition simple ou de courts fragments tels ceux inclus dans cette documentation. Dès que la
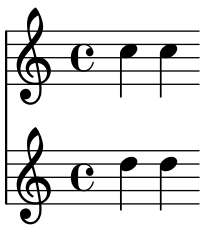
structure s'étoffe, il devient nécessaire de créer explicitement tous les contextes, à l'aide des commandes \new ou \context. Leur syntaxe est très similaire :

[\new | \context] *Contexte* [= *nom*] [*musique*]

où peuvent intervenir aussi bien \new que \context. Le *Contexte* est le nom du contexte à créer, qui éventuellement s'appellera plus particulièrement *nom* ; il contient l'expression musicale unique *musique* qui devra être interprétée dans ce contexte par les graveurs ou exécutants.
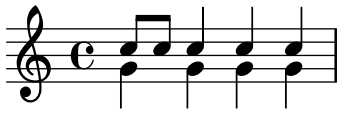
Le préfixe \new non suivi d'un nom s'utilise principalement pour créer une partition avec plusieurs portées :

```
<<
  \new Staff \relative {
    % leave the Voice context to be created implicitly
    c''4 c
  }
  \new Staff \relative {
    d''4 d
}
>>
```



et pour regrouper des voix sur une même portée :

```
\new Staff <<
  \new Voice \relative {
    \voiceOne c''8 c c4 c c
  }
  \new Voice \relative {
    \voiceTwo g'4 g g g
  }
>>
```
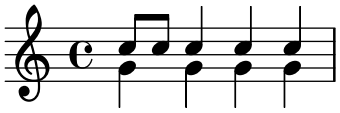


\new est à priviliégier lorsque les contextes ne sont pas nommés.

La différence entre les commandes \new et \context se situe au niveau de leurs effets :

- La commande \new, suivie ou non d'un nom, crée un tout nouveau contexte même s'il en existe déjà un portant le même nom :

```
\new Staff <<
  \new Voice = "A" \relative {
    \voiceOne c''8 c c4 c c
  }
  \new Voice = "A" \relative {
    \voiceTwo g'4 g g g
  }
>>
```
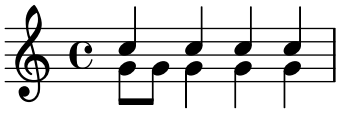


- La commande \context avec nommage crée un contexte distinct uniquement dans le cas où ne préexiste aucun contexte du même nom dans la même hiérarchie de contextes. Dans le cas contraire, il sert de référence au contexte précédemment créé, et son expression musicale est transmise dans ce contexte pour interprétation.

  Le nommage des contextes se révèle fort utile dans le cadre des paroles ou de la basse chiffrée comme indiqué dans Section 9.2.1 [Travail avec des paroles et variables], page 368, ou Section "Modèles pour ensemble vocal" dans *Manuel d'initiation* pour le premier cas et Section 15.3.3 [Gravure de la basse chiffrée], page 545, pour le second. Par ailleurs, cette procédure est tout à fait pertinente lorsque l'on sépare mise en forme de la partition et contenu musical. Les deux formulations ci-après sont tout à fait valides :

```
\score {
  <<
    % score layout
    \new Staff <<
      \new Voice = "one" {
        \voiceOne
      }
      \new Voice = "two" {
        \voiceTwo
      }
    >>
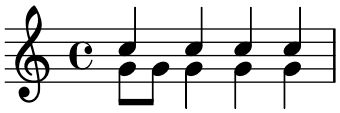
    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



```
\score {
  <<
    % score layout
    \new Staff <<
      \context Voice = "one" {
        \voiceOne
      }
      \context Voice = "two" {
```

```
        \voiceTwo
      }
    >>

    % musical content
    \context Voice = "one" {
      \relative {
        c''4 c c c
      }
    }
    \context Voice = "two" {
      \relative {
        g'8 g g4 g g
      }
    }
  >>
}
```



Par ailleurs, le recours à des variables produira les mêmes effets – voir Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

- La commande `\context` utilisée sans nommage recherche le premier de tous les contextes du même type précédemment créés dans la même hiérarchie de contextes ; l'expression musicale lui est alors transmise pour interprétation. Bien que rarement utilisée, cette formulation de `\context` sans nommage ni expression musicale permet de définir le contexte dans lequel une procédure Scheme comportant une clause `\applyContext` devra s'exécuter.

```
\new Staff \relative {
  c'1
  \context Timing
    \applyContext #(lambda (ctx)
                     (newline)
                     (display (ly:context-current-moment ctx)))
  c1
}
```

Un contexte auquel il est ultérieurement fait référence doit impérativement être nommé. C'est le cas par exemple lorsque des paroles sont associées à de la musique :

```
\new Voice = "tenor" musique
...
\new Lyrics \lyricsto "tenor" paroles
```

L'association de paroles à de la musique est abordée en détails à la rubrique Section 9.1.4 [Durée automatique des syllabes], page 359.

Les propriétés de tous les contextes d'un même type se modifient au sein d'un bloc `\layout`, selon une syntaxe différente – voir Section 33.5.1 [Modification de tous les contextes d'un même type], page 771. Une telle construction permet de séparer mise en forme et contenu musical. Lorsque un seul contexte requiert une adaptation, mieux vaut recourir à un bloc `\with` – voir Section 33.5.2 [Modification d'un contexte particulier], page 773.

Voir aussi

Manuel d'initiation : Section "Organisation du code source avec des variables" dans *Manuel d'initiation*.

Manuel de notation : Section 9.1.4 [Durée automatique des syllabes], page 359, Section 33.5.2 [Modification d'un contexte particulier], page 773.

33.3 Conservation d'un contexte

En règle générale, un contexte disparaît dès qu'il n'y a plus rien à faire. Autrement dit, un contexte `Voice` disparaît dès après le dernier événement qu'il contient, et un contexte `Staff` dès que les contextes `Voice` qu'il supporte ne contiennent plus rien. Ceci peut avoir des conséquences néfastes lorsqu'il est fait référence à un contexte alors disparu, comme dans le cas d'un changement de portée introduit par la commande `\change`, l'association de paroles à l'aide de la commande `\lyricsto` ou si des événements surviennent à nouveau pour ce contexte précédemment actif.

Une exception cependant à cette règle : à l'intérieur d'une construction `{...}` (séquence musicale), la notion de construction du « contexte en cours » descendra dès lors qu'un élément de la séquence se trouve dans un sous-contexte par rapport au contexte courant. Ceci évite la création intempestive de contextes implicites dans nombre de situations, mais signifie que le premier contexte dans lequel on descend sera gardé actif jusqu'à la fin de l'expression.

Par contre, en présence d'un contexte `Staff` ou dans une construction `<< ... >>`, un seul des contextes `Voice` inclus restera actif jusqu'à la fin du contexte `Staff` ou de la construction `<< ... >>`, y compris s'il y a des « trous ». Le contexte alors persistant sera le premier rencontré dans la construction `{ ... }` sans tenir compte des éventuels `<< ... >>` qu'elle pourrait contenir.

Un contexte restera actif dès lors qu'il s'y passera toujours quelque chose. Un contexte `Staff` restera actif si l'une des voix qu'il supporte est toujours active. L'un des moyens de s'en assurer consiste à ajouter des silences invisibles parallèlement à la musique. Vous devrez les ajouter dans tous les contextes `Voice` qui doivent rester actifs. Nous vous conseillons, lorsque plusieurs voix interviennent de manière sporadique, de toutes les maintenir actives plutôt que de vous fier aux exceptions mentionnées plus haut.
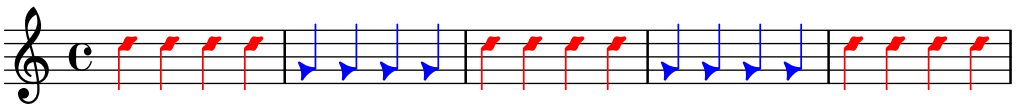
Dans l'exemple suivant, les deux voix A et B sont maintenues actives jusqu'à la fin du morceau :

```
musicA = \relative { d''4 d d d }
musicB = \relative { g'4 g g g }
keepVoicesAlive = {
  <<
    \new Voice = "A" { s1*5 }  % keep voice 'A' alive for 5 bars
    \new Voice = "B" { s1*5 }  % keep voice 'B' alive for 5 bars
  >>
}

music = {
  \context Voice = "A" {
    \voiceOneStyle \musicA
  }
  \context Voice = "B" {
    \voiceTwoStyle \musicB
  }
  \context Voice = "A" { \musicA }
```

```
  \context Voice = "B" { \musicB }
  \context Voice = "A" { \musicA }
}

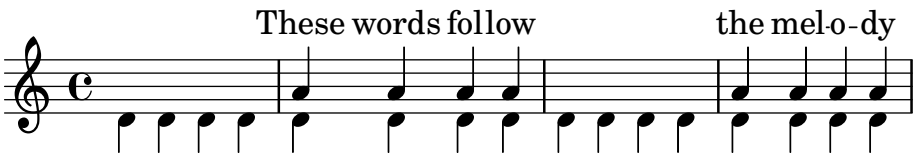
\score {
  \new Staff <<
    \keepVoicesAlive
    \music
  >>
}
```



L'exemple suivant illustre la manière d'écrire selon cette méthode une mélodie discontinue à laquelle se rattachent des paroles. Dans la réalité, mélodie et accompagnement feraient l'objet de portées séparées.

```
melody = \relative { a'4 a a a }
accompaniment = \relative { d'4 d d d }
words = \lyricmode { These words fol -- low the mel -- o -- dy }

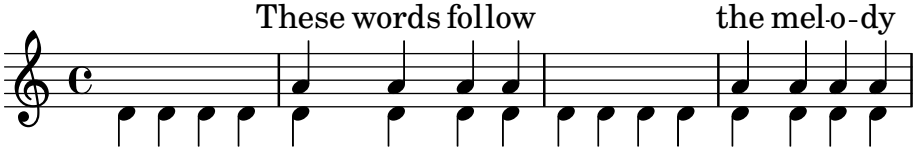
\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne s1*4  % Keep Voice "melody" alive for 4 bars
        }
        {
          \new Voice = "accompaniment" {
            \voiceTwo \accompaniment
          }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
          \context Voice = "accompaniment" { \accompaniment }
          <<
            \context Voice = "melody" { \melody }
            \context Voice = "accompaniment" { \accompaniment }
          >>
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "music" }
    \lyricsto "melody" { \words }
  >>
}
```



Une autre méthode, qui s'avère plus productive dans nombre de cas, consiste à maintenir active la ligne mélodique en y insérant des silences invisibles tout au long de l'accompagnement :

```
melody = \relative {
  s1  % skip a bar
  a'4 a a a
  s1  % skip a bar
  a4 a a a
}
accompaniment = \relative {
  d'4 d d d
  d4 d d d
  d4 d d d
  d4 d d d
}
words = \lyricmode { These words fol -- low the mel -- o -- dy }

\score {
  <<
    \new Staff = "music" {
      <<
        \new Voice = "melody" {
          \voiceOne \melody
        }
        \new Voice = "accompaniment" {
          \voiceTwo \accompaniment
        }
      >>
    }
    \new Lyrics \with { alignAboveContext = "music" }
    \lyricsto "melody" { \words }
  >>
}
```



33.4 Modification des greffons de contexte

Les contextes, tels que `Score` ou `Staff`, ne contiennent pas que des propriétés ; ils mettent également en œuvre certains sous-programmes (*plug-ins* pour employer le terme consacré) nommés « graveurs » (*engravers* pour reprendre le terme anglais). Ces sous-programmes sont chargés de créer les différents éléments de notation : on trouve ainsi dans le contexte `Voice` un graveur `Note_heads_engraver`, chargé des têtes de notes et, dans le contexte `Staff`, un graveur `Key_engraver`, chargé de l'armure.

Vous trouverez une description exhaustive de chaque graveur dans Section "Engravers and Performers" dans *Référence des propriétés internes*. Chaque contexte mentionné dans Section "Contexts" dans *Référence des propriétés internes* répertorie les graveurs mis en œuvre.
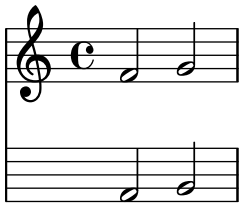
On peut faire, au moyen de ces graveurs, sa propre « cuisine », en modifiant les contextes à volonté.

Lorsqu'un contexte est créé, par la commande `\new` ou `\context`, on peut y adjoindre un bloc `\with` (en anglais « avec »), dans lequel il est possible d'ajouter (commande `\consists`) ou d'enlever (commande `\remove`) des graveurs :

```
\new contexte \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ...musique...
}
```

Ici chacun des points de suspension '...' devront être remplacés par le nom des graveurs désirés. Dans l'exemple suivant, on enlève du contexte `Staff` la métrique (graveur `Time_signature_engraver`) et la clef (graveur `Clef_engraver`).

```
<<
  \new Staff \relative {
    f'2 g
  }
  \new Staff \with {
     \remove Time_signature_engraver
     \remove Clef_engraver
  } \relative {
    f'2 g2
  }
>>
```



La clef et le chiffre de mesure ont disparu de la deuxième portée. C'est une méthode quelque peu radicale puisqu'elle affectera toute la portée jusqu'à la fin de la partition. L'espacement s'en trouve également affecté, ce qui peut être ou non l'effet recherché. Une méthode plus sophistiquée aurait été de rendre ces objets transparents (voir Section "Visibilité et couleur des objets" dans *Manuel d'initiation*).

Problèmes connus et avertissements

L'ordre dans lequel les graveurs sont spécifiés correspond à leur ordre d'apparition dans le processus d'élaboration de la partition. En règle générale, l'ordre dans lequel les graveurs sont mentionnés importe peu. Il se peut toutefois qu'un graveur écrive une propriété qui sera interprétée par un autre, ou qu'un graveur crée un objet graphique qui sera traité par un autre ; l'ordre d'apparition de ces graveurs prendra alors toute son importance.

Pour information, les ordonnancements suivants sont importants :

- le `Bar_engraver` devrait toujours être le premier ;
- le `New_fingering_engraver` doit toujours précéder le `Script_column_engraver` ;
- le `Timing_translator` doit toujours précéder le `Bar_number_engraver`.

Voir aussi

Fichiers d'initialisation : `ly/engraver-init.ly`.

33.5 Modification des réglages par défaut d'un contexte

Les propriétés des contextes et objets graphiques se modifient à l'aide des commandes `\set` et `\override` – voir Chapitre 35 [Modification de propriétés], page 786. Ces commandes créent des événements musicaux qui feront que la modification produira ses effets dès l'instant où la musique est traitée.

Le propos est ici de voir comment modifier les valeurs *par défaut* des propriétés de contexte ou d'objet graphique dès la création de ces contextes. Deux manières de procéder sont envisageables : l'une consiste à modifier les valeurs pour tous les contextes d'un même type, l'autre s'attache à adapter les valeurs par défaut d'une instance particulière d'un contexte.

33.5.1 Modification de tous les contextes d'un même type

L'adaptation des réglages par défaut d'un contexte, qu'il s'agisse de `Score`, `Staff` ou `Voice`, peut se réaliser indépendamment de la musique dans un bloc `\layout` – placé dans le bloc `\score` auquel ces modifications doivent s'appliquer – au moyen d'un bloc `\context`.
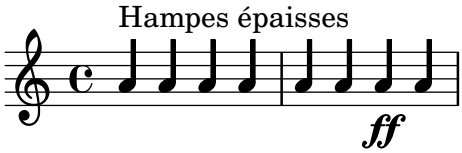
Les réglages dévolus à la sortie MIDI viendront quant à eux se placer dans un bloc `\midi` – voir Section 33.1.1 [Définitions de la sortie – hiérarchie des contextes], page 760.

```
\layout {
  \context {
    \Voice
    [réglage de contexte pour tous les contextes Voice]
  }
  \context {
    \Staff
    [réglage de contexte pour tous les contextes Staff]
  }
}
```

La spécification des adaptations peut se faire de différentes manières :

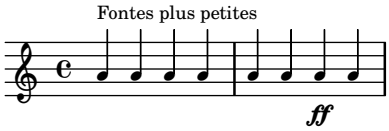
- à l'aide d'une commande `\override`, sans lui adjoindre le nom du contexte :

```
\score {
  \relative {
    a'4^"Hampes épaisses" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      \override Stem.thickness = 4.0
    }
  }
}
```



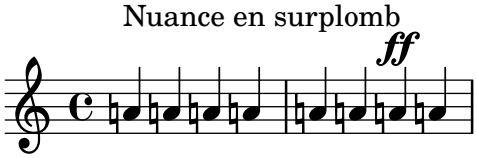
- en définissant directement une propriété de contexte :

```
\score {
  \relative {
    a'4^"Fontes plus petites" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Staff
      fontSize = -4
    }
  }
}
```



- à l'aide d'une commande prédéfinie comme `\dynamicUp`, ou bien une expression musicale telle que `\accidentalStyle dodecaphonic` :
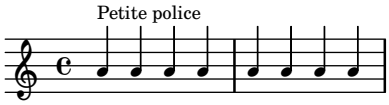
```
\score {
  \relative {
    a'4^"Nuance en surplomb" a a a
    a4 a a\ff a
  }
  \layout {
    \context {
      \Voice
      \dynamicUp
    }
    \context {
      \Staff
      \accidentalStyle dodecaphonic
    }
  }
}
```



- à l'aide d'une variable personnalisée contenant un bloc `\with` ; pour de plus amples informations sur le bloc `\with`, voir Section 33.5.2 [Modification d'un contexte particulier], page 773.
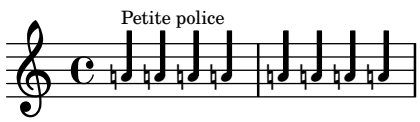
```
StaffDefauts = \with {
  fontSize = -4
}

\score {
  \new Staff {
    \relative {
      a'4^"Petite police" a a a
      a4 a a a
    }
  }
  \layout {
    \context {
      \Staff
      \StaffDefauts
    }
  }
}
```



Les instructions destinées à modifier les propriétés peuvent se placer dans un bloc `\layout` sans pour autant être incluses dans un bloc `\context`. Expliciter des réglages de la sorte équivaut à inclure les commandes de modification des propriétés au début de chacun des contextes du type en question. Lorsque le contexte n'est pas spécifié, *tous* les contextes de bas niveau seront affectés – voir Section 33.1.5 [Contextes de bas niveau – les voix], page 762. La syntaxe appropriée répond aux mêmes critères que si la commande était écrite dans le flot musical.

```
\score {
  \new Staff {
    \relative {
      a'4^"Petite police" a a a
      a4 a a a
    }
  }
  \layout {
    \accidentalStyle dodecaphonic
    \set fontSize = -4
    \override Voice.Stem.thickness = 4.0
  }
}
```



33.5.2 Modification d'un contexte particulier

Dans le cas d'un contexte pris individuellement, ses propriétés se modifient à l'aide d'un bloc `\with`. Toutes les autres instances de contexte appartenant au même type seront affectées des réglages prédéfinis par LilyPond, modifiés le cas échéant au sein d'un bloc `\layout`. Le bloc `\with` se place directement à la suite de la commande `\new` *type-de-contexte*.

```
\new Staff \with {
```

```
  [réglages pour ce contexte pris individuellement]
} {
  ...
}
```

De la même manière, si la musique est saisie à la suite d'une commande abrégée, telle que `\chords` au lieu de `\chordmode`, l'instruction `\with` viendra se placer juste après la commande spécifiant le mode :

```
\chords \with {
  [réglages pour ce contexte (implicite) pris individuellement]
} {
  ...
}
```
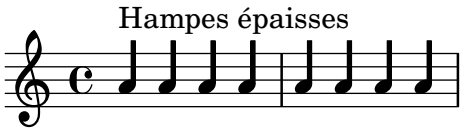
puisque c'est le contexte implicite alors créé qui devra être modifié. Cette manière de procéder s'applique à toutes les autres formes abrégées de spécification du mode de saisie (`\drums`, `\figures`) – voir Chapitre 19 [Modes de saisie], page 605.

Dans la mesure où une telle « modification de contexte » est spécifiée au sein même de la musique, ses effets toucheront **toutes** les sorties (imprimable **et** MIDI), contrairement à ce qui se passe lorsque les adaptations sont réalisées dans la définition d'une sortie.

La spécification des adaptations peut se faire de différentes manières :

- à l'aide d'une commande `\override`, sans lui adjoindre le nom du contexte :

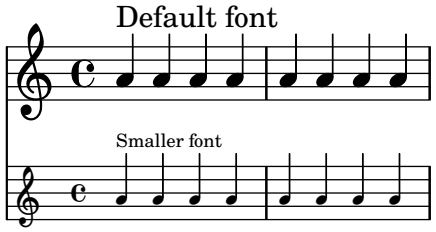
```
\score {
  \new Staff {
    \new Voice \with { \override Stem.thickness = 4.0 } {
      \relative {
        a'4^"Hampes épaisses" a a a
        a4 a a a
      }
    }
  }
}
```



- en définissant directement une propriété de contexte :
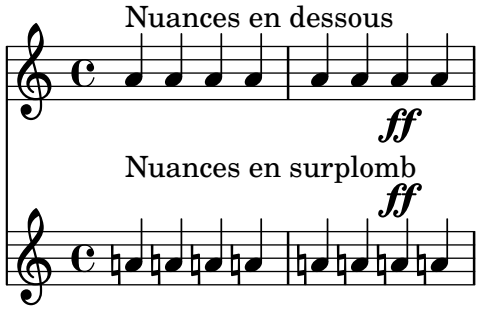
```
\score {
  <<
    \new Staff {
      \relative {
        a'4^"Default font" a a a
        a4 a a a
      }
    }
    \new Staff \with { fontSize = -4 } {
      \relative {
        a'4^"Smaller font" a a a
        a4 a a a
      }
    }
```

```
  >>
}
```



- à l'aide d'une commande prédéfinie comme `\dynamicUp` :

```
\score {
  <<
    \new Staff {
      \new Voice {
        \relative {
          a'4^"Nuances en dessous" a a a
          a4 a a\ff a
        }
      }
    }
    \new Staff \with { \accidentalStyle dodecaphonic } {
      \new Voice \with { \dynamicUp } {
        \relative {
          a'4^"Nuances en surplomb" a a a
          a4 a a\ff a
        }
      }
    }
  >>
}
```



Voir aussi

Manuel de notation : Chapitre 19 [Modes de saisie], page 605.

33.5.3 Ordre de préséance

La valeur d'une propriété qui doit s'appliquer à un instant particulier est déterminée comme suit :

- s'il y a une instruction `\override` ou `\set` active dans le flot d'information, sa valeur s'applique,
- en l'absence de quoi sera utilisée la valeur par défaut telle que définie dans une clause `\with` stipulée à l'initialisation du contexte,

- en l'absence de quoi sera retenue la valeur par défaut issue du bloc `\context` approprié le plus récent dans les blocs `\layout` ou `\midi`,
- en l'absence de quoi s'appliqueront les réglages prédéfinis de LilyPond.

Voir aussi

Manuel d'initiation : Section "Modification des propriétés d'un contexte" dans *Manuel d'initiation*.

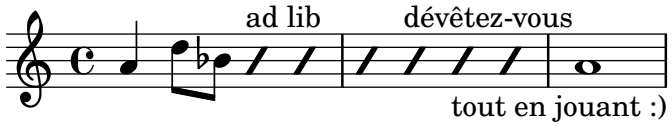
Manuel de notation : Section 33.1.5 [Contextes de bas niveau – les voix], page 762, Section 35.3 [Les commandes `\override` et `\revert`], page 788, Section 35.2 [Les commandes `\set` et `\unset`], page 786, Section 27.1 [Le bloc `\layout`], page 705, Section 33.1 [Tout savoir sur les contextes], page 760.

33.6 Définition de nouveaux contextes

Généralités en matière de définition d'un contexte

Les contextes tels que `Staff` ou `Voice` sont faits de briques de construction empilées. En combinant divers graveurs, il est possible de créer de nouveaux types de contextes.

Dans l'exemple suivant on construit, de zéro, un nouveau contexte très semblable à `Voice`, mais qui n'imprime que des têtes de notes en forme de barre oblique au centre de la portée. Un tel contexte peut servir, par exemple, à indiquer un passage improvisé dans un morceau de jazz.



On a rassemblé les réglages dans un bloc `\context`, lui-même placé dans le bloc `\layout` :

```
\layout {
  \context {
    ...
  }
}
```

En lieu et place des points (. . .), voici les éléments à saisir :

Tout d'abord, il convient de donner un nom à notre nouveau contexte :

```
\name ImproVoice
```

Comme il est très semblable à un contexte `Voice`, nous souhaitons que toutes les commandes associées au `Voice` déjà existant restent valables. D'où nécessité de la commande `\alias`, qui va l'associer au contexte `Voice` :

```
\alias Voice
```

Ce contexte doit pouvoir imprimer des notes et des indications textuelles ; on ajoute donc les graveurs appropriés ainsi que celui dévolu au regroupement sous forme de colonne des notes, hampes et silences qui interviennent au même moment musical :

```
\consists Note_heads_engraver
\consists Text_engraver
\consists Rhythmic_column_engraver
```

Toutes les notes devraient s'afficher au centre de la portée :

```
\consists Pitch_squash_engraver
squashedPosition = 0
```

Le graveur Pitch_squash_engraver intercepte les notes créées par le Note_heads_engraver, et les « écrase » pour qu'elles aient toutes la même position verticale, définie par squashedPosition : ici il s'agit de la valeur 0, c'est-à-dire la ligne du milieu.

On veut que les notes aient la forme d'une barre oblique, sans aucune hampe :

```
\override NoteHead.style = #'slash
\hide Stem
```

Tous ces modules doivent communiquer sous le contrôle du contexte. Les mécanismes permettant aux contextes de communiquer sont établis dès lors que le mot-clé \type précède le contexte. La plupart des contextes mentionnés au sein d'un bloc \layout sont de type Engraver_group. Certains contextes spécifiques reposent sur d'autres types de contexte. Recopier un contexte préexistant pour en modifier la définition lui affecte le type adéquat. Dans la mesure où notre exemple consiste à créer une définition de toute pièce, son type doit être explicitement spécifié.

```
\type Engraver_group
```

Récapitulons ; on se retrouve avec le bloc suivant :

```
\context {
  \name ImproVoice
  \type Engraver_group
  \consists Note_heads_engraver
  \consists Text_engraver
  \consists Rhythmic_column_engraver
  \consists Pitch_squash_engraver
  squashedPosition = 0
  \override NoteHead.style = #'slash
  \hide Stem
  \alias Voice
}
```

Ce n'est pas tout. En effet, on veut intégrer le nouveau contexte ImproVoice dans la hiérarchie des contextes. Tout comme le contexte Voice, sa place est au sein du contexte Staff. Nous allons donc modifier la définition du contexte Staff, au moyen de la commande \accepts :

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Souvent, lorsque l'on réutilise le définition d'un contexte, le contexte qui en résultera devrait pouvoir s'utiliser partout où le contexte originel serait utile. Écrire

```
\layout {
  ...
  \inherit-acceptability résultant originel
}
```

fait en sorte que les contextes de type *résultant* soient acceptés par le contextes qui acceptent *originel*. Par exemple, libeller

```
\layout {
  ...
  \inherit-acceptability ImproVoice Voice
}
```

a pour effet d'ajouter un \accepts pour ImproVoice à la fois aux définitions de Staff et RhythmicStaff.

Le contraire de \accepts est \denies ; il est parfois utile lorsque l'on recycle des définitions de contextes déjà existantes.

Enfin, tout cela doit prendre place dans le bloc `\layout`, comme ceci :

```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
    \accepts ImproVoice
  }
}
```

On peut alors saisir la musique, comme dans l'exemple plus haut :

```
\relative {
  a'4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"dévêtez-vous"
    c c_"tout en jouant :)"
  }
  a1
}
```

Voir aussi

Référence des propriétés internes : Section "Contexts" dans *Référence des propriétés internes*, Section "Engravers and Performers" dans *Référence des propriétés internes*.

Création de contextes en MIDI

En matière de sortie MIDI, la syntaxe permettant de définir de nouveaux types de contextes est identique, à ceci près que la section `\context` viendra se placer au sein du bloc `\midi` et le `\type` sera `Performer_group` au lieu de `Engraver_group`. Le terme *engraver* (graveur) fait référence à un module de contexte chargé de générer une sotie visuelle, alors qu'un *performer* (exécutant) est dévolu exclusivement à une sortie MIDI. Lorsque les noms de modules comportent *translator* (traducteur) au lieu de *engraver* ou *performer*, ils sont alors fonctionnels tant pour des sorties graphiques qu'audio. Afin d'adapter une définition de contexte pour le bloc `\midi`, il faudra

- la copier dans un bloc `\midi`,
- changer `Engraver_group` en `Performer_group`,
- supprimer les `\consists` propres aux graveurs (ils ne sont pas pertinents) et éventuellement ajouter des `\consists` pour les exécutants.

Il faut garder à l'esprit, afin de préserver une cohérence d'interprétation entre les sorties imprimables et MIDI, qu'il est de bon ton de reporter toute définition de contexte personnalisée dans le bloc `\midi`. Cette duplication devrait au moins comporter les commandes spécifiant la hiérarchie du contexte, telles que `\accepts`, `\defaultchild` et `\inherit-acceptability`. Recopier aussi les alias est recommandé.

Ainsi, dans le but de compléter l'exemple vu plus haut, on peut y ajouter :

```
\midi {
  \context {
    \name ImproVoice
    \type Performer_group
    \alias Voice
```

```
      \consists Note_performer
      \consists Beam_performer
      \consists Dynamic_performer
      \consists Tie_performer
      \consists Slur_performer
    }
    \context {
      \Staff
      \accepts ImproVoice
    }
  }
```

Le contexte `ImproVoice` sera alors fonctionnel pour la sortie MIDI.

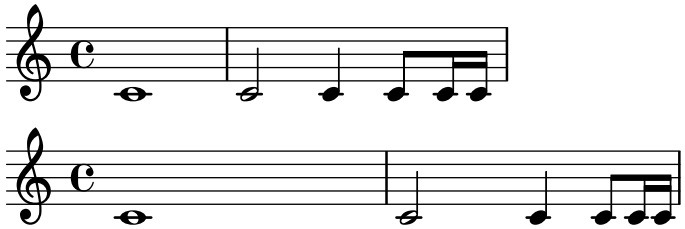
Remplacement du contexte `Score`

Dans le but d'écrire un contexte `MonScore` qui agirait comme étant le contexte le plus haut, à l'instar de ce que fait le contexte `Score`, il faudra expliciter une instruction `\inherit-acceptability MonScore Score`. L'exemple suivant définit un contexte `ProportionalScore` pour lequel la notation proportionnelle est activée (cf. Section 30.6 [Notation proportionnelle], page 746).

```
\layout {
  \context {
    \Score
    \name ProportionalScore
    \alias Score
    proportionalNotationDuration = #1/8
  }
  \inherit-acceptability ProportionalScore Score
}

music = { c'1 2 4 8 16 16 }

\new Score \music
\new ProportionalScore \music
```



Dans la mesure où le contexte sommital doit contenir certains graveurs fondamentaux, hériter des réglages de `\Score` est facile dans la plupart des cas. Néanmoins, la définition d'un contexte de niveau `Score` à partir de zéro requiert que l'argument à `\type` soit égal à `Score_engraver` (ou `Score_performer` dans le bloc `\midi`) en lieu et place de `Engraver_group`. Par ailleurs, affecter au contexte sommital l'alias `Score` est fortement recommandé dans la mesure où un certain nombre de graveurs ont besoin d'accéder au contexte sommital par son alias.

33.7 Ordonnancement des contextes

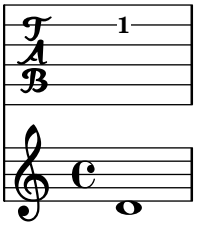
Les contextes viennent en principe se positionner selon leur ordre d'apparition dans le fichier source. Lorsque plusieurs contextes sont imbriqués, le contexte englobant supportera les

différents contextes mentionnés dans le fichier source, à la stricte condition qu'ils soient dûment « agréés ». Les contextes imbriqués qui ne font pas partie de la « liste d'agréments » du contexte englobant se retrouveront en dessous de celui-ci au lieu d'y être imbriqués.

La liste des « agréments » d'un contexte se gère à l'aide des instructions `\accepts` ou `\denies` – `\accepts` pour ajouter un contexte à la liste, `\denies` pour retirer l'agrément.

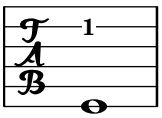
Par exemple, un `TabStaff` accepte par défaut des contextes `TabVoice` mais ne supporte pas de contextes `Voice`. Un contexte `Voice` apparaissant dans un `TabStaff` se verrait repositionné sur une portée séparée.

```
\score {
  \new TabStaff <<
    \new TabVoice { c'1 }
    \new Voice { d'1 }
  >>
}
```



Néanmoins, et grâce à une instruction `\accepts`, un contexte `Voice` peut se voir ajouté au `TabStaff` :

```
\score {
  \new TabStaff <<
    \new TabVoice { c'1 }
    \new Voice { d'1 }
  >>
\layout {
  \context {
    \TabStaff
    \accepts Voice
    }
  }
}
```



L'instruction `\denies` permet, lorsqu'un nouveau contexte reprend les définitions d'un contexte existant, d'en ajuster les composantes. C'est par exemple le cas du contexte `VaticanaStaff`, réplique du contexte `Staff`, au sein duquel le contexte `VaticanaVoice` se substitue au contexte `Voice` dans la « liste d'agrément ».

Gardez à l'esprit que, face à une instruction qui ne s'appliquerait à aucun contexte déjà existant, LilyPond crée un nouveau contexte implicite.

Lors de la définition d'un contexte, les types de contextes sous-jacents susceptibles d'être créés implicitement sont spécifiés à l'aide d'une commande `\defaultchild`. Un certain nombre d'événements musicaux requièrent un contexte de plus bas niveau ; face à un tel événement, LilyPond crée autant de « sous-contextes » que nécessaire, jusqu'au contexte ne comportant aucun `\defaultchild`.

La création implicite de contexte peut donc finir par engendrer de manière intempestive une nouvelle portée ou une autre partition. L'utilisation d'une instruction `\new` pour créer explicitement des contextes permet d'éviter ces problèmes.

Il arrive qu'un contexte ne doive exister que pendant un court instant, ce qui est le cas par exemple pour une *ossia*. Le plus simple consiste alors à initialiser la définition d'un contexte à l'endroit approprié, en parallèle avec le fragment correspondant dans la musique principale. Ce contexte temporaire sera par défaut positionné sous les autres contextes existants. Le repositionner au-dessus du contexte « principal » demande de le définir ainsi :

```
\new Staff \with { alignAboveContext = "principal" }
```

Il en va de même pour les contextes temporaires de paroles au sein d'un système à plusieurs portées comme un `ChoirStaff` lorsque, par exemple, un couplet supplémentaire apparaît à l'occasion d'une reprise. Ce contexte de paroles temporaire se place par défaut sous les portées inférieures. Lui adjoindre une instruction `alignBelowContext` dès son initialisation permet de l'accoler au contexte de paroles (nommé) qui contient le premier couplet.

Des exemples de repositionnement de contexte temporaire sont disponibles aux rubriques Section "Expressions musicales imbriquées" dans *Manuel d'initiation*, Section 6.2 [Modification de portées individuelles], page 258, et Section 9.2 [Situations particulières en matière de paroles], page 368.

Voir aussi

Manuel d'initiation : Section "Expressions musicales imbriquées" dans *Manuel d'initiation*.

Manuel de notation : Section 6.2 [Modification de portées individuelles], page 258, Section 9.2 [Situations particulières en matière de paroles], page 368.

Manuel d'utilisation : Voir Section "Apparition d'une portée supplémentaire" dans *Utilisation des programmes*.

Fichiers d'initialisation : `ly/engraver-init.ly`.

34 En quoi consiste la référence des propriétés internes
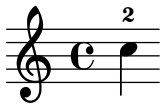
Voir aussi

Manuel de notation : Section B.24 [Conventions de nommage], page 989.

34.1 Navigation dans les références du programme

Comment, par exemple, déplacer le doigté dans le fragment suivant ?

```
c''-2
```



Sur la page de la documentation relative aux doigtés, c'est-à-dire Section 7.1.2 [Doigtés], page 290, se trouve l'indication suivante :

> **Voir aussi**
>
> Référence des propriétés internes : Section "Fingering" dans *Référence des propriétés internes*.

Ladite référence est disponible au format HTML, ce qui rend la navigation bien plus aisée. Il est possible soit de la lire en ligne, soit de la télécharger dans ce format. La démarche présentée ici sera plus difficile à comprendre dans un document au format PDF.

Suivons le lien Section "Fingering" dans *Référence des propriétés internes*. En haut de la nouvelle page, on peut lire

> Fingering objects are created by the following engraver(s): Section "Fingering_engraver" dans *Référence des propriétés internes* and Section "New_fingering_engraver" dans *Référence des propriétés internes*.

En d'autres termes, *Les indications de doigtés* (`Fingering` en anglais) *sont créées par les graveurs Section "Fingering_engraver" dans Référence des propriétés internes et Section "New_fingering_engraver" dans Référence des propriétés internes.*

En suivant derechef les liens propres à la référence du programme, on suit en fait le cheminement qui aboutit à la création de la partition :

- Section "Fingering" dans *Référence des propriétés internes*: Section "Fingering" dans *Référence des propriétés internes* objects are created by the following engraver(s): Section "Fingering_engraver" dans *Référence des propriétés internes*.
- Section "Fingering_engraver" dans *Référence des propriétés internes*: Music types accepted: Section "fingering-event" dans *Référence des propriétés internes*
- Section "fingering-event" dans *Référence des propriétés internes*: Music event type `fingering-event` is in Music expressions named Section "FingeringEvent" dans *Référence des propriétés internes*

Ce cheminement se produit, bien sûr, en sens inverse : nous sommes ici partis du résultat, et avons abouti aux événements (en anglais *Events*) engendrés par le fichier d'entrée. L'inverse est également possible : on peut partir d'un événement et suivre le cheminement de LilyPond qui aboutit à la création d'un ou plusieurs objets graphiques.

La référence des propriétés internes peut également se parcourir comme un document normal. On y trouve des chapitres tels que `Music definitions` Section "Translation" dans *Référence des propriétés internes*, ou encore Section "Backend" dans *Référence des propriétés internes*. Chaque chapitre recense toutes les définitions employées, et les propriétés sujettes à ajustements.

La Référence des propriétés internes n'est pas traduite en français – notamment du fait qu'elle est en évolution constante, tout comme LilyPond. En revanche, les termes musicaux font l'objet d'un Section "glossaire" dans *Glossaire* fort utile pour les utilisateurs non anglophones.

34.2 Interfaces de rendu

Tous les éléments de notation sont considérés comme des objets graphiques (en anglais *Graphical Object*, d'où le diminutif *Grob*). Chaque objet est doté d'un certain nombre de propriétés (l'épaisseur du trait, l'orientation, etc.), et lié à d'autres objets. Le fonctionnement de ces objets est décrit en détail dans Section "grob-interface" dans *Référence des propriétés internes*.

Prenons l'exemple des doigtés (en anglais *Fingering*). La page `Fingering` de la Référence des propriétés internes établit une liste de définitions propres à ce type d'objet :

> `padding` (dimension, in staff space):
>
> `0.5`

Ce qui signifie que les doigtés doivent être maintenus à une certaine distance (*padding*) des notes : 0,5 unités *staff-space* (espace de portée).

Chaque objet peut avoir plusieurs attributs, en tant qu'élément typographique ou musical. Ainsi, un doigté (objet *Fingering*) possède les attributs suivants :

- Sa taille ne dépend pas de l'espacement horizontal, contrairement aux liaisons ou ligatures.
- C'est du texte – un texte vraiment court, certes.
- Ce texte est imprimé au moyen d'une fonte, contrairement aux liaisons ou ligatures.
- Sur l'axe horizontal, le centre de ce symbole doit être aligné avec le centre de la note.
- Sur l'axe vertical, le symbole doit être proche de la note et de la portée.
- Sur l'axe vertical encore, il doit également s'ordonner avec les éventuels autres symboles, ponctuations ou éléments textuels.

Faire appliquer ces différents attributs est le rôle des *interfaces*, que l'on trouve en bas de la page Section "Fingering" dans *Référence des propriétés internes*.

> This object supports the following interfaces: Section "item-interface" dans *Référence des propriétés internes*, Section "self-alignment-interface" dans *Référence des propriétés internes*, Section "side-position-interface" dans *Référence des propriétés internes*, Section "text-interface" dans *Référence des propriétés internes*, Section "text-script-interface" dans *Référence des propriétés internes*, Section "font-interface" dans *Référence des propriétés internes*, Section "finger-interface" dans *Référence des propriétés internes*, and Section "grob-interface" dans *Référence des propriétés internes*.

En français,

> Cet objet admet les interfaces suivantes :

Suit la liste des interfaces en question, présentées comme autant de liens qui conduisent aux pages dédiées à chacune d'entre elles. Chaque interface est dotée d'un certain nombre de propriétés, dont certaines peuvent être modifiées, d'autres non (les *Internal properties*, ou propriétés internes).

Pour aller encore plus loin, plutôt que de simplement parler de l'objet `Fingering`, ce qui ne nous avance pas à grand chose, on peut aller explorer son âme même, dans les fichiers source de LilyPond (voir Section "Autres sources de documentation" dans *Manuel d'initiation*), en l'occurrence le fichier `scm/define-grobs.scm` :

```
(Fingering
  . ((padding . 0.5)
     (avoid-slur . around)
```
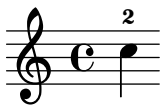
```
(slur-padding . 0.2)
(staff-padding . 0.5)
(self-alignment-X . 0)
(self-alignment-Y . 0)
(script-priority . 100)
(stencil . ,ly:text-interface::print)
(direction . ,ly:script-interface::calc-direction)
(font-encoding . fetaText)
(font-size . -5)  ; don't overlap when next to heads.
(meta . ((class . Item)
(interfaces . (finger-interface
               font-interface
               text-script-interface
               text-interface
               side-position-interface
               self-alignment-interface
               item-interface))))))
```

. . .où l'on découvre que l'objet `Fingering` n'est rien de plus qu'un amas de variables et de réglages. La page de la Référence des propriétés internes est en fait directement engendrée par cette définition.

34.3 Détermination de la propriété d'un objet graphique (grob)

Nous voulions changer la position du chiffre **2** dans le fragment suivant :

```
c''-2
```



Dans la mesure où le **2** est placé, verticalement, à proximité de la note qui lui correspond, nous allons devoir trouver l'interface en charge de ce placement, qui se trouve être `side-position-interface`. Sur la page de cette interface, on peut lire :

> `side-position-interface`
>
> Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Ce qui signifie

> `side-position-interface`
>
> Placer l'objet affecté à proximité d'autres objets. La propriété `direction` indique où positionner l'objet (à droite ou à gauche, en haut ou en bas).

En dessous de cette description se trouve décrite la variable `padding` :

> `padding` (dimension, in staff space)
>
> Add this much extra space between objects that are next to each other.

Ce qui signifie

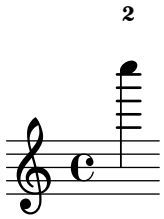
> Ajouter tel espace supplémentaire entre des objets proches les uns des autres.

En augmentant la valeur de `padding`, on peut donc éloigner le doigté de la note. La commande suivante insère trois unités d'espace vide entre la note et le doigté :

```
\once \override Voice.Fingering.padding = 3
```

En ajoutant ce tampon avant la création du doigté (de l'objet `Fingering`), donc avant `c2`, on obtient le résultat suivant :

```
\once \override Voice.Fingering.padding = 3
c''-2
```



Dans le cas présent, le réglage intervient dans le contexte `Voice`, ce qui pouvait également se déduire de la Référence des propriétés internes, où la page du graveur Section "Fingering_engraver" dans *Référence des propriétés internes* indique :

> Fingering_engraver is part of contexts: ... Section "Voice" dans *Référence des propriétés internes*

Ce qui signifie

> Le graveur Fingering_engraver fait partie des contextes : ... Section "Voice" dans *Référence des propriétés internes*

35 Modification de propriétés

35.1 Vue d'ensemble de la modification des propriétés

Les contextes comportent deux types différents de propriétés : des propriétés de contexte et des propriétés d'objet graphique. Les propriétés de contexte sont celles qui s'appliqueront globalement au contexte en tant que tel ; elles gèrent la manière dont le contexte apparaîtra. Les propriétés d'objet graphique, par contre, s'appliquent à des types particuliers d'objet qui apparaissent dans le contexte en question.

Les propriétés des contextes se libellent sous la forme `minusculeMajuscule`. Leur rôle consiste principalement à traduire la musique en notation : par exemple, `localAlterations` déterminera quand imprimer une altération accidentelle, et `currentBarNumber` quel numéro de mesure imprimer. La valeur des propriétés des contextes peut évoluer au fur et à mesure que l'on avance dans le morceau – `currentBarNumber` en est l'illustration parfaite.

La commande `\set` (et sa contrepartie `\unset`) permet de modifier les valeurs des propriétés de contexte. La commande `\override` (et sa contrepartie `\revert`) permet de modifier les valeurs des propriétés des objets graphiques.

Voir aussi

Référence des propriétés internes : Section "All layout objects" dans *Référence des propriétés internes*, Section "Backend" dans *Référence des propriétés internes*, Section "OverrideProperty" dans *Référence des propriétés internes*, Section "PropertySet" dans *Référence des propriétés internes*, Section "RevertProperty" dans *Référence des propriétés internes*.

35.2 Les commandes `\set` et `\unset`

Chaque contexte peut avoir plusieurs **propriétés**, c'est-à-dire des variables qu'il inclut. Ces dernières peuvent être modifiées « à la volée », c'est-à-dire pendant que la compilation s'accomplit. C'est ici le rôle de la commande `\set` qui répond à la syntaxe :

```
\set contexte.propriété = valeur
```

où *valeur* doit être précédée d'un '`#`' si elle est un objet Scheme.

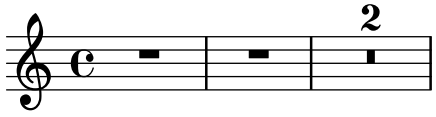
La commande `\unset` permet d'annuler la définition d'une propriété :

```
\unset contexte.propriété
```

si et seulement si cette *propriété* a été définie dans ce *contexte* précis. En d'autres termes, la commande `\unset` doit impérativement affecter le même contexte que la commande `\set` d'origine, même en cas d'imbrication.
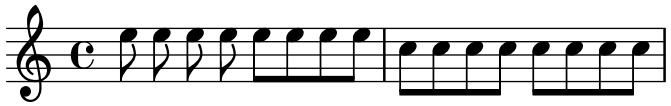
Ainsi la propriété de contexte `skipBars` permet de condenser les mesures vides de notes, en des silences multimesures – voir Section 6.3.4 [Compression de mesures vides], page 282, – dès lors qu'elle est valorisée à `#t` :

```
R1*2
\set Score.skipBars = ##t
R1*2
```
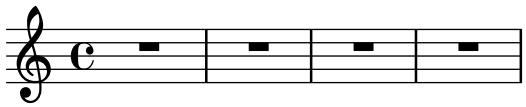


Si l'argument *contexte* n'est pas spécifié, alors la propriété cherchera à s'appliquer dans le contexte le plus restreint où elle est employée : le plus souvent `ChordNames`, `Voice`, `TabVoice` ou `Lyrics`.

```
\set Score.autoBeaming = ##f
\relative {
  e''8 e e e
  \set autoBeaming = ##t
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```



Notez que le contexte le plus restreint peut ne pas contenir le graveur contenant la *propriété* qui vous intéresse : ainsi, la propriété `skipBars`, évoquée plus haut, ne relève pas du contexte `Voice`, mais du contexte `Score` – le code suivant ne fonctionnera pas.
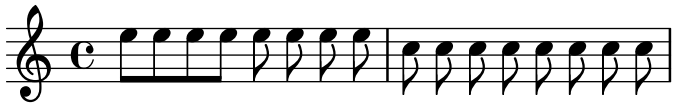
```
R1*2
\set skipBars = ##t
R1*2
```



Les contextes s'organisent de façon hiérarchique : aussi, lorsqu'un contexte de niveau supérieur est spécifié (par exemple `Staff`), la propriété sera modifiée dans tous les contextes inférieurs (tous les contextes `Voice`, par exemple) qu'il contient.

Tous les contextes héritent des réglages établis au plus haut niveau dans le contexte `Global` via `\grobdescriptions`. Certains de ces réglages par défaut font toutefois l'objet de dérogation dans la définition de contextes particuliers.

```
\set Score.autoBeaming = ##t
\relative {
  \unset autoBeaming
  e''8 e e e
  \unset Score.autoBeaming
  e8 e e e
} \\
\relative {
  c''8 c c c c8 c c c
}
```



Si l'on se trouve dans le contexte le plus restreint, il n'est pas obligatoire, là encore, de spécifier le *contexte*. Ainsi, les deux lignes suivantes
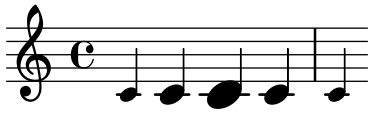
```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

sont équivalentes si elles apparaissent dans un contexte `Voice`.

Comme indiqué ci-dessus, `\unset` restaure la valeur par défaut d'une propriété de contexte. Néanmoins, il est parfois utile de temporairement modifier une valeur puis de revenir à

la précédente. Les deux commandes \pushContextProperty et \popContextProperty sont prévues à cet effet : la première insère la valeur courante dans une pile, alors que la seconde supprime cette valeur de la pile et restaure la propriété à sa valeur initiale.

```
{
  c'
  \pushContextProperty Staff.fontSize
  \set Staff.fontSize = 3
  c'
  \pushContextProperty Staff.fontSize
  \set Staff.fontSize = 6
  c'
  \popContextProperty Staff.fontSize
  c'
  \popContextProperty Staff.fontSize
  c'
}
```



35.3 Les commandes \override et \revert
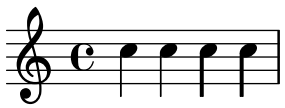
La commande \override permet de modifier la mise en forme des objets graphiques. Les descriptions d'objet graphique, dont les noms commencent par une majuscule, puis comprennent une ou plusieurs majuscules (de style TotoTata), contiennent les réglages « par défaut » pour les objets graphiques. Ces réglages sont sous forme de listes associatives Scheme ; on peut les consulter dans le fichier scm/define-grobs.scm.

\override répond à la syntaxe :

\override [*contexte.*]*NomObjet.propriété* = *valeur*

Nous pouvons donc par exemple accroître l'épaisseur des hampes en jouant sur la propriété thickness de l'objet stem :
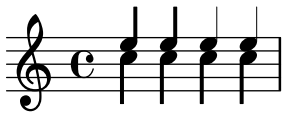
```
c''4 c''
\override Voice.Stem.thickness = 3.0
c''4 c''
```



Lorsqu'aucun contexte n'est spécifié dans une clause \override, celle-ci s'appliquera au contexte le plus bas :

```
\override Staff.Stem.thickness = 3.0
<<
  \relative {
    e''4 e
    \override Stem.thickness = 0.5
    e4 e
  } \\
  \relative {
    c''4 c c c
  }
```

```
>>
```



Certaines « sous-propriétés » sont parfois contenues dans une propriété. La commande devient alors :

```
\override Stem.details.beamed-lengths = #'(4 4 3)
```

ou, pour modifier les extrémités d'un objet à extension :

```
\override TextSpanner.bound-details.left.text = "texte de gauche"
\override TextSpanner.bound-details.right.text = "texte de droite"
```
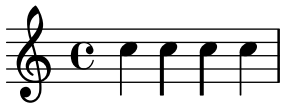
Les effets d'un `\override` prennent fin à l'aide de l'instruction `\revert`.

La syntaxe de la commande `\revert` est :

```
\revert [contexte.]NomObjet.propriété
```
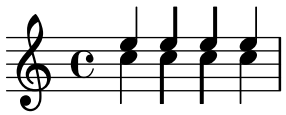
Par exemple :

```
\relative {
  c''4
  \override Voice.Stem.thickness = 3.0
  c4 c
  \revert Voice.Stem.thickness
  c4
}
```



Les effets d'un `\override` ou d'un `\revert` s'appliquent dès l'endroit où ils apparaissent, et à tous les objets dans le contexte mentionné :
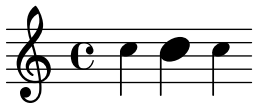
```
<<
  \relative {
    e''4
    \override Staff.Stem.thickness = 3.0
    e4 e e
  } \\
  \relative {
    c''4 c c
    \revert Staff.Stem.thickness
    c4
  }
>>
```



35.4 La commande \once

Les instructions \override et \set doivent être précédées d'un \once dès lors que les effets de l'accommodement ne concernent que l'événement qui les suit directement :

```
c''4
\once \set fontSize = 4.7
c''4
c''4
```



Voir aussi

Référence des propriétés internes : Section "Backend" dans *Référence des propriétés internes*.

35.5 \set ou \override

Les instructions \set et \override manipulent toutes deux des propriétés associées à des contextes. Les propriétés suivent une *hiérarchie des contextes* selon laquelle chaque contexte en comporte un ou plusieurs en dessous de lui-même. Une propriété qui n'a pas été définie dans un contexte particulier adopte les valeurs définies par son contexte supérieur hiérarchique direct.

Les valeurs et durée de vie des propriétés d'un contexte sont dynamiques et ne sont accessibles qu'au moment où la musique est interprétée. Lors de la création d'un contexte, ses propriétés sont initialisées à partir de la définition du contexte correspondant et de ses éventuelles adaptations. Toute modification ultérieure ne sera obtenue que par des commandes d'adaptation des propriétés, libellées au sein même de la musique.

Les définitions d'un objet graphique (*graphical object* abrégé en *grob*) constituent une catégorie *spécifique* de propriétés de contexte, dans la mesure où leur structure, enregistrement et utilisation diffèrent des propriétés de contextes habituelles.

Contrairement aux propriétés de contextes habituelles, les définitions de *grob* sont subdivisées en propriétés de *grob*. Un *grob* est créé par un graveur lors de l'interprétation d'une expression musicale et reçoit ses propriétés initiales à partir de la définition de *grob* en cours dans le contexte du graveur. Le graveur (ou tout autre « agent » de LilyPond) peut alors ajouter ou modifier des propriétés à cet objet, sans pour autant affecter la définition du *grob* dans ce contexte.

Ce que LilyPond appelle « propriétés de *grob* » dans le cadre de l'affinage par l'utilisateur sont en fait les propriétés de la définition d'un objet dans un contexte.

Contrairement aux propriétés de contexte habituelles, les définitions d'un *grob* doivent être enregistrées pour pouvoir garder trace de ses composants, les propriétés individuelles du *grob* (ainsi que leurs sous-propriétés) séparément. Il sera dès lors possible de définir ces composants dans différents contextes et ainsi disposer d'une définition globale du *grob* à l'instant où la création de cet objet assemblera les éléments relatifs aux différents contextes attachés au contexte en cours et ceux qui l'englobent.

Les définitions de *grob* se manipulent à l'aide des commandes \override et \revert, et leur nom commence par une capitale (comme 'NoteHead') alors que les propriétés de contexte ordinaires – elles commencent par une minuscule – se manipulent avec \set et \unset.

Les instructions spéciales \tweak et \overrideProperty modifient les propriétés de *grob* en court-circuitant totalement les propriétés de contexte. En fait, elles capturent les *grobs* au moment de leur création pour y injecter directement des propriétés soit émanant d'un événement musical retouché par un \tweak, soit lorsqu'ils sont d'une qualité particulière (un \overrideProperty).

35.6 Les commandes \tweak et \single

L'utilisation d'un \override pour modifier les propriétés d'un objet graphique affectera toutes les instances de l'objet en question au sein du contexte, et ce dès son apparition. Il peut parfois être préférable de n'affecter qu'un seul objet en particulier plutôt que tous les objets du contexte. C'est là le rôle de l'instruction \tweak, dont la syntaxe est :

```
\tweak [objet-de-rendu.]objet-propriété valeur
```

La commande \tweak s'applique à l'objet qui apparaît immédiatement après valeur. Mention de l'*objet-de-rendu* est requise pour lever toute ambiguïté dans le cas où l'expression musicale viendrait à générer indirectement d'autres objets de type différent – par exemple, NoteHead est à l'origine de Stem.
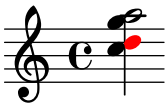
Pour une introduction à la syntaxe et l'utilisation des retouches, voir le chapitre Section "Méthodes de retouche" dans *Manuel d'initiation*.

Entre autres éléments qui sont susceptibles de se produire au même instant, nous citerons :

- les têtes de notes au sein d'un accord,
- les signes d'articulation,
- les liaisons de prolongation sur des notes d'un accord,
- les crochets de n-olets démarrant au même instant

Dans l'exemple suivant, l'une des têtes de note de l'accord est colorisée, et l'aspect d'une autre est changé.

```
< c''
  \tweak color #red
  d''
  g''
  \tweak duration-log 1
  a''
> 4
```
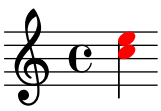


L'instruction \tweak permet aussi de modifier l'aspect d'une liaison :

```
\relative { c'-\tweak thickness 5 ( d e f) }
```



La commande \tweak peut aussi s'appliquer à un accord, auquel cas elle affecte chacune de ses notes :
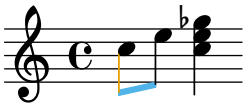
```
{ \tweak color #red <c'' e''>4 }
```



La commande \tweak simple ne saurait servir à modifier un élément qui ne serait pas explicitement mentionné dans le fichier source. C'est notamment le cas des hampes, ligatures automatiques ou altérations, dans la mesure où elles seront ultérieurement générées et après les têtes de note (objets NoteHead), plutôt qu'au fil des éléments musicaux saisis.

De tels objets créés indirectement ne peuvent être ajustés que par une forme développée de la commande \tweak, autrement dit lorsque l'objet est explicitement mentionné :
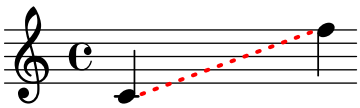
```
\tweak Stem.color #(universal-color 'orange)
\tweak Beam.color #(universal-color 'skyblue) c''8 e''
<c'' e'' \tweak Accidental.font-size -3 ges''>4
```



La commande \tweak ne peut non plus servir à modifier clefs ou métriques, puisqu'elles seront immanquablement séparées du \tweak par l'insertion automatique d'autres éléments requis pour spécifier le contexte.

Plusieurs commandes \tweak en enfilade permettent d'affecter un même élément de notation :

```
c'
  -\tweak springs-and-rods #ly:spanner::set-spacing-rods
  -\tweak minimum-length 15
  -\tweak style #'dashed-line
  -\tweak dash-fraction 0.2
  -\tweak thickness 3
  -\tweak color #red
  \glissando
f''
```
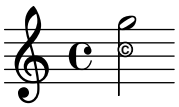


Vous pouvez examiner le flux musical généré par une portion d'un fichier source, y compris les éléments automatiquement insérés, en suivant les indications portées à la rubrique Section "Affichage d'expressions musicales" dans *Extension de LilyPond*. Ceci s'avère tout à fait approprié pour déterminer ce qui peut se modifier à l'aide d'un \tweak ou bien aider à rectifier votre source de telle sorte que le \tweak produise ses effets.

La commande \single prend un ou plusieurs \override (destinés à s'appliquer à un instant musical donné ou au-delà) et les convertit en un *tweak* **unique** qui s'appliquera aux objets spécifiques alors créés.

Le fichier ly/property-init.ly contient de nombreuses définitions de multiples commandes \override qui peuvent alors faire l'objet d'un \single. Par exemple, la fonction \easyHeadsOn peut s'utiliser avec \single pour n'affecter qu'une seule des notes d'un accord :

```
\relative c' {
  < \single \easyHeadsOn c' g' >2
}
```



Voir aussi

Manuel d'initiation : Section "Méthodes de retouche" dans *Manuel d'initiation*.

Manuel d'extension : Section "Affichage d'expressions musicales" dans *Extension de LilyPond*.

Problèmes connus et avertissements

Lorsqu'il y a plusieurs liaisons de prolongation dans un accord, la commande `\tweak` ne permet de modifier les points de contrôle que pour la première rencontrée dans le fichier source.

35.7 La commande `\offset`

Bien qu'il soit possible d'affecter de nouvelles valeurs aux propriétés d'un objet graphique à l'aide des commandes `\override`, `\tweak` ou `\overrideProperty`, il est souvent plus pratique de modifier de telles propriétés par rapport à une valeur par défaut. Ceci est la raison d'être de la commande `\offset`.

La commande `\offset` répond à la syntaxe suivante :

```
[-]\offset propriété décalages élément
```

La commande `\offset` agit par addition du contenu de *décalages* au réglage par défaut de la propriété *propriété* de l'objet graphique indiqué par *élément*.

Selon la manière dont la commande est formulée, `\offset` agira tantôt comme un `\tweak`, tantôt comme un `\override`. Les différences entre ces utilisations seront abordées après avoir recensé les propriétés qui peuvent être soumises à un `\offset`.

Propriétés acceptant des décalages

Bon nombre de propriétés d'objet graphique, mais pas toutes, peuvent faire l'objet d'un décalage. Si d'aventure *propriété* ne peut être affectée, l'objet restera inchangé et sera émis un message d'avertissement. En pareil cas, l'objet doit être modifié par un `\override` ou un `\tweak`.

Il est toujours possible de procéder à tâtons et laisser les avertissement indiquer si tel objet peut ou ne peut pas être soumis à `\offset`. Néanmoins, une approche plus systématique est possible.

Les critères énoncés ci-après déterminent l'égibilité d'une propriété à être modifiée par la commande `\offset`.

- La propriété possède un « réglage par défaut » au niveau de la définition de l'objet graphique. Les propriétés en question sont listées, pour chacun des *grobs*, dans Section "All layout objects" dans *Référence des propriétés internes* – on les trouvera aussi dans le fichier `scm/define-grobs.scm`.
- La propriété prend une valeur numérique. Les valeurs numériques comprennent `number`, liste de `numbers`, `number-pair` et `number-pair-list`. Les pages de Section "All layout objects" dans *Référence des propriétés internes* répertorient le type de donnée propre à chaque propriété. Peu importe que le réglage par défaut soit une fonction.
- La propriété ne saurait constituer une « sous-propriété » – une propriété résidant au sein d'une autre propriété.
- Les propriétés réglées sur des valeurs infinies ne peuvent faire l'objet d'un décalage. Il n'y a aucun moyen d'influencer l'infini, qu'il soit positif ou négatif.

Les exemples qui suivent s'arrêtent sur plusieurs propriétés d'objet graphique au regard des critères énoncés ci-dessus.

- Propriétés qui peuvent être décalées

  `Hairpin.height`

  Cette propriété n'est pas une sous-propriété, et est référencée à Section "Hairpin" dans *Référence des propriétés internes*. En tant que valeur, elle prend une « dimension, exprimée en espace de portée » réglée à `0.6666` – clairement une valeur `number` non infini.

`Arpeggio.positions`
: La page Section “Arpeggio” dans *Référence des propriétés internes* référence une propriété `positions` qui accepte une « paire de nombres ». Sa valeur par défaut est `ly:arpeggio::positions` – une fonction de rappel qui sera évaluée au cours de la phase de typographie pour donner une paire de nombres pour tout objet `Arpeggio`.

- Propriétés qui ne peuvent être décalées

`Hairpin.color`
: Aucune référence à `color` n'est mentionnée dans Section “Hairpin” dans *Référence des propriétés internes*.

`Hairpin.circled-tip`
: La référence à `Hairpin.circled-tip` dans Section “Hairpin” dans *Référence des propriétés internes* indique que cette propriété prend une valeur `boolean`. Les booléens ne sont pas des nombres.

`Stem.details.lengths`
: Bien que mentionnée dans Section “Stem” dans *Référence des propriétés internes* et ayant par défaut une liste de `numbers`, il s'agit d'une « sous-propriété ». Il n'existe à ce jour aucune prise en charge des « propriétés imbriquées ».

`\offset` en tant que dérogation

Lorsque *élément* est un nom d'objet graphique comme `Arpeggio` ou `Staff.OttavaBracket`, le comportement de la commande `\offset` est assimilable à un `\override` sur le type d'objet spécifié.

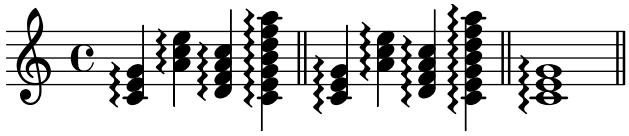
`\offset` *propriété décalages* [*contexte.*]*NomGrob*

Notez bien qu'il n'y a jamais de tiret avant un comportement « dérogatoire », tout comme il n'y en a jamais avec la commande `\override` elle-même.

L'exemple suivant utilise la forme « dérogation » pour allonger les arpeggios affichés dans la première mesure, jusqu'à couvrir l'accord dans son intégralité. Les arpeggios sont étirés d'un demi espace de portée à leur sommet ainsi qu'à leur base. Est aussi indiquée la même opération sur le premier accord à l'aide d'une simple dérogation appliquée à la propriété `positions`. Cette méthode n'est pas la plus illustrative pour « étirer d'un demi espace de portée » dans la mesure où les extrémités doivent être spécifiées en coordonnées absolues plutôt que relatives. De plus, des dérogations individuelles seraient nécessaires pour les autres accords en raison de leurs différentes amplitude et position.

```
arpeggioMusic = {
  <c' e' g'>\arpeggio <a' c'' e''>\arpeggio
  <d' f' a' c''>\arpeggio <c' e' g' b' d'' f'' a''>\arpeggio
}

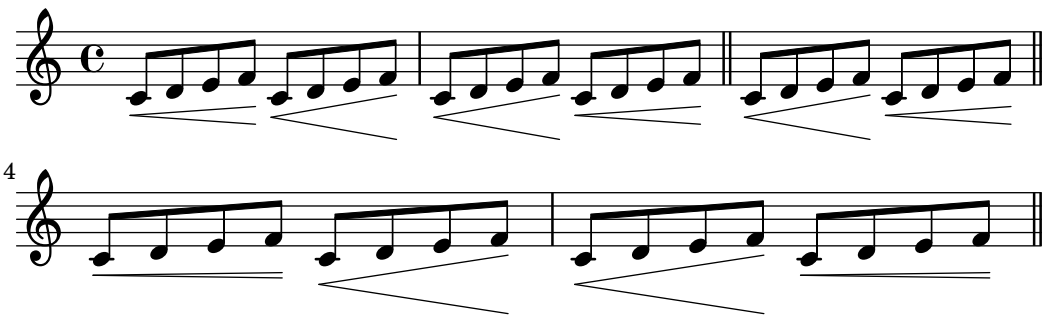
{
  \arpeggioMusic
  \bar "||"
  \offset positions #'(-0.5 . 0.5) Arpeggio
  \arpeggioMusic
  \bar "||"
  \once \override Arpeggio.positions = #'(-3.5 . -0.5)
  <c' e' g'>1\arpeggio
  \bar "||"
}
```



Dans cette utilisation d'*override*, `\offset` peut se préfixer de `\once` ou `\temporary` et être annulé à l'aide d'un `\revert` suivi de *propriété* – voir Section "Fonctions de substitution intermédiaires" dans *Extension de LilyPond*. Ceci tient au fait que `\offset` crée effectivement un `\override` de *propriété*.

```
music = { c'8\< d' e' f'\! }

{
  \music
  \offset height 1 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
  \once \offset height 1 Hairpin
  \music \music
  \bar "||"
  \override Hairpin.height = 0.2
  \music
  \temporary \offset height 2 Hairpin
  \music
  \music
  \revert Hairpin.height
  \music
  \bar "||"
}
```

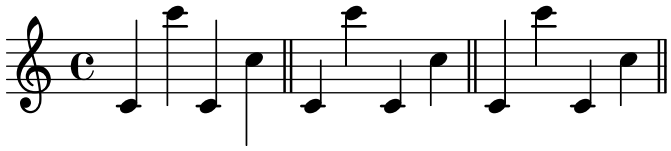


Tout comme `\override`, la forme « dérogation » de `\offset` peut s'utiliser avec `\undo` et `\single`.

```
longStem = \offset length 6 Stem

{
 \longStem c'4 c''' c' c''
 \bar "||"
 \undo \longStem c'4 c''' c' c''
 \bar "||"
 \single \longStem c'4 c''' c' c''
 \bar "||"
```

```
}
```



\offset en tant qu'affinage
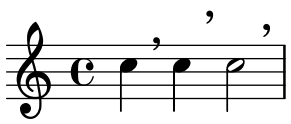
Lorsque *élément* est une expression musicale, comme (ou \arpeggio, le résultat sera la même expression musicale à laquelle aura été appliqué un affinage.

```
[-]\offset [NomGrob.]propriété décalages expression-musicale
```

La syntaxe de \offset dans sa forme « affinage » est en tout point analogue à la commande \tweak, autant dans l'ordre des arguments que dans la présence ou non du tiret l'introduisant.

L'exemple suivant utilise la forme « affinage » pour ajuster le positionnement vertical de l'objet BreathingSign. Les effets de la simple commande \tweak sont présent à titre de comparaison. Leur syntaxe est équivalente. Néanmoins, le résultat de \tweak est moins intuitif dans la mesure où BreathingSign.Y-offset est calculé en référence à la ligne médiane. Il n'est pas nécessaire de savoir comment se calcule Y-offset dans le cas d'un \offset.

```
{
  c''4
  \breathe
  c''4
  \offset Y-offset 2 \breathe
  c''2
  \tweak Y-offset 3 \breathe
}
```
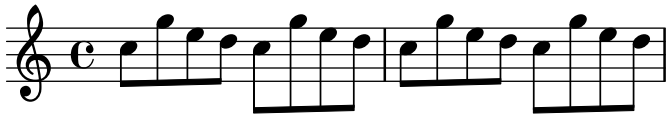


Dans cet exemple, les objets affinés étaient créés directement à partir du code saisi : la commande \breathe était une instruction explicite pour renvoyer un objet BreathingSign. Puisque la cible de la commande était sans ambiguïté, point n'était besoin de spécifier le nom de l'objet. Cependant, lorsqu'un objet est créé *indirectement*, mention du nom de l'objet devient requise. Il en va de même pour la commande \tweak.

Dans l'exemple qui suit, l'objet Beam est abaissé de deux espaces de portée par application de \offset à la propriété positions.

La première application de \offset requiert mention du nom de l'objet puisque rien dans le code ne crée explicitement de ligature. Dans la seconde application, la ligature est explicitement créée par l'expression musicale [, ce qui dispense de mentionner le nom de l'objet. Cette deuxième application comporte par ailleurs un raccourci : un unique number s'appliquera aux deux membres d'un number-pair.

```
{
  c''8 g'' e'' d''
  \offset Beam.positions #'(-2 . -2)
  c''8 g'' e'' d''
  c''8 g'' e'' d''
  c''8-\offset positions -2 [ g'' e'' d'']
}
```



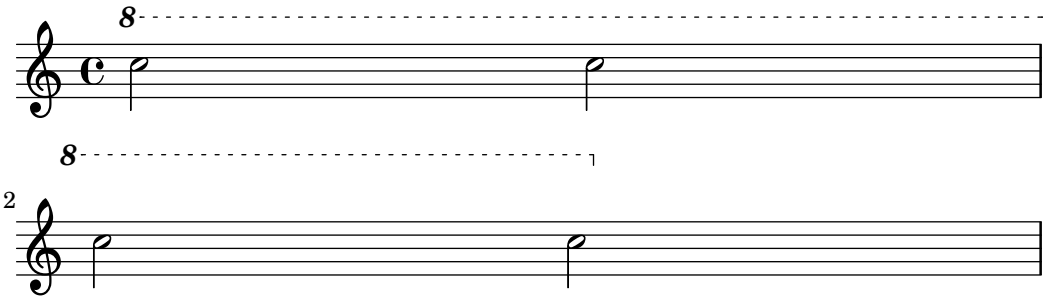
\offset et les bandeaux avec rupture

Il est aussi possible de modifier indépendamment les segments d'un objet étendu rencontrant des sauts de ligne. Dans ce cas, *décalages* est constitué d'une liste de valeurs pour le type de donnée requis par la propriété.

Utilisée de telle manière, la commande \offset est similaire à la commande \alterBroken – voir Section 36.4.1 [Modification de bandeaux avec rupture], page 804. Cependant, et contrairement à la commande \alterBroken, les valeurs fournies à \offset sont relatives.

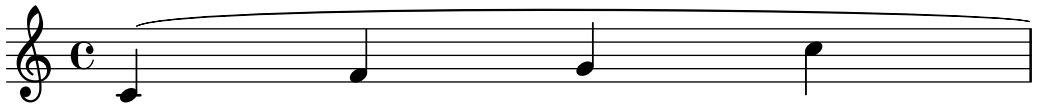
Dans l'exemple suivant est déplacé l'objet « segmenté » OttavaBracket au travers de sa propriété staff-padding. Puisque cette propriété est affectée d'un number, *décalages* est alimenté d'une liste de numbers afin de prendre en compte les deux segments créés par le saut de ligne. La portion de crochet de la première ligne n'est en fait pas touchée puisque 0 est ajouté à la valeur par défaut de staff-padding. Le segment de la deuxième ligne est haussé de trois espaces de portée par rapport à sa hauteur par défaut. La hauteur par défaut est de 2, bien qu'il ne soit pas nécessaire de le savoir.

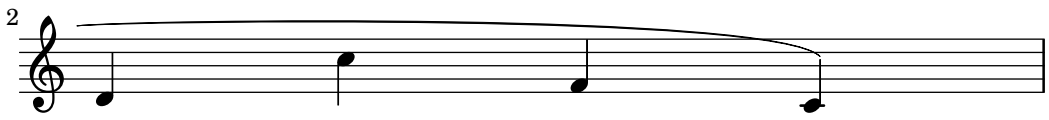
```
{
 \offset staff-padding #'(0 3) Staff.OttavaBracket
  \ottava 1
  c'''2 c'''
  \break
  c'''2 c'''
}
```



L'exemple ci-dessous reproduit les effets de la commande \shape en décalant la propriété control-points de l'objet Slur. Ici, *décalages* est constitué d'une liste de number-pair-lists, une pour chaque segment de la liaison. Cet exemple produit un résultat identique à ce qui est illustré dans Section 36.11 [Modification de l'allure des éléments], page 830.

```
{
  c'4-\offset control-points #'(
              ((0 . 0) (0 . 0) (0 . 0) (0 . 1))
              ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
             ) ( f'4 g' c''
  \break
  d'4 c'' f' c')
}
```





Voir Section “Ajustement du positionnement vertical d'une liaison” dans *Morceaux choisis* pour la création d'une fonction Scheme `\offsetPositions` afin d'aider à gérer le décalage vertical des liaisons.

35.8 Modification de listes associatives

Certaines propriétés configurables par l'utilisateur se présentent en interne comme étant des listes associatives – les puristes diront des *alists*. Une *alist* est en fait constituée de plusieurs paires de *clés* et *valeurs*. La structure d'un liste associative ressemble à :

```
'((clé1 . valeur1)
  (clé2 . valeur2)
  (clé3 . valeur3)
  ...)
```

Dans le cas où cette liste représente les propriétés d'un objet graphique ou bien l'une des variables du bloc \paper, chaque clé peut être modifiée individuellement sans que cela affecte les autres.

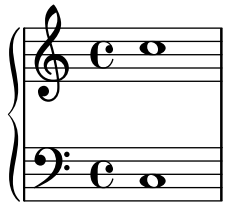
Par exemple, pour réduire l'espacement entre deux portées adjacentes d'un même système, on utilisera la propriété `staff-staff-spacing` qui est attachée à l'objet graphique `StaffGrouper`. Cette propriété est constituée d'une liste de quatre clés : `basic-distance`, `minimum-distance`, `padding` et `stretchability`. Ses réglages par défaut tels que mentionnés à la rubrique *Backend* de la référence des propriétés internes – voir Section “StaffGrouper” dans *Référence des propriétés internes* – sont :

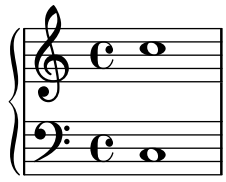
```
'((basic-distance . 9)
  (minimum-distance . 7)
  (padding . 1)
  (stretchability . 5))
```

Afin de rapprocher nos deux portées, il suffit de réduire la valeur (`9`) de la clé `basic-distance` au niveau de celle de la clé `minimum-distance` (`7`). La modification d'une seule clé individuellement peut se réaliser sous la forme d'une *déclaration imbriquée* :

```
% default space between staves
\new PianoStaff <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>

% reduced space between staves
\new PianoStaff \with {
  % this is the nested declaration
  \override StaffGrouper.staff-staff-spacing.basic-distance = 7
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```
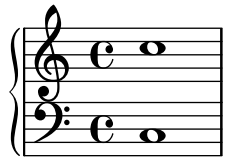




Le recours à une déclaration imbriquée touchera la clé indiquée (`basic-distance` dans l'exemple ci-dessus) sans pour autant modifier les autres clés de la propriété considérée.

Considérons maintenant que nous souhaitions que les portées soient le plus proche possible les unes des autres, à la limite du chevauchement. Il suffirait de mettre les quatre clés à zéro. Nous pourrions saisir quatre déclarations, chacune d'elles touchant une clé. Nous pouvons tout aussi bien redéfinir la propriété en une seule clause, sous la forme d'une liste associative :

```
\new PianoStaff \with {
  \override StaffGrouper.staff-staff-spacing =
    #'((basic-distance . 0)
       (minimum-distance . 0)
       (padding . 0)
       (stretchability . 0))
} <<
  \new Staff { \clef treble c''1 }
  \new Staff { \clef bass   c1   }
>>
```



N'oubliez pas que dès lors qu'une clé n'apparaît pas dans la liste, elle retourne à sa valeur *sauf-mention-contraire*. Autrement dit, dans le cas de `staff-staff-spacing` qui nous occupe, toutes les clés non mentionnées seront ramenées à zéro – à l'exception de `stretchability` qui prend par défaut la valeur de `basic-distance`. Les deux assertions suivantes sont donc équivalentes.

```
\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7))

\override StaffGrouper.staff-staff-spacing =
  #'((basic-distance . 7)
     (minimum-distance . 0)
     (padding . 0)
     (stretchability . 7))
```

L'une des conséquences, parfois involontaire, de ceci est la suppression de réglages standards effectués par un fichier d'initialisation chargé à chaque compilation d'un fichier source. Dans l'exemple précédent, les réglages standards de `padding` et `minimum-distance`, tels que

déterminés par `scm/define-grobs.scm`, se voient ramenés à leur valeur *si-non-définie* ; autrement dit, les deux clés sont mises à zéro. La définition d'une propriété ou d'une variable sous forme de liste associative, quelle qu'en soit la taille, réinitialisera toujours les clés non mentionnées à leur valeur *si-non-définie*. Si telle n'est pas votre intention, nous vous recommandons alors de régler la valeur des clés individuellement par des déclarations imbriquées.

Note : Les déclarations imbriquées ne sont pas fonctionnelles dans le cas des listes associatives des propriétés de contexte – telles `beamExceptions`, `keyAlterations`, `timeSignatureSettings`, etc. Ces propriétés ne sont modifiables qu'au travers d'une complète redéfinition de leur liste associative.

36 Propriétés et contextes utiles

Nous allons voir, au fil des paragraphes qui suivent, différentes approches permettant de fignoler l'apparence d'une partition.

Voir aussi

Manuel d'initiation : Section "Autres sources de documentation" dans *Manuel d'initiation*, Section "Retouche de partition" dans *Manuel d'initiation*.

Manuel de notation : Chapitre 34 [En quoi consiste la référence des propriétés internes], page 782, Chapitre 35 [Modification de propriétés], page 786.

Manuel d'extension : Section "Interfaces pour programmeurs" dans *Extension de LilyPond*.

Fichiers d'initialisation : `scm/define-grobs.scm`.

Morceaux choisis : Section "Affinages" dans *Morceaux choisis*.

Référence des propriétés internes : Section "All layout objects" dans *Référence des propriétés internes*.

36.1 Direction et positionnement

En matière de typographie musicale, l'orientation et le positionnement de nombreux éléments est affaire de goût. Par exemple, les hampes peuvent être ascendantes ou descendantes, les paroles, nuances ou autres indications d'expression peuvent apparaître au-dessus ou en dessous de la portée, les indications textuelles s'alignent tantôt par la gauche, tantôt par la droite, ou être centrées. La plupart de ces choix peuvent être laissés à l'appréciation de LilyPond. Il peut être préférable, dans certains cas, d'imposer l'orientation ou le positionnement des éléments.

36.1.1 Indicateurs de position d'une articulation

Certains positionnements sont opérés par défaut – toujours au-dessus ou toujours en dessous (nuances ou points d'orgue) – alors que d'autres alterneront selon l'orientation des hampes (liaisons ou accents).

Le positionnement par défaut peut être outrepassé à l'aide d'un *indicateur de positionnement*, qui vient s'insérer juste avant l'articulation. LilyPond met à votre disposition trois indicateurs de positionnement : `^` (pour « au-dessus »), `_` (pour « au-dessous »), et `-` (pour « appliquer le positionnement par défaut »). L'indicateur de positionnement n'est pas obligatoire ; LilyPond considère alors qu'il y a un `-`. Un indicateur de positionnement est cependant **obligatoire** dans les cas suivants :

- une commande `\tweak`,
- une commande `\markup`,
- une commande `\tag`,
- les indications de corde, par exemple `-"corde"`,
- les indications de doigté, par exemple `-1`,
- les raccourcis d'articulation, par exemple `-.`, `->` ou `--`.

Les indicateurs de positionnement n'affectent que la note qui suit :

```
\relative {
  c''2( c)
  c2_( c)
  c2( c)
  c2^( c)
}
```



36.1.2 La propriété `direction`

Le positionnement ou l'orientation de nombreux objets de rendu sont gérés par la propriété `direction`.

La propriété `direction` peut prendre la valeur `1`, qui signifie « ascendant » ou « au-dessus », ou `-1`, qui signifie « descendant » ou « au-dessous ». Les symboliques `UP` et `DOWN` peuvent remplacer respectivement `1` et `-1`. Les valeurs `0` ou `CENTER` permettent de réaffecter à la propriété `direction` son comportement par défaut. Certaines commandes prédéfinies permettent par ailleurs de spécifier un comportement en matière d'orientation ou positionnement ; elles sont de la forme

`\xxxUp, \xxxDown et \xxxNeutral`

auquel cas `\xxxNeutral` signifie « retour au comportement par défaut » – voir Section "Objets inclus dans la portée" dans *Manuel d'initiation*.

Dans quelques cas particuliers, comme l'indication d'un *arpeggio*, la valeur affectée à la propriété `direction` déterminera si l'objet doit se placer à gauche ou à droite de son parent. Un `-1` ou `LEFT` signifiera alors « sur la gauche », et un `1` ou `RIGHT` « sur la droite ». Comme de bien entendu, un `0` ou `CENTER` signifiera « appliquer le positionnement par défaut ».

Notez que ces commandes resteront effectives jusqu'à ce qu'elles soient annulées.

```
\relative {
  c''2( c)
  \slurDown
  c2( c)
  c2( c)
  \slurNeutral
  c2( c)
}
```



En matière de musique polyphonique, il est souvent plus judicieux d'utiliser des contextes `Voice` explicites que de modifier l'orientation des objets. Pour de plus amples informations, voir Section 5.2 [Plusieurs voix], page 227.

Voir aussi

Manuel d'initiation : Section "Objets inclus dans la portée" dans *Manuel d'initiation*.

Manuel de notation : Section 5.2 [Plusieurs voix], page 227.

36.2 Distances et unités de mesure

LilyPond considère deux types de distances : les distances absolues et les distances relatives ou extensibles.

Les distances absolues permettent de spécifier les marges, indentations et autres détails de mise en page ; elles s'expriment par défaut en millimètres. Vous pouvez utiliser d'autres systèmes de mesure dès lors que la quantité est suivie de la mesure : `\mm`, `\cm`, `\in` (pouces), `\pt` (points, 1/72,27 pouce) ou `\bp` (big points).

	mm	**cm**	**in**	**pt**	**bp**
mm	1	0.1	0.0394	2.8453	2.8346
cm	10	1	0.3937	28.4528	28.3465
in	25.4	2.54	1	72.27	72
pt	0.3515	0.0351	0.0138	1	0.996
bp	0.3538	0.0354	0.0139	1.0038	1

Les mesures de mise en page peuvent aussi s'exprimer en unité extensible de portée `\staff-space` (voir ci-après). Pour plus d'information concernant la mise en page, voir la rubrique Chapitre 26 [Mise en forme de la page], page 691.

Les distances relatives ou extensibles s'expriment toujours en « espace de portée » ou, plus rarement, en « demi espace de portée ». L'espace de portée (*staff-space*) correspond à la distance qui sépare deux lignes adjacentes d'une portée. Sa valeur par défaut est déterminée globalement par la taille de portée. Elle peut aussi s'ajuster ponctuellement en jouant sur la propriété `staff-space` de l'objet `StaffSymbol`. Les distances relatives s'ajustent automatiquement dès qu'une modification de la taille globale de portée ou bien de la propriété `staff-space` du `StaffSymbol` intervient. Cependant, les tailles de fonte ne s'ajusteront automatiquement que si la modification touche la taille globale des portées. La taille globale de portée permet ainsi de gérer l'aspect général de la partition – voir Section 27.2 [Définition de la taille de portée], page 707.

Lorsque seulement une portion de partition doit apparaître dans une taille, comme par exemple une portée d'ossia ou une note de bas de page, influer sur la taille globale de portée affecterait l'intégralité de la partition. Il convient donc dans ce cas de modifier à la fois la propriété `staff-space` du `StaffSymbol` et la taille des fontes. La fonction Scheme `magstep` est tout spécialement chargée d'adapter une modification du `staff-space` aux fontes. Pour de plus amples informations, reportez-vous à la rubrique Section "Longueur et épaisseur des objets" dans *Manuel d'initiation*.

Voir aussi

Manuel d'initiation : Section "Longueur et épaisseur des objets" dans *Manuel d'initiation*.

Manuel de notation : Section 27.2 [Définition de la taille de portée], page 707, Chapitre 26 [Mise en forme de la page], page 691.

36.3 Dimensions

Les dimensions d'un objet graphique spécifient la position des bords droit et gauche ainsi que des bords supérieur et inférieur de la boîte englobante de ces objets, en tant que distance par rapport au point de référence de l'objet et en unité d'espace de portée. Ces positions sont normalement codées sous la forme de deux paires Scheme. Par exemple, la commande de *markup* `\with-dimensions` prend trois arguments, les deux premiers étant des paires Scheme donnant la position des bords gauche et droit et celle des bords inférieur et supérieur :

```
\with-dimensions #'(-5 . 10) #'(-3 . 15) argument3
```

Ceci spécifie une boîte englobante pour *argument3* dont le bord gauche est à −5, le bord droit à 10, le bord inférieur à −3 et le bord supérieur à 15 espaces de portée du point de référence de cet objet.

Pour de plus amples information sur la manière dont sont définies de telles boîtes, y compris les définitions formelles de l'espace horizontal et vertical tel que les commandes de *markup* `\hspace` et `\vspace` le déterminent, voir Section "Le modèle de boîte de LilyPond" dans *Extension de LilyPond*.

Voir aussi

Manuel de notation : Section A.1.8 [Other markup commands], page 902, (documentation sur des commandes similaires à `\with-dimensions`, telles que `\with-dimension` ou `\with-dimension-from`), Section 36.2 [Distances et unités de mesure], page 803.
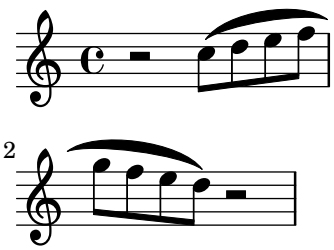
36.4 Extenseurs et prolongateurs

De nombreux objets de notation musicale s'étendent sur plusieurs notes, voire même sur plusieurs mesures. Il en va ainsi des liaisons, ligatures, crochets de n-olet, crochets de reprise, crescendos, trilles ou glissandos. Ces objets, que l'on englobe sous l'appellation « d'extenseurs », sont pourvus de propriétés spécifiques destinées à contrôler leur apparence et leur comportement. Ils font aussi appel à des méthodes spécifiques en matière de retouches liées au fait que certains peuvent être rompus lors d'un saut de ligne.

36.4.1 Modification de bandeaux avec rupture

Lorsqu'un bandeau ou l'extension d'un objet rencontre un saut de ligne ou une rupture, chacun de ses tronçons hérite des attributs de l'objet originel. Par voie de conséquence, la modification d'une extension avec rupture produira les mêmes effets sur chacun de ses segments. Dans l'exemple ci-dessous, la modification apportée à `thickness` s'applique aussi bien avant qu'après le saut de ligne.

```
\relative c'' {
  r2
  \once\override Slur.thickness = 10
  c8( d e f
  \break
  g8 f e d) r2
}
```



La commande `\alterBroken` permet de modifier indépendamment l'apparence de chacune des parties d'un bandeau. Selon le cas, cette commande générera soit un `\override`, soit un `\tweak` qui s'appliquera à la propriété du bandeau.

La commande `\alterBroken` répond à la syntaxe :

`[-]\alterBroken` *`propriété valeurs élément`*

L'argument *valeurs* est constitué d'une liste de valeurs, une pour chaque tronçon. Lorsque *élément* est un nom d'objet graphique, tels `Slur` ou `Staff.PianoPedalBracket`, il en résulte un
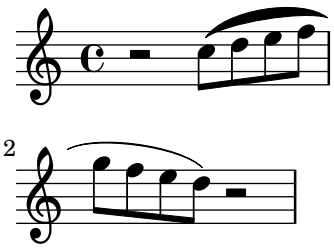
\override du type de *grob* spécifié. Lorsque *élément* est une expression musicale comme « (» ou « [», en résulte cette même expression musicale à laquelle s'applique un \tweak.

Le tiret introduisant la commande \alterBroken est impératif dans le cadre d'un \tweak ; il est superflu pour un \override.

Dans sa variante \override, la commande \alterBroken peut se préfixer d'un \once ou d'un \temporary qui seront annulés par un \revert suivi de la *propriété* – voir Section "Fonctions de substitution intermédiaires" dans *Extension de LilyPond*.

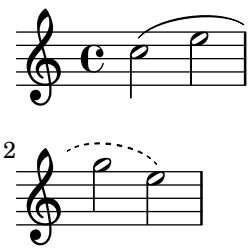
Le code ci-dessous applique un \override indépendant à chacun des segments du phrasé de l'exemple précédent :

```
\relative c'' {
  r2
  \alterBroken thickness #'(10 1) Slur
  c8( d e f
  \break
  g8 f e d) r2
}
```



La commande \alterBroken peut s'utiliser avec tout objet étendu, y compris Tie, PhrasingSlur, Beam et TextSpanner. Par exemple, un éditeur préparant une édition critique pourrait faire ressortir l'absence d'une partie de liaison de phrasé dans l'une des sources, en optant pour un tracé pointillé du seul segment ajouté. L'exemple ci-dessous illustre la manière de procéder, ici avec la variante \tweak de la commande :

```
% The empty list is conveniently used below, because it is the
% default setting of dash-definition, resulting in a solid curve.
\relative {
  c''2-\alterBroken dash-definition #'(() ((0 1.0 0.4 0.75))) \(e
  \break
  g2 e\)
}
```



Il est important de considérer que \alterBroken affectera à chaque portion d'un bandeau interrompu la valeur correspondante de *valeurs*. Si d'aventure il y a moins de valeurs que de tronçons, toute portion additionnelle se verra assigner une liste vide. Ceci peut conduire à des résultats inattendus dans le cas où la propriété de rendu ne bascule pas sur une liste vide par défaut. En pareil cas, chaque segment devrait se voir assigner une valeur appropriée.

Problèmes connus et avertissements

Les sauts de ligne peuvent intervenir à différents endroits pour répondre à des modifications de mise en forme. Les réglages adoptés par `\alterBroken` peuvent devenir inadaptés si le bandeau n'est plus rompu ou est découpé en plus de segments que prévu. L'introduction explicite d'un `\break` peut alors pallier ces situations.

La commande `\alterBroken` est inopérante sur les propriétés d'un bandeau qui sont traitées avant la procédure de saut de ligne, comme `direction`.

Voir aussi

Manuel d'extension : Section "Retouches complexes" dans *Extension de LilyPond*.
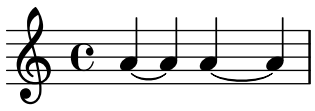
36.4.2 Affectation d'une longueur minimale aux extenseurs

L'interface `spanner-interface` fournit trois propriétés qui s'appliquent à certains extenseurs.

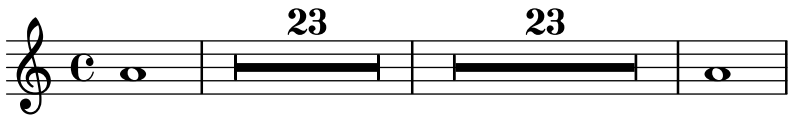
La propriété `minimum-length`

La longueur minimale d'un extenseur est déterminée par la propriété `minimum-length`. Au plus sa valeur est élevée, au plus l'espacement des notes qui le bornent sera grand. Forcer sa valeur restera néanmoins sans effet pour un certain nombre d'extenseurs dont la longueur dépend d'autres considérations. Voici quelques exemples de mise en œuvre de cette propriété.
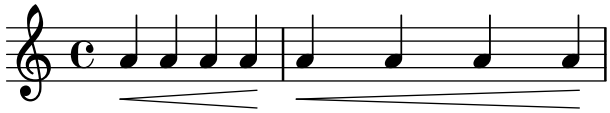
```
a'~ a'
a'
% increase the length of the tie
-\tweak minimum-length 5
~ a'
```



```
\relative \compressMMRests {
  a'1
  R1*23
  % increase the length of the rest bar
  \once \override MultiMeasureRest.minimum-length = 20
  R1*23
  a1
}
```
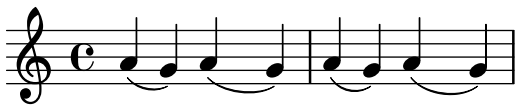



```
\relative {
  a' \< a a a \!
  % increase the length of the hairpin
  \override Hairpin.minimum-length = 20
  a \< a a a \!
}
```



Cette propriété permet aussi de jouer sur l'envergure d'une liaison d'articulation ou de phrasé.

```
\relative {
  a'( g)
  a
  -\tweak minimum-length 5
  ( g)

  a\( g\)
  a
  -\tweak minimum-length 5
  \( g\)
}
```
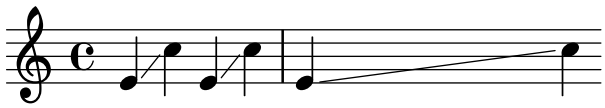


Certains objets requièrent un appel explicite à la procédure `set-spacing-rods` pour que la propriété `minimum-length` produise ses effets. La propriété `set-spacing-rods` doit alors prendre pour valeur `ly:spanner::set-spacing-rods`. Par exemple, la longueur minimale d'un glissando ne pourra être forcée tant que la propriété `springs-and-rods` n'aura pas été définie :

```
% default
e' \glissando c''

% not effective alone
\once \override Glissando.minimum-length = 20
e' \glissando c''
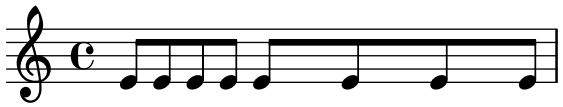
% effective only when both overrides are present
\once \override Glissando.minimum-length = 20
\once \override Glissando.springs-and-rods =
                  #ly:spanner::set-spacing-rods
e' \glissando c''
```



Il en va de même pour l'objet `Beam` (ligature) :

```
% not effective alone
\once \override Beam.minimum-length = 20
e'8 e' e' e'

% effective only when both overrides are present
\once \override Beam.minimum-length = 20
\once \override Beam.springs-and-rods =
                  #ly:spanner::set-spacing-rods
e'8 e' e' e'
```



La propriété minimum-length-after-break

La propriété minimum-length-after-break peut s'utiliser pour dimensionner le tronçon d'extenseur placé après un saut de ligne. Tout comme la propriété minimum-length, elle nécessite souvent, pour prendre effet, de régler la propriété springs-and-rods sur ly:spanner::set-spacing-rods.

```
{
  \once \override Tie.minimum-length-after-break = 20
  a1~
  \break
  a1

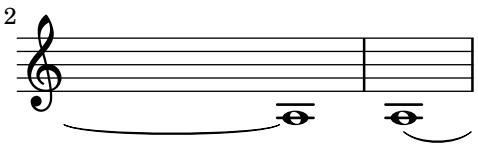
  \once \override Slur.minimum-length-after-break = 20
  a1(
  \break
  d'1)

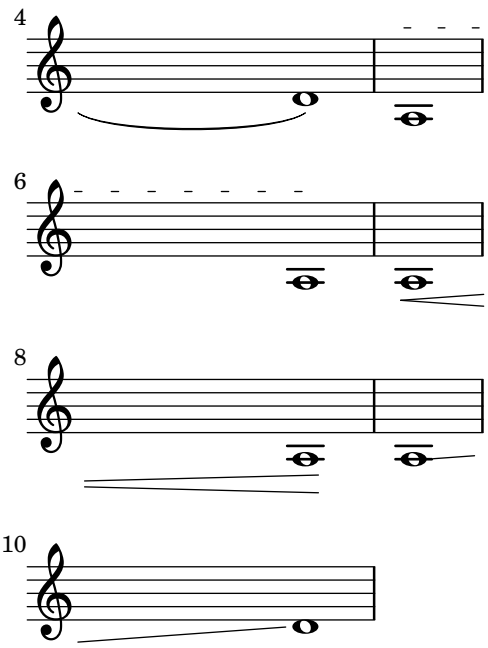
  \once \override TextSpanner.springs-and-rods =
    #ly:spanner::set-spacing-rods
  \once \override TextSpanner.minimum-length-after-break = 20
  a1\startTextSpan
  \break
  a1\stopTextSpan

  \once \override Hairpin.after-line-breaking = ##t
  \once \override Hairpin.to-barline = ##f
  \once \override Hairpin.minimum-length-after-break = 20
  a1\<
  \break
  a1\!

  \once \override Glissando.springs-and-rods =
    #ly:spanner::set-spacing-rods
  % for completeness; not necessary for manual breaks
  \once \override Glissando.breakable = ##t
  \once \override Glissando.after-line-breaking = ##t
  \once \override Glissando.minimum-length-after-break = 20
  a1\glissando
  \break
  d'1
}
```



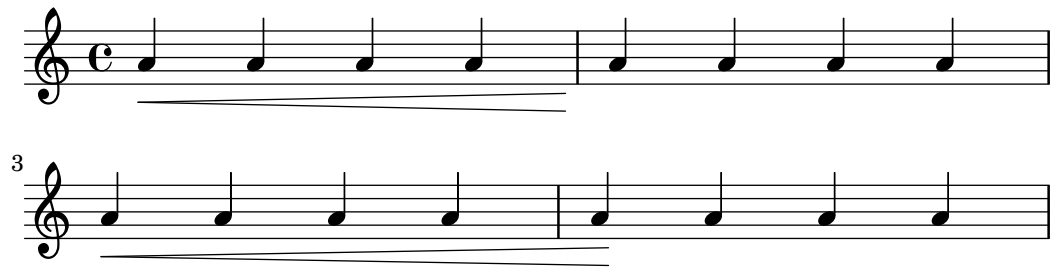




36.4.3 Contrôle du point d'arrêt des bandeaux

La propriété `to-barline` fournie par la `spanner-interface`, activée par défaut, permet aux soufflets et autres extenseurs finissant sur la première note d'une mesure de s'arrêter visuellement au niveau de la barre de mesure qui la précède. Le fait de désactiver la propriété `to-barline` aura pour effet de prolonger l'extenseur au delà de la barre de mesure et jusqu'à la note qui le borne :

```
\relative {
  a' \< a a a a \! a a a \break
  \override Hairpin.to-barline = ##f
  a \< a a a a \! a a a
}
```
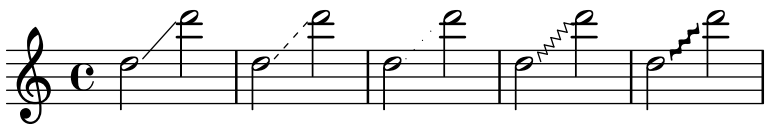


Cette propriété n'est pas opérationnelle pour tous les extenseurs. Il serait en effet quelque peu surprenant de l'activer (lui affecter `#t`) dans le cas d'une liaison d'articulation ou de phrasé !

36.5 Styles de ligne

La Section "line-interface" dans *Référence des propriétés internes* regroupe tous les objets affichant des lignes. Tous les objets pris en charge par cette interface peuvent adopter différents styles de ligne. L'exemple ci-dessous indique les différentes valeurs disponibles.

```
\relative {
  d''2 \glissando d'2
  \once \override Glissando.style = #'dashed-line
  d,2 \glissando d'2
  \override Glissando.style = #'dotted-line
  d,2 \glissando d'2
  \override Glissando.style = #'zigzag
  d,2 \glissando d'2
  \override Glissando.style = #'trill
  d,2 \glissando d'2
}
```
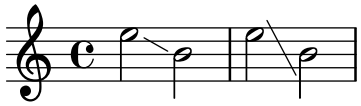


Certains objets peuvent adopter d'autres styles spécifiques.

36.6 Lignes de prolongation

Certaines indications portées à l'attention de l'exécutant – tels *rallentando*, *accelerando* et *trilles* – apparaissent sous la forme d'un texte qui peut s'étendre sur plusieurs mesures à l'aide d'une ligne parfois pointillée ou ondulée.

```
e''2 \glissando b'
\once \override Glissando.bound-details.left.Y = 3
\once \override Glissando.bound-details.right.Y = -2
e''2 \glissando b'
```

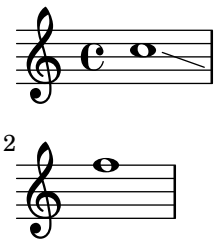


La propriété `Y` est valorisée en unités d'*espace de portée*, la ligne médiane de la portée correspondant au point zéro. Pour le glissando qui nous occupe, il s'agit du `Y` à l'aplomb (coordonnée-X) du centre de la tête de chacune des deux notes, si tant est que la ligne doive s'étendre entre ces deux points.

Si le `Y` n'est pas défini, sa valeur sera calculée en fonction de la position verticale du point d'attachement de l'extenseur.

Dans le cas où l'extenseur est interrompu par un saut de ligne, les terminaisons à cet endroit se gèrent à l'aide des sous-clés `left-broken` et `right-broken` de `bound-details`, comme ci-dessous :

```
\override Glissando.bound-details.right-broken.Y = -3
c''1 \glissando \break
f''1
```



Les sous-propriétés `left` et `right` du `bound-details` disposent d'autres clés modifiables de la même manière que `Y` :

`Y` Détermine l'ordonnée (coordonnée-Y) de la terminaison, avec un décalage en espaces de portée par rapport à la ligne médiane de la portée. Il s'agit par défaut du centre de l'objet d'attachement, qui est le centre vertical de la tête de note pour un glissando.

En ce qui concerne les extenseurs horizontaux, tels ceux comportant du texte ou les trilles, il est fixé à `0`.

`attach-dir`
Détermine le début et la fin de la ligne sur l'axe des abscisses, relativement à l'objet de rattachement. Une valeur de `-1` (ou `LEFT`) aura pour effet de commencer ou terminer la ligne sur la gauche de la tête de note de rattachement.
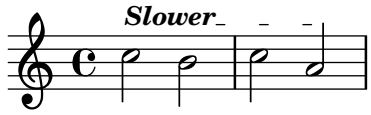
`X` Il s'agit de l'abscisse (coordonnée-X) absolue de la terminaison. Elle se calcule à la volée, et son forçage n'apporte rien de plus.

`stencil` Les extenseurs linéaires peuvent commencer ou finir par un symbole, enregistré dans cette sous-propriété. Elle est conçue pour un usage interne, aussi nous vous conseillons de plutôt recourir à `text`.

`text`

Il s'agit d'un *markup* qui se poursuivra par l'extenseur. C'est la sous-propriété utilisée pour ajouter *cresc.*, *tr* ou autre texte à un extenseur horizontal.

```
\override TextSpanner.bound-details.left.text
   = \markup { \small \bold Slower }
\relative { c''2\startTextSpan b c a\stopTextSpan }
```
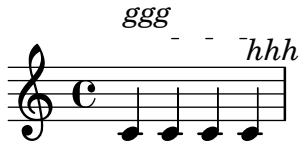


`stencil-align-dir-y`
`stencil-offset`
Lorsqu'aucune de ces deux sous-propriétés n'est définie, le stencil est simplement positionné à l'extrémité, centré sur la ligne telle que définie par les sous-propriétés `X` et `Y`. L'utilisation de `stencil-align-dir-y` ou `stencil-offset` permettra d'aligner le symbole verticalement par rapport au coin de la ligne :

```
\override TextSpanner.bound-details
            .left.stencil-align-dir-y = -2
\override TextSpanner.bound-details
            .right.stencil-align-dir-y = #UP

\override TextSpanner.bound-details.left.text = "ggg"
\override TextSpanner.bound-details.right.text = "hhh"

\relative { c'4^\startTextSpan c c c \stopTextSpan }
```



Vous n'aurez pas manqué de constater qu'une valeur négative place le texte *en haut* – contrairement à ce que l'on serait en droit d'attendre. Ceci est dû au fait
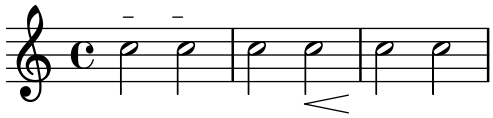
que la valeur `-1` ou `DOWN` signifie « aligner le bord *inférieur* du texte sur la ligne d'extension ». Une valeur égale à `1` ou `UP` alignera le sommet du texte sur cette ligne d'extension.

`arrow` L'activation de cette sous-propriété (lui affecter `#t`) ajoutera à l'extenseur une terminaison en flèche.

`padding` Cette sous-propriété contrôle l'espace qui doit séparer l'extrémité de la ligne et la fin réelle de l'extenseur. Sans ce « décalage », le trait indiquant un glissando commencerait et finirait au beau milieu de chacune des têtes de note.

La fonction `\endSpanners` permet d'interrompre l'extenseur qui vient dès la note suivante. Autrement dit, il ne s'étendra que sur une seule note, ou jusqu'à la prochaine barre de mesure si `to-barline` a été activé et que survient une barre avant la note suivante.

```
\relative c'' {
  \endSpanners
  c2 \startTextSpan c2 c2
  \endSpanners
  c2 \< c2 c2
}
```



L'utilisation de `\endSpanners` permet de s'affranchir d'insérer un `\stopTextSpan` pour clôturer un `\startTextSpan` ou un `\!` pour terminer un soufflet.

Voir aussi

Référence des propriétés internes : Section “Glissando” dans *Référence des propriétés internes*, Section “line-spanner-interface” dans *Référence des propriétés internes*, Section “TextSpanner” dans *Référence des propriétés internes*, Section “TrillSpanner” dans *Référence des propriétés internes*, Section “VoiceFollower” dans *Référence des propriétés internes*.

36.7 Visibilité des objets

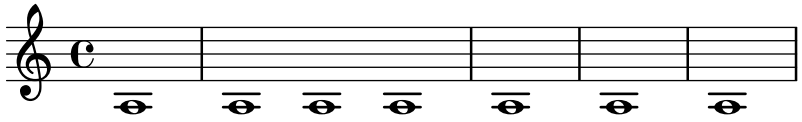
La visibilité des objets de rendu se contrôle de quatre façons différentes : vous pouvez supprimer leur stencil, les rendre transparents, les coloriser en blanc ou bien encore forcer leur propriété `break-visibility`. Les trois premières options peuvent s'appliquer à tous les objets, la dernière étant réservée aux objets *changeables*. Le Manuel d'initiation contient une introduction à ces quatre techniques, à la rubrique Section “Visibilité et couleur des objets” dans *Manuel d'initiation*.

LilyPond met en œuvre quelques techniques particulières adaptées à certains objets ; elles sont couvertes par une rubrique spécifique.

36.7.1 Suppression des stencils

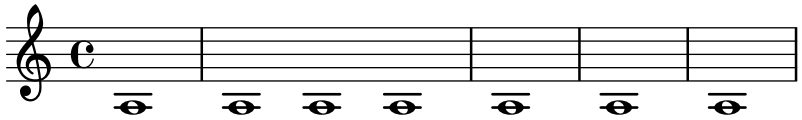
Tout objet de rendu se voit attribuer une propriété `stencil`. Elle est par défaut définie par la fonction chargée de dessiner cet objet. Lorsque cette propriété est désactivée de force – en lui attribuant la valeur `#f` – aucune fonction ne sera appelée ; l'objet ne sera donc pas dessiné. Le retour au comportement par défaut s'opère à l'aide d'un `\revert`.

```
a1 a
\override Score.BarLine.stencil = ##f
a a
\revert Score.BarLine.stencil
a a a
```



Cette opération relativement courante fait l'objet du racourci `\omit` :
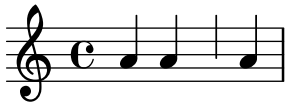
```
a1 a
\omit Score.BarLine
a a
\undo \omit Score.BarLine
a a a
```



36.7.2 Transparence des objets
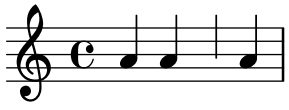
Tout objet de rendu dispose d'une propriété de transparence, qui est par défaut définie à `#f`. Le fait de l'activer rendra l'objet transparent tout en préservant la place qu'il occupe.

```
a'4 a'
\once \override NoteHead.transparent = ##t
a' a'
```



Cette opération relativement courante fait l'objet du racourci `\hide` :

```
a'4 a'
\once \hide NoteHead
a' a'
```



36.7.3 Blanchiment des objets

Tout objet de rendu dispose d'une propriété couleur, par défaut définie à `black` (noir). Le fait de la forcer à `white` (blanc) rendra l'objet indistinct du fond blanc. Néanmoins, lorsque cet objet en recouvre d'autres, la couleur de leurs points de jonction dépendra de l'ordre dans lequel ils sont dessinés, ce qui peut laisser apparaître un fantôme de l'objet blanchi comme ci-dessous :

```
\override Staff.Clef.color = #white
a'1
```



Cet inconvénient peut être évité en modifiant l'ordre dans lequel les objets sont dessinés. Chaque objet de rendu dispose d'une propriété `layer` (calque ou niveau) à laquelle est affecté un nombre entier. Les objets ayant la plus faible valeur sont dessinés en premier, puis les autres, de telle sorte qu'un objet ayant une valeur plus élevée les recouvrira. La plupart des objet ont un `layer` valorisé à `1` – quelques uns, dont `StaffSymbol` et `BarLine`, ont une valeur à `0`. L'ordre d'impression d'objets ayant une même valeur de `layer` est indéterminé.

La clef de l'exemple précédent a par défaut un `layer` à `1` ; elle est donc dessinée après les lignes de la portée – `layer` valorisé par défaut à `0` – et donc les recouvre. Pour changer cet état de fait, l'objet `Clef` doit avoir un `layer` de valeur inférieure, disons `-1`, pour pouvoir être dessiné avant.

```
\override Staff.Clef.color = #white
\override Staff.Clef.layer = #-1
a'1
```
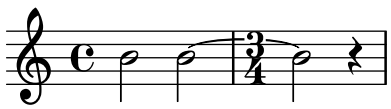


Morceaux choisis

Utilisation de la propriété `whiteout`

Tout objet graphique peut s'imprimer sur un fond blanc afin de masquer une partie des objets qu'il recouvre. Ceci trouve toute son utilité pour améliorer certaines collisions, notamment dans des situations où un repositionnement d'objets est irréaliste. Il faut alors explicitement définir la propriété `layer` afin de contrôler quels objets seront masqués par le fond blanc.

Dans l'exemple ci-dessous, la liaison est en collision avec la métrique ; la situation est améliorée dès lors que la portion de liaison qui traverse la métrique est masquée par l'affectation de la propriété `whiteout` à l'objet `TimeSignature`. Pour ce faire, `TimeSignature` est déplacé sur un calque au-dessus de celui de `Tie` – il reste sur le calque par défaut (1) –, puis le `StaffSymbol` est placé sur un calque supérieur à celui de `TimeSignature` pour ne pas être masqué.

```
{
  \override Score.StaffSymbol.layer = 4
  \override Staff.TimeSignature.layer = 3
  b'2 b'~
  \once \override Staff.TimeSignature.whiteout = ##t
  \time 3/4
  b' r4
}
```



36.7.4 Utilisation de `break-visibility`

La plupart des objets de rendu ne sont imprimés qu'une seule fois ; certains cependant, tels les barres de mesure, clefs, métriques ou armures, apparaîtront deux fois lors d'un saut de ligne – une première fois en fin de ligne, puis à nouveau au début de la ligne suivante. Ces objets, que l'on peut traiter de *changeables* (*breakable* en anglais) disposent de la propriété `break-visibility` spécialement chargée de contrôler leur visibilité aux trois endroits où il sont susceptibles d'apparaître : en début de ligne, en cours de ligne ou en fin de ligne – si tant est qu'un changement s'y produise.

Par exemple, la métrique est imprimée par défaut au début de la première ligne, et nulle part ailleurs. En cas de modification, une nouvelle métrique sera imprimée à l'endroit du changement. Dans le cas où ce changement intervient en fin de ligne, la nouvelle métrique s'imprime au début de la ligne suivante, et une métrique « de précaution » viendra se placer au bout de la ligne précédente.

Ce comportement est géré par la propriété `break-visibility`, dont vous trouverez une explication à la rubrique Section "Visibilité et couleur des objets" dans *Manuel d'initiation*. Cette propriété est constituée d'un vecteur de trois booléens qui, dans l'ordre, déterminent si l'objet sera imprimé à la fin, en cours, et au début d'une ligne – on pourrait aussi dire avant un saut de ligne, là où il n'y a pas de saut de ligne, et après un saut de ligne.

Les huit combinaisons possibles sont aussi disponibles sous la forme de fonctions prédéfinies, regroupées dans le fichier `scm/output-lib.scm`. Le tableau suivant vous les présente ; les trois dernières colonnes indiquent l'endroit où l'objet sera visible.

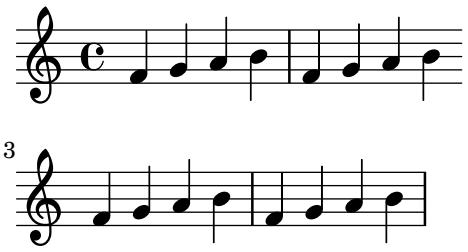
Forme fonctionnelle	**Forme vectorielle**	**Avant saut**	**Hors saut**	**Après saut**
`all-visible`	`#(#t #t #t)`	oui	oui	oui
`begin-of-line-visible`	`#(#f #f #t)`	non	non	oui
`center-visible`	`#(#f #t #f)`	non	oui	non
`end-of-line-visible`	`#(#t #f #f)`	oui	non	non
`begin-of-line-invisible`	`#(#t #t #f)`	oui	oui	non
`center-invisible`	`#(#t #f #t)`	oui	non	oui
`end-of-line-invisible`	`#(#f #t #t)`	non	oui	oui
`all-invisible`	`#(#f #f #f)`	non	non	non

Les réglages par défaut de la propriété `break-visibility` diffèrent selon l'objet de rendu. Le tableau suivant présente, pour la plupart des objets comportant la propriété `break-visibility`, ces réglages par défaut.

Objet de rendu	**Contexte habituel**	**Réglage par défaut**
`BarLine`	`Score`	calculé
`BarNumber`	`Score`	`begin-of-line-visible`
`BreathingSign`	`Voice`	`begin-of-line-invisible`
`Clef`	`Staff`	`begin-of-line-visible`
`Custos`	`Staff`	`end-of-line-visible`
`Divisio`	`Staff`	`begin-of-line-invisible`
`DoublePercentRepeat`	`Voice`	`begin-of-line-invisible`
`KeySignature`	`Staff`	`begin-of-line-visible`
`ClefModifier`	`Staff`	`begin-of-line-visible`
`RehearsalMark`	`Score`	`end-of-line-invisible`
`TimeSignature`	`Staff`	`all-visible`

Voici un exemple d'utilisation de la forme vectorielle pour contrôler la visibilité des barres de mesure :

```
\relative {
  f'4 g a b
  f4 g a b
  % Remove bar line at the end of the current line
  \once \override Score.BarLine.break-visibility = ##(#f #t #t)
  \break
  f4 g a b
  f4 g a b
}
```



Lors d'un forçage de `break-visibility` sous une forme vectorielle, les trois éléments doivent impérativement être mentionnés. Ces formes vectorielles ne sont d'ailleurs pas prises en charge par tous les objets de rendu, et certaines combinaisons peuvent entraîner des erreurs ; nous citerons entre autres :
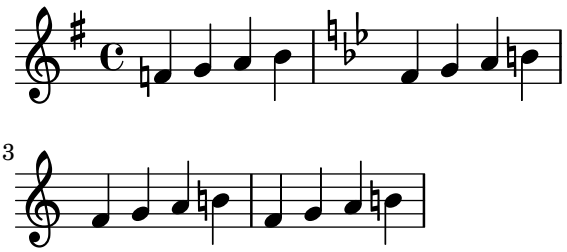
- Une barre de mesure ne peut s'imprimer en début de ligne.
- Un numéro de mesure ne peut apparaître au début de la première ligne, à moins d'être différent de 1.
- Clef – voir ci-après.
- Les répétitions en pourcentage sont soit toutes imprimées, soit aucune. Vous devrez utiliser `begin-of-line-invisible` pour les voir et `all-invisible` pour les masquer.
- Armure – voir ci-après.
- Modificateur de clef – voir ci-après.

36.7.5 Considérations spécifiques

Visibilité après changement explicite
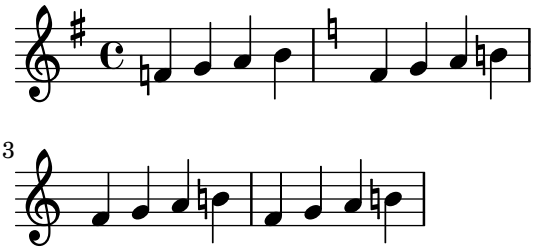
La propriété `break-visibility` contrôle la visibilité des armures ou changements de clef en début de ligne uniquement, donc après un saut. Elle ne produit aucun effet sur la visibilité d'une armure ou d'une clef après un changement explicite de tonalité ou de clef, ni en cours, ni en fin de ligne. Dans l'exemple suivant, l'armure est présente même après le passage en si bémol majeur malgré l'activation de `all-invisible` (*tous invisibles*).

```
\relative {
  \key g \major
  f'4 g a b
  % Try to remove all key signatures
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b
  \break
  f4 g a b
  f4 g a b
}
```



La visibilité lors de ces changements explicites d'armure ou de clef est géré respectivement par les propriétés `explicitKeySignatureVisibility` et `explicitClefVisibility`. Leur fonctionnement est en tout point identique à celui de la propriété `break-visibility` – forme vectorielle à trois éléments ou forme fonctionnelle comme indiqué ci-avant. Toutes deux sont attachées au contexte `Staff` (la portée) et non directement aux objets de rendu ; elles sont de ce fait introduites par une instruction `\set`. Leur valeur par défaut est de toujours imprimer les objets – réglage sur `all-visible`. Ces deux propriétés gèrent uniquement la visibilité des armures et clefs lors d'un changement explicite, et en dehors d'un début de ligne ; il faudra en pareil cas forcer la `break-visibility` de ces objets pour les supprimer.
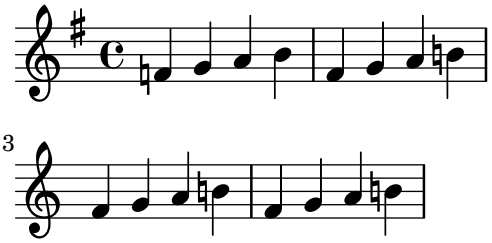
```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Visibilité des bécarres de précaution

L'impression d'altérations de précaution au moment d'un changement explicite de tonalité sera annulée dès lors que vous aurez désactivé la propriété `printKeyCancellation` du contexte `Staff` :
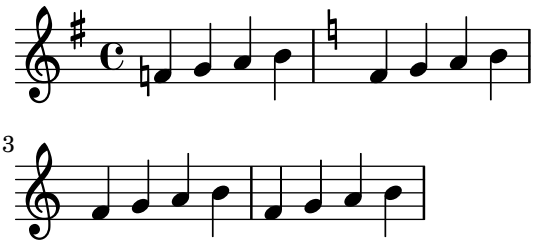
```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \override Staff.KeySignature.break-visibility = #all-invisible
  \key bes \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Avec de tels réglages particuliers, seules les altérations accidentelles permettront d'indiquer le changement de tonalité.
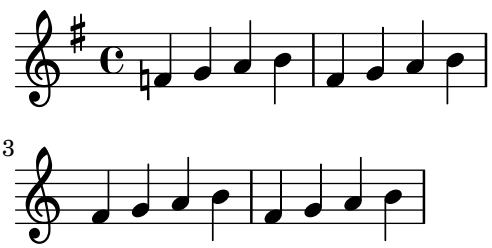
Notez bien que lors d'une bascule en do majeur ou la mineur, seuls les « bécarres d'annulation » permettent d'identifier le changement de tonalité. En pareil cas, désactiver `printKeyCancellation` sera sans effet :

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \set Staff.printKeyCancellation = ##f
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



La suppression des bécarres d'annulation même lors d'un passage en do majeur ou la mineur n'interviendra qu'après modification de la visibilité de l'objet `KeyCancellation` :

```
\relative {
  \key g \major
  f'4 g a b
  \set Staff.explicitKeySignatureVisibility = #all-invisible
  \override Staff.KeyCancellation.break-visibility = #all-invisible
  \key c \major
  f4 g a b \break
  f4 g a b
  f4 g a b
}
```



Clefs transposées

L'indication de transposition d'une clef est produite par l'objet de rendu `ClefModifier`. Sa visibilité étant gérée par héritage direct de l'objet `Clef`, nul n'est besoin de forcer un quelconque `break-visibility` au niveau des objets `ClefModifier` pour éliminer une indication de transposition lorsque la clef est invisible.

Lors d'un changement explicite de clef, la propriété `explicitClefVisibility` gère à la fois le symbole de la clef et l'indication de transposition qui lui est attachée.

Voir aussi

Manuel d'initiation : Section "Visibilité et couleur des objets" dans *Manuel d'initiation*.

36.8 Rotation des objets

Qu'il s'agisse des objets de rendu ou d'éléments textuels sous forme de *markup*, vous pouvez les faire pivoter selon vos désirs et à partir de n'importe quel point. La méthode diffère cependant selon ce que vous désirez manipuler.
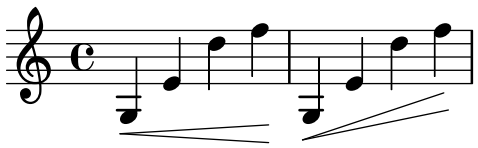
36.8.1 Rotation des objets de mise en forme

Tout objet de rendu disposant de la `grob-interface` est susceptible de pivoter, grâce à la propriété `rotation`. Celle-ci prend en argument une liste de trois éléments : l'angle de rotation – dans le sens inverse des aiguilles d'une montre – ainsi que les coordonnées `x` et `y` du point appartenant à l'objet en question et à partir duquel doit s'effectuer cette rotation. L'angle est exprimé en degrés, les coordonnées en espaces de portée.

L'angle et les coordonnées ne peuvent se déterminer que par tâtonnement.

Il existe assez peu de situations où faire pivoter un objet de mise en forme soit réellement opportun ; en voici une :
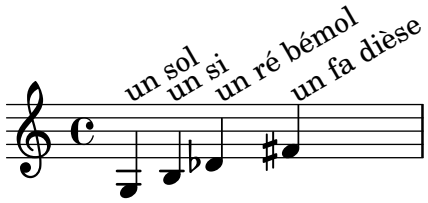
```
g4\< e' d'' f''\!
\override Hairpin.rotation = #'(15 -1 0)
g4\< e' d'' f''\!
```



36.8.2 Rotation des étiquettes

Tout texte faisant l'objet d'un *markup* peut pivoter selon n'importe quel angle, à l'aide de la commande `\rotate`. Celle-ci prend deux arguments : l'angle de rotation exprimé en degrés – dans le sens inverse des aiguilles d'une montre – et le texte à basculer. Il ne s'agit pas ici de faire pivoter les extrémités du texte ; celles-ci récupéreront leurs coordonnées x et y du *markup* pivoté. Dans l'exemple ci-dessous, la propriété `outside-staff-priority` à été fixée à `#f` afin de désactiver l'évitement automatique des collisions qui pourrait repousser certains textes trop haut.

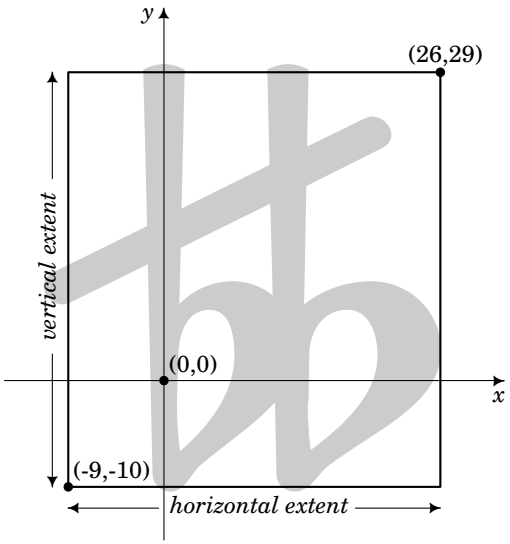
```
\override TextScript.outside-staff-priority = ##f
g4^\markup { \rotate #30 "un sol" }
b^\markup { \rotate #30 "un si" }
des'^\markup { \rotate #30 "un ré bémol" }
fis'^\markup { \rotate #30 "un fa dièse" }
```



36.9 Alignement des objets

Tous les objets graphiques ont un *point de référence* ; en matière de coordonnées, ce point de référence est réputé être à la position (0,0).

Tous les objets graphiques ont aussi une *étendue horizontale* et une *étendue verticale*. L'étendue horizontale est représentée par une paire de nombres indiquant l'écart entre le point de référence et les bords gauche et droit – l'écart à gauche étant négatif. L'étendue verticale est représentée par une paire de nombres indiquant l'écart entre le point de référence et les bords inférieur et supérieur – l'écart vers le bas étant négatif. Dans l'image suivante, le point en coordonnée 0 est le point de référence ; l'étendue horizontale est `#'(-9 . 26)`, et l'étendue verticale est `#'(-10 . 29)`.



Voir Section “Le modèle de boîte de LilyPond” dans *Extension de LilyPond* pour plus de détails sur l'étendue des glyphes musicaux.

Les objets graphiques ont aussi des *points d'alignement*, qui ne sont pas nécessairement le même que leur point de référence. Un point d'alignement est un point relatif à un objet que LilyPond utilise pour aligner cet objet avec d'autres objets. Les points de *self-alignment* sont relatifs à l'objet lui-même ; les points de *parent-alignment* sont relatifs à l'objet parent si tant est qu'il y en ait un.

La plupart des objets graphiques ont des parents qui influencent leur positionnement. Pour ces objets, LilyPond bouge l'objet de telle sorte que son point d'alignement soit à la même position que le point d'alignement de son parent. Par exemple, `TextScript.parent-alignment-X` est la propriété d'un objet `TextScript` qui spécifie le point d'alignement sur l'axe des abscisses

(coordonnée X) relativement au parent de cet objet `TextScript` (son objet `NoteColumn` associé) sur lequel LilyPond alignera cet objet `TextScript`. Lorsque l'on place horizontalement un objet `TextScript`, LilyPond aligne le point de *self-alignment* de l'objet sur un point de *parent-alignment* relatif à l'objet `NoteColumn` qui est parent du `TextScript`. Si l'objet `NoteColumn` a plusieurs enfants, ces objets enfants peuvent avoir des points de *parent-alignment* différents. Voir Section 36.9.3 [Utilisation de `self-alignment-interface`], page 822, pour plus d'information.

La position d'un objet sur la portée est donnée par la valeur des propriétés `X-offset` et `Y-offset`. La valeur de `X-offset` indique l'écart en abscisse (coordonnée X) par rapport au point de référence de l'objet parent ; la valeur de `Y-offset` indique l'écart par rapport à la ligne médiane de la portée.[1] Les valeurs de `X-offset` et `Y-offset` peuvent être fournies arbitrairement, ou bien être calculé par des procédures spécifiques qui détermineront l'alignement par rapport à l'objet parent.

> **Note :** Nombre d'objets sont affectés par des considérations spécifiques en matière de positionnement ; jouer sur les valeurs de `X-offset` ou `Y-offset` se révélera inefficace en pareil cas, même si l'objet dispose de la `self-alignment-interface`. Fixer arbitrairement les propriétés `X-offset` ou `Y-offset` annihilera alors les effets de la propriété `self-alignment` correspondante.

Par exemple, une altération peut se repositionner verticalement grâce à son `Y-offset` ; toute modification de son `X-offset` restera par contre sans effet.

De nombreux objets peuvent s'aligner antérieurement placés à l'aide de la `self-alignment-interface` et/ou la `side-position-interface`. La liste des objets concernée est disponible aux rubriques Section "self-alignment-interface" dans *Référence des propriétés internes* et Section "side-position-interface" dans *Référence des propriétés internes*.

Les indications de repère s'alignent sur des objets de rupture – tels les barres de mesure, clefs, métriques et armures. Certaines propriétés spécifiques – dépendant de la `break-aligned-interface` – permettent de gérer le positionnement des indications de repère sur ces objets.

Voir aussi

Manuel de notation : Section 36.9.4 [Utilisation de `break-aligned-interface`], page 826.
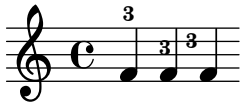
Manuel d'extension : Section "Fonctions de rappel" dans *Extension de LilyPond*.

36.9.1 Détermination directe de `X-offset` et `Y-offset`

Vous pouvez fournir, pour de nombreux objets, des valeurs numériques aux propriétés `X-offset` et `Y-offset`. Voici par exemple une note avec indication du doigté tout d'abord avec un positionnement par défaut, puis repositionnement par modification successive du `X-offset` et du `Y-offset`.

```
f'-3
f'-\tweak X-offset 0
  -\tweak Y-offset 0
  -3
f'-\tweak X-offset -1
  -\tweak Y-offset 0.5
  -3
```

[1] Pour les objets non directement associés à une portée, d'autres conventions peuvent s'appliquer. Par exemple, les valeurs d'un `Y-offset` des objets `InstrumentName` sont mesurés à partir du haut de la portée supérieure du système.



36.9.2 Utilisation de `side-position-interface`

Un objet disposant de la `side-position-interface` peut se voir accolé à son voisin de telle sorte que les bords des deux objets se touchent. Un tel objet peut se positionner au-dessus, en dessous, à droite ou à gauche de son parent. Ce parent ne saurait être stipulé ; il est déterminé par l'ordre d'apparition des éléments dans le flux des saisies. La plupart de ces objets ont pour parent la tête de note qui leur est associée.

Les valeurs des propriétés `side-axis` et `direction` détermineront l'endroit où viendra se positionner l'objet, selon les préceptes suivants :

Propriété `side-axis`	**Propriété** `direction`	**Positionnement**
0	-1	gauche
0	1	droite
1	-1	en dessous
1	1	au-dessus

Pour un `side-axis` à 0, le `X-offset` devrait engager la procédure `ly:side-position-interface::x-aligned-side`. Celle-ci renverra la valeur adéquate de `X-offset` permettant d'accoler l'objet sur la droite ou sur la gauche de son parent, selon la valeur de `direction`.

Pour un `side-axis` à 1, le `Y-offset` devrait engager la procédure `ly:side-position-interface::y-aligned-side`. Celle-ci renverra la valeur adéquate de `Y-offset` permettant d'accoler l'objet au-dessus ou en dessous de son parent, selon la valeur de `direction`.

Les valeurs 0 et 1 peuvent se remplacer par les symboles Scheme `X` et `Y` pour spécifier l'axe concerné. De façon similaire, on peut utiliser les symboles `DOWN` (`LEFT`) et `UP` (`RIGHT`) au lieu des valeurs -1 et 1 pour spécifier la direction.

36.9.3 Utilisation de `self-alignment-interface`

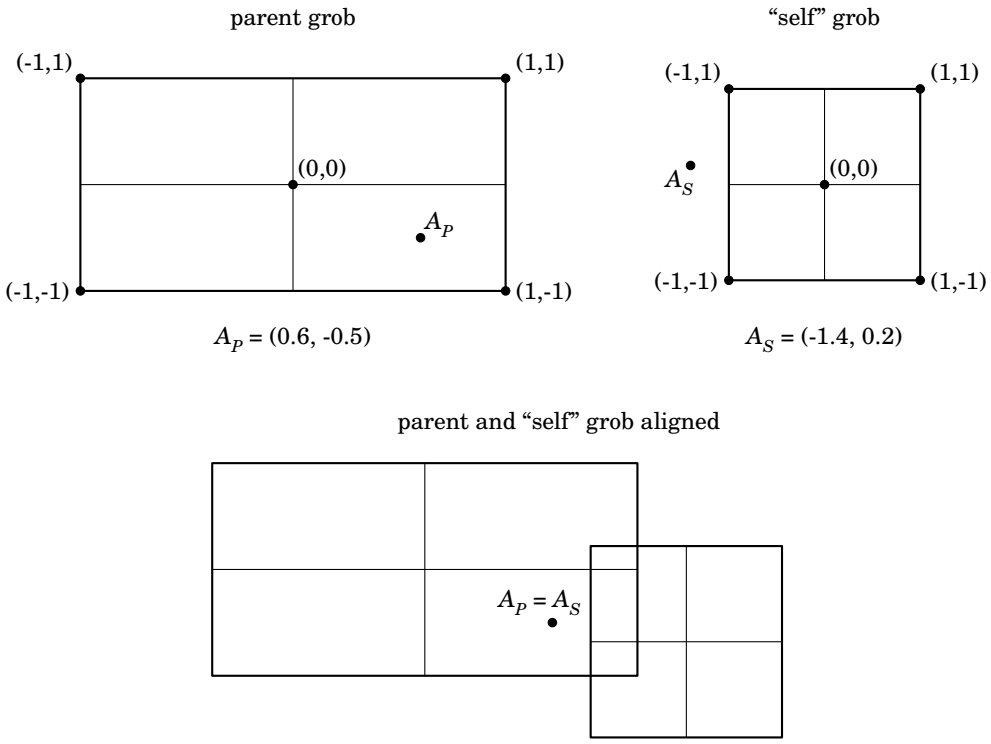
La `self-alignment-interface` définit les points d'alignement des objets. Dans LilyPond, cette interface s'applique à de nombreux objets graphiques – il sont listés dans Section "self-alignment-interface" dans *Référence des propriétés internes*.

L'interface d'alignement se base sur la dimension de la boîte englobant l'objet graphique. La coordonnée X Y du point d'alignement d'un objet est un multiple de la moitié de la largeur (hauteur) de cette boîte englobante. En d'autre termes, une unité correspond à la moitié de l'étendue X Y de l'objet.

Une coordonnée X Y à -1 représente le coin gauche (inférieur) de la boîte englobante. Une coordonnée X Y à 0 en représente le centre. Une coordonnée X Y à 1 en représente le coin droit (supérieur).

Une coordonnée peut être n'importe quel nombre réel, y compris des valeurs inférieures à -1 ou supérieures à 1.

L'image suivante montre le cas général, où à la fois le parent et l'objet « self » ont des paramètres d'alignement horizontal et vertical non nuls A_P et A_S (ces deux points sont ici choisis arbirtrairemen afin d'illustrer le concept).



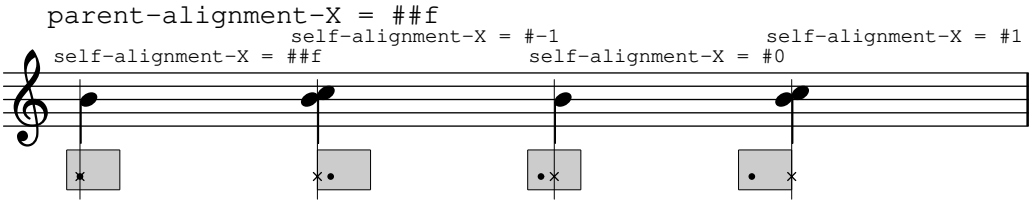
Réalignement d'objets horizontalement

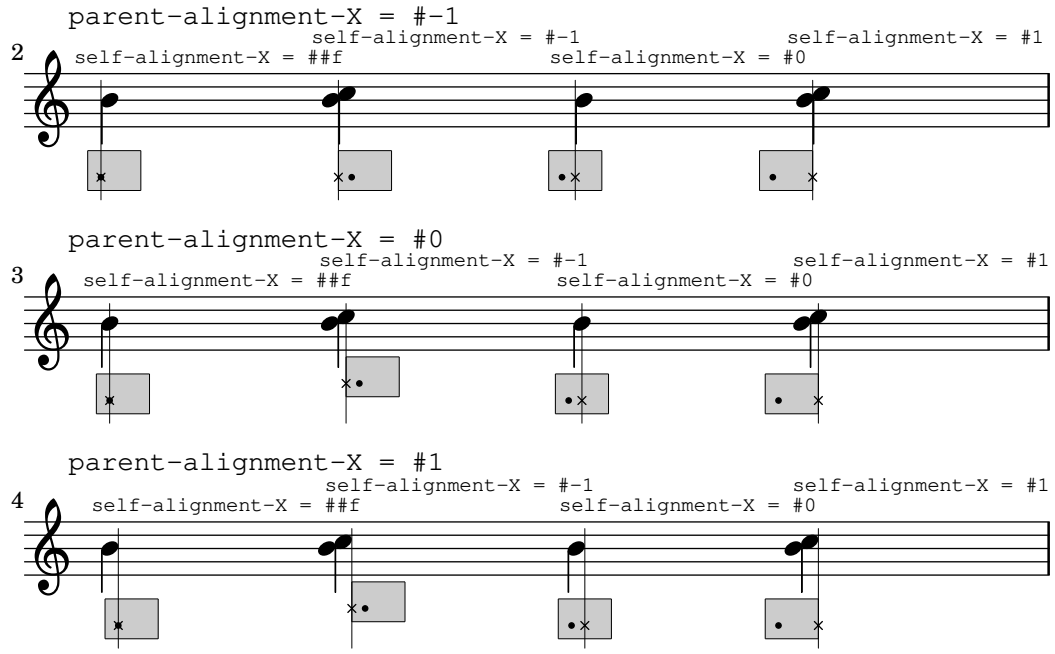
`self-alignment-X` peut contenir un nombre réel, l'unité de base étant la moitié de l'étendue horizontale de l'objet. Une valeur négative décalera l'objet vers la droite, une valeur positive vers la gauche. La valeur `0` permet de centrer l'objet sur le point de référence de son parent. Une valeur de `-1` alignera le bord gauche de l'objet sur le point de référence de son parent, et une valeur de `1` alignera le bord droit de l'objet sur le point de référence de son parent. Les valeurs symboliques `LEFT`, `CENTER` et `RIGHT` correspondent respectivement à `-1`, `0` et `1`.

Une valeur à `#f` aligne horizontalement l'objet en utilisant son point de référence. Cette valeur n'est valide que lorsque l'objet a un parent ; LilyPond émettra un erreur en l'absence de parent.

Lorsqu'un objet a un parent, son alignement horizontal est lui aussi affecté par la propriété `parent-alignment-X`.

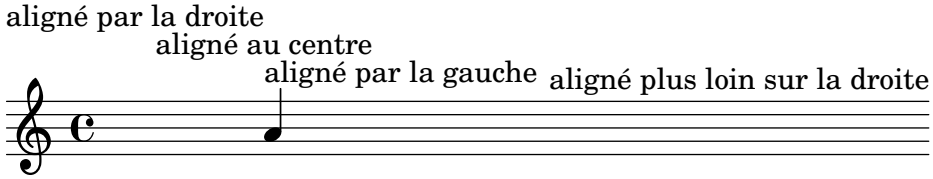
L'image suivant illustre diverses combinaisons de valeurs de `self-alignment-X` et `parent-alignment-X` pour des objets `TextScript` dont les parents sont leurs objets `NoteColumn` associés. Les objets `TextScript` sont ici des boîtes grisées dont l'étendue horizontale est `'(-1 . 3)` et l'étendue verticale `'(-1 . 2)` ; le point de référence des boîtes est représenté par un point. Le point de référence d'un objet `NoteColumn` est toujours (0,0) ; les objets `NoteHead` suspendus (têtes de note d'un accord l'une sur l'autre, côté non par défaut d'une hampe) dépassent mais ne seront pas pris en compte. La position horizontale du point-parent d'alignement est représenté par une ligne verticale, et le point d'alignement du `TextScript` est indiqué par une croix.





Aussi bien `\override` que `\tweak` peuvent venir en aide afin de modifier la valeur de `self-alignment-X`. Toutefois, le recours à la commande `\tweak` permet de traiter séparément plusieurs annotations affectées à une même note.

```
a'-\tweak self-alignment-X -1 ^"aligné par la gauche"
  -\tweak self-alignment-X 0 ^"aligné au centre"
  -\tweak self-alignment-X #RIGHT ^"aligné par la droite"
  -\tweak self-alignment-X -2.5 ^"aligné plus loin sur la droite"
```



Réalignement d'objets verticalement

L'alignement vertical suit le même principe : la propriété `Y-offset` doit alors engager la procédure `ly:self-alignment-interface::y-aligned-on-self`. Toutefois, il arrive bien souvent que d'autres mécanismes interviennent dans l'alignement vertical. La valeur de `Y-offset` n'étant que l'une des variables qui seront prises en compte, l'ajustement pour certains objets peut se révéler fastidieux. L'unité de base est relativement réduite, puisqu'elle est de la moitié de l'étendue verticale de l'objet ; le nombre à fournir en argument pourrait donc être relativement élevé. Une valeur de `-1` alignera le bord inférieur de l'objet sur le point de référence de son parent, et une valeur de `1` alignera le bord supérieur de l'objet sur le point de référence de son parent. La valeur `0` permet de centrer l'objet sur le point de référence de son parent. Les valeurs symboliques `DOWN`, `CENTER` et `UP` correspondent respectivement à `-1`, `0` et `1`.

Une valeur à `#f` aligne verticalement l'objet en utilisant son point de référence. Cette valeur n'est valide que lorsque l'objet a un parent ; LilyPond émettra un erreur en l'absence de parent.

Il est à noter que `parent-alignment-Y` ne marche pratiquement jamais : il y a trop de contraintes extérieures qui influencent la manière dont un objet sera positionné relativement à son objet parent. Prenons l'image précédente pour illustrer : la valeur du point de référence du parent (l'objet `NoteColumn`) dépend de l'objet attaché (l'objet `TextScript`) en même temps que les valeurs de certaines propriétés de décalage – comme on peut le constater, LilyPond fait tout

son possible pour replacer les objets TextScript au plus près de la portée. En raison de cette interdépendance entre l'objet et son parent, accéder ou modifier parent-alignment-Y se révèle normalement infructueux et génère souvent un message de « dépendance circulaire ».

Le moyen classique pour ajuster les distances verticales consiste à jouer sur les différentes propriétés de décalage dont LilyPond dispose, comme outside-staff-padding.

Réalignement d'objets sur les deux axes

Définir à la fois self-alignment-X et self-alignment-Y permet de réaligner un objet sur les deux axes.

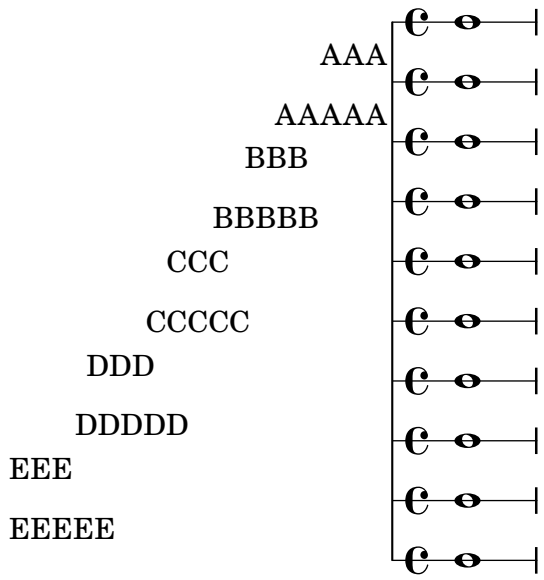
Dans l'exemple ci-dessous, nous ajustons le positionnement des noms d'instrument. L'objet InstrumentName est un objet qui n'a pas de parent : self-alignment-X calculera un alignement relatif à la marge de gauche et au début du système (soit l'indentation du système), et self-alignment-Y alignera l'objet InstrumentName par rapport à la ligne médiane de la portée considérée.

```
sn = #(define-music-function (name x y) (string? number? number?)
       #{
         \new RhythmicStaff \with {
           instrumentName = #name
           \override InstrumentName.self-alignment-X = #x
           \override InstrumentName.self-alignment-Y = #y
         } { f'1 }
       #})

<<
  \sn "AAA" #1 #3
  \sn "AAAAA" #1 #3
  \sn "BBB" #0.5 #1.5
  \sn "BBBBB" #0.5 #1.5
  \sn "CCC" #0 #0
  \sn "CCCCC" #0 #0
  \sn "DDD" #-0.5 #-1.5
  \sn "DDDDD" #-0.5 #-1.5
  \sn "EEE" #-1 #-3
  \sn "EEEEE" #-1 #-3
>>

\layout {
  indent = 5\cm

  \context {
    \RhythmicStaff
    \override VerticalAxisGroup.staff-staff-spacing =
      #'((minimum-distance . 0)
         (basic-distance . 0)
         (padding . 2.5))
  }
}
```



36.9.4 Utilisation de `break-aligned-interface`

Indications de repère, marques textuelles, numéros de mesure et plus généralement tous les objets prenant en charge la Section “break-alignable-interface” dans *Référence des propriétés internes* – que l'on pourrait regrouper sous l'appellation de « jalons » – sont susceptibles de se positionner sur des objets de notation tels que barre de mesure, armure, métrique et plus communément tout objet prenant en charge la Section “break-aligned-interface” dans *Référence des propriétés internes*. Pour être plus précis, ces derniers disposent d'une propriété `break-align-symbol`, qui est un symbole Scheme. Un jalon, donc un objet disposant de `break-alignable-interface`, peut définir son point d'ancrage aux éléments disposant de `break-aligned-interface`, ce au moyen d'une liste de symboles qui correspondent aux valeurs de `break-align-symbols` des éléments sur lesquels il peut s'aligner. Une liste exhaustive des symboles est disponible au chapitre Section “break-alignment-interface” dans *Référence des propriétés internes*.

Chaque objet possède son propre point de référence par défaut, sur lequel viendront s'aligner les jalons :
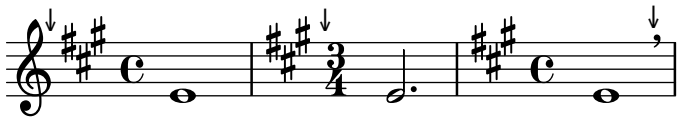
```
\override Score.TextMark.self-alignment-X = #CENTER

% Le repère textuel s'alignera sur la droite de la clef
\override Score.TextMark.break-align-symbols = #'(clef)
\clef treble
\key a \major
\textMark "↓"
e'1

% Le repère textuel s'alignera sur la gauche de la métrique
\override Score.TextMark.break-align-symbols = #'(time-signature)
\clef treble
\key a \major
\time 3/4
\textMark "↓"
e'2.

% Le repère textuel sera centré au-dessus de la respiration
\override Score.TextMark.break-align-symbols = #'(breathing-sign)
\clef treble
```

```
\key a \major
\time 4/4
e'1
\breathe
\textEndMark "↓"
```

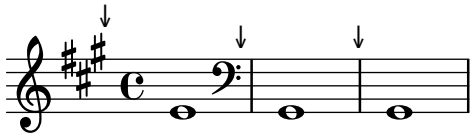


Les différents objets sur lesquels l'alignement pourrait intervenir seront regroupés dans une liste. Si l'un des objets est invisible à l'endroit voulu, en raison d'un réglage de `break-visibility` ou bien par forçage de la visibilité des armures et clefs, le repère ou le numéro de mesure viendra s'aligner sur le premier élément de cette liste qui soit visible. Dans le cas où aucun objet de la liste n'est visible, l'alignement se fera sur la barre de mesure ou, dans le cas où la barre de mesure est invisible, à l'endroit même où la barre prendrait place.

```
\override Score.TextMark.self-alignment-X = #CENTER
% Le repère textuel s'alignera sur la droite de l'armure
\override Score.TextMark.break-align-symbols = #'(key-signature clef)
\clef treble
\key a \major
\textMark "↓"
e'1

% Le repère textuel s'alignera sur la droite de la clef
\set Staff.explicitKeySignatureVisibility = #all-invisible
\override Score.TextMark.break-align-symbols = #'(key-signature clef)
\clef bass
\key a \major
\textMark "↓"
gis,1

% Le repère textuel sera centré au-dessus de la barre de mesure
\set Staff.explicitKeySignatureVisibility = #all-invisible
\set Staff.explicitClefVisibility = #all-invisible
\override Score.TextMark.break-align-symbols = #'(key-signature clef)
\clef treble
\key a \major
\textMark "↓"
e'1
```
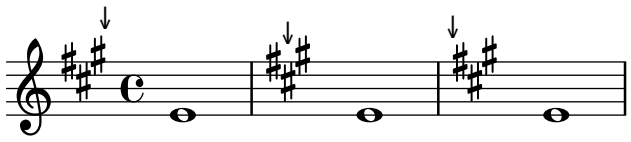


L'alignement d'un jalon sur un objet de notation peut se modifier, comme l'illustre l'exemple suivant. Toutefois, si la partition comporte plusieurs portées, ce réglage devra apparaître dans chacune des portées.

```
\override Score.TextMark.self-alignment-X = #CENTER
% Le repère textuel s'alignera sur la droite de l'armure
\override Score.TextMark.break-align-symbols = #'(key-signature)
\clef treble
\key a \major
\time 4/4
\textMark "↓"
e'1

% Le repère textuel sera centré au-dessus de l'armure
\once \override
  Score.KeySignature.break-align-anchor-alignment = #CENTER
\textMark "↓"
\key a \major
e'1

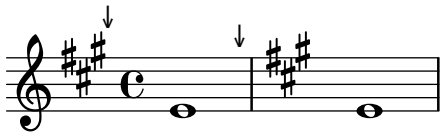
% Le repère textuel s'alignera sur la gauche de l'armure
\once \override
  Score.KeySignature.break-align-anchor-alignment = #LEFT
\key a \major
\textMark "↓"
e'1
```



Le bord gauche d'un jalon peut se décaler arbitrairement sur la gauche ou sur la droite. La valeur est exprimée en espaces de portée.

```
\override Score.TextMark.self-alignment-X = #CENTER
% Le repère textuel s'alignera sur la gauche de l'armure
% puis décalé de 3,5 espaces de portée vers la droite
\override Score.TextMark.break-align-symbols = #'(key-signature)
\once \override Score.KeySignature.break-align-anchor = 3.5
\key a \major
\textMark "↓"
e'1

% Le repère textuel s'alignera sur la gauche de l'armure
% puis sera décalé de 2 espaces de portée vers la gauche
\once \override Score.KeySignature.break-align-anchor = -2
\key a \major
\textMark "↓"
e'1
```



Voir aussi

Référence des propriétés internes : Section “break-alignable-interface” dans *Référence des propriétés internes*, Section “break-alignment-interface” dans *Référence des propriétés internes*.

36.10 Modification des stencils
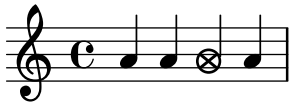
Tout objet de rendu dispose d'une propriété `stencil` attachée à la `grob-interface`. En règle générale, cette propriété référence par défaut une fonction spécifique à l'objet et taillée sur mesure pour fournir le symbole qui va le représenter dans l'output. Par exemple, le réglage standard de la propriété `stencil` de l'objet `MultiMeasureRest` est `ly:multi-measure-rest::print`.

Le symbole standard d'un objet quel qu'il soit peut être remplacé à partir du moment où la propriété `stencil` référence une procédure différente et écrite à cet effet. Ceci requiert une bonne maîtrise du fonctionnement interne de LilyPond, mais est grandement facilité dans bien des cas et permet d'obtenir le résultat escompté.

En effet, rien ne nous interdit, à partir de la propriété `stencil`, d'appeler la procédure qui génère du texte, `ly:text-interface::print` en l'occurrence, et d'adjoindre à l'objet une propriété `text` qui contiendra, sous forme de *markup*, le symbole à dessein. Grâce à l'extrême flexibilité des *markups*, vous pourrez parvenir à bien des choses – voir à ce sujet Section 8.2.4 [Éléments graphiques dans du texte formaté], page 342.

C'est la technique employée ici, où l'une des têtes de note est remplacée par une croix inscrite dans un cercle :

```
XinO = {
  \once \override NoteHead.stencil = #ly:text-interface::print
  \once \override NoteHead.text = \markup {
    \combine
      \halign #-0.7 \draw-circle #0.85 #0.2 ##f
      \musicglyph "noteheads.s2cross"
  }
}
\relative {
  a' a \XinO a a
}
```



Tous les glyphes `Feta` de la fonte Emmentaler sont accessibles à l'aide de la commande de *markup* `\musicglyph` – voir Section B.8 [La fonte Emmentaler], page 935.

L'insertion de fichier `EPS` ou d'instructions Postscript sont accessibles par les commandes de *markup* `\epsfile` et `\postscript` respectivement – voir l'annexe Section A.1.3 [Graphical markup], page 868.

Voir aussi

Manuel de notation : Section A.1 [Commandes pour *markup*], page 837, Section 8.2.4 [Éléments graphiques dans du texte formaté], page 342, Section A.1.3 [Graphical markup], page 868, Section B.8 [La fonte Emmentaler], page 935, Section 8.2 [Mise en forme du texte], page 329.

36.11 Modification de l'allure des éléments

36.11.1 Modification des liaisons

Les liaisons, qu'elles soient de prolongation (`Tie`), d'articulation (`Slur`), de phrasé (`PhrasingSlur`), de laisser-vibrer (`LaissezVibrerTie`) ou de reprise (`RepeatTie`), sont dessinées sous la forme de courbes de Bézier de degré trois. Lorsque l'aspect de la liaison automatiquement calculé n'est pas satisfaisant, il peut être modifié manuellement de deux manières différentes :

1. en spécifiant l'ajustement qui doit être apporté aux points de contrôle de la courbe calculée automatiquement, ou
2. en fournissant explicitement les quatre points de contrôle qui permettront de définir cette courbe.

Ces deux méthodes sont expliquées ci-dessous. La première convient mieux dans le cas d'une légère adaptation de la courbe ; la seconde sera plus efficace lorsqu'il s'agira de créer une courbe sur une seule et unique note.

Courbes de Bézier cubiques

Quatre points définissent une courbe de Bézier cubique. Les premier et quatrième points sont les points de départ et d'arrivée de la courbe ; les deux autres points de contrôle – P1 et P2 – en détermineront l'allure. On peut trouver sur le *web* des animations à propos du dessin de telles courbes ; la description qui suit en propose un aperçu. La courbe se trace en partant du point P0, en se dirigeant vers P1 et en arrivant au point P3 selon la direction P2-P3. La courbe est à l'intérieur du quadrilatère défini par les points de contrôle.
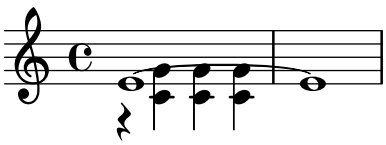


Tout déplacement (translation, rotation, échelonnement) des points de contrôle sera répercuté sur le dessin de la courbe.

Spécification de l'ajustement des points de contrôle

Voici par exemple une liaison de prolongation dont l'allure n'est pas des plus heureuses, même en optant pour un `\tieDown`.

```
<<
  { e'1~ 1 }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```



L'ajustement des points de contrôle de cette liaison de tenue à l'aide de `\shape` permet d'éviter les collisions.

L'instruction `\shape` obéit à la syntaxe

`[-]\shape` *`déplacements élément`*

Ceci aura pour effet de repositionner les points de contrôle de *élément* des différents montants fournis par *déplacements*. L'argument *déplacements* est constitué d'une liste de paires de nombres ou bien d'une liste de telles listes. Chacun des membres de l'une des paires indique
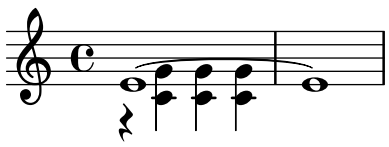
l'ajustement de la coordonnée d'un point de contrôle. Lorsque *élément* est textuel, il en résulte une dérogation particulière appliquée au type d'objet considéré, alors que dans le cas d'une expression musicale sera appliqué un affinage approprié.

En d'autres termes, la fonction `\shape` se comporte soit comme un `\once \override`, soit comme un `\tweak` selon que l'argument *élément* est un nom d'objet – tel « Slur » – ou une expression musicale tel un « (». L'argument *déplacements* spécifie les ajustements à apporter aux quatre points de contrôle, sous la forme d'une liste de paires `(dx . dy)` dont les valeurs sont exprimées en espace de portée ; on utilisera une liste de listes de paires dans le cas où la courbe comporte plusieurs segments.

La fonction sera précédée d'un tiret si et seulement si elle doit s'appliquer sous forme de `\tweak`.
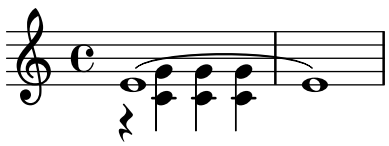
Pour l'exemple qui nous occupe, l'adaptation sous forme dérogatoire – recours à `\once\override` – de la fonction `\shape`, nous pouvons remonter la liaison d'un demi espace de portée :

```
<<
  {
    \shape #'((0 . 0.5) (0 . 0.5) (0 . 0.5) (0 . 0.5)) Tie
    e'1~ 1
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```
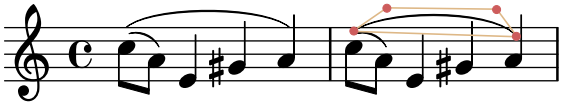


La liaison est maintenant mieux positionnée ; mais sa partie centrale pourrait être un peu plus relevée, en procédant comme ci-dessous, cette fois en utilisant la formulation d'affinage – la forme `\tweak` :

```
<<
  {
    e'1-\shape #'((0 . 0.5) (0 . 1) (0 . 1) (0 . 0.5)) ~ e'
  }
\\
  \relative { r4 <g' c,> <g c,> <g c,> }
>>
```



LilyPond dispose de la fonction `\vshape` qui apporte une assistance dans cette procédure d'affinage. Son nom signifie *visual shape*. Elle agit exactement comme la fonction `\shape`, à ceci près que seront aussi affichés les points de contrôle et le polygone.
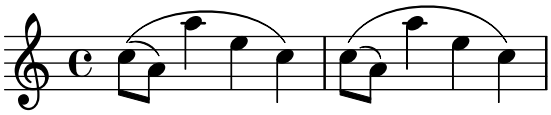
```
\relative {
  c''8\( a) e4 gis a\)
  \vshape #'((0 . -0.3) (0.5 . -0.2)
             (0.5 . -0.3) (0 . -0.7)) PhrasingSlur
  c8\( a) e4 gis a\)
}
```



Il est conseillé de commencer avec \vshape et d'ajuster jusqu'à obtenir un galbe satisfaisant, puis de simplement supprimer le « v » de \vshape.
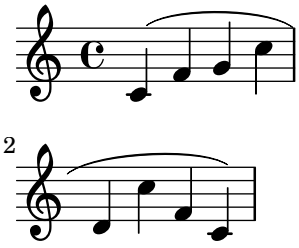
Il est possible de gérer distinctement le galbe de deux courbes débutant au même instant musical :

```
\relative {
  c''8\( a) a'4 e c\)
  \shape #'((0.7 . -0.4) (0.5 . -0.4)
            (0.3 . -0.3) (0 . -0.2)) Slur
  \shape #'((0 . 0) (0 . 0.5)
            (0 . 0.5) (0 . 0)) PhrasingSlur
  c8\( a) a'4 e c\)
}
```
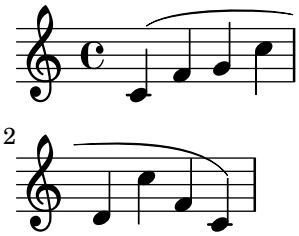


La fonction \shape permet aussi d'adapter les points de contrôle d'une courbe qui se prolonge après un saut de ligne. Chaque portion de la courbe peut se voir appliquer sa propre liste d'ajustements. Lorsque l'un des segments ne nécessite pas de retouche, il suffit de lui fournir une liste vide. Dans l'exemple suivant, le saut de ligne laisse à croire qu'il y a non pas une seule mais deux liaisons :

```
\relative {
  c'4( f g c
  \break
  d,4 c' f, c)
}
```
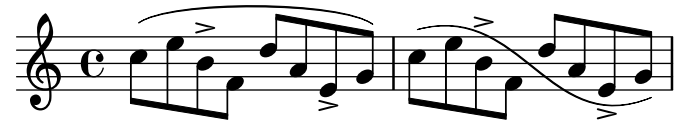


Regalber les deux moitiés de la liaison rend plus évident le fait qu'elle s'étend par delà le saut de ligne :

```
% () peut faire office de raccourci à ((0 . 0) (0 . 0) (0 . 0) (0 . 0))
% lorsque l'un des segments ne nécéssite pas de modification
\relative c' {
  \shape #'(
             (( 0 . 0) (0 . 0) (0 . 0) (0 . 1))
             ((0.5 . 1.5) (1 . 0) (0 . 0) (0 . -1.5))
           ) Slur
  c4( f g c
  \break
  d,4 c' f, c)
}
```



La présence d'une courbe en esse requiert obligatoirement d'ajuster manuellement les points de contrôle – LilyPond n'optera jamais automatiquement pour un tel galbe.

```
\relative c'' {
  c8( e b-> f d' a e-> g)
  \shape #'((0 . -1) (5.5 . -0.5) (-5.5 . -10.5) (0 . -5.5))
         PhrasingSlur
  c8\( e b-> f d' a e-> g\)
}
```



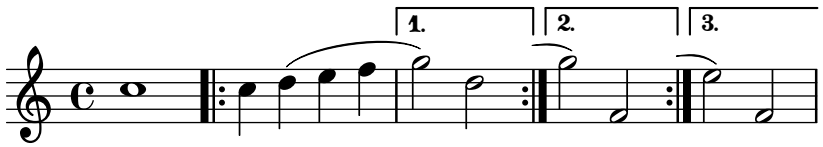
Déclaration explicite des points de contrôle

Les coordonnées des points de contrôle sont données en unités d'espace de portée. L'abscisse est relative au point de référence de la note de départ de la liaison ; l'ordonnée est relative à la ligne médiane de la portée. Les différentes coordonnées sont entrées sous la forme d'une liste de quatre paires de nombres décimaux (ou nombres réels). L'une des manières de procéder consiste à tout d'abord estimer les coordonnées des deux extrémités puis, par tâtonnement, déterminer les deux points intermédiaires. Gardez à l'esprit que ces valeurs pourront devoir être revues si la musique ou sa mise en forme sont modifiées.

L'une des situation où spécifier explicitement les points de contrôle se révèle être tout à fait appropriée est lorsqu'ils se réfèrent à une seule et unique note. L'exemple suivant illustre l'un des moyens d'indiquer une liaison qui se prolonge sur les alternatives d'une répétition.

```
\relative {
  c''1
  \repeat volta 3 { c4 d( e f }
  \alternative {
    \volta 1 { g2) d }
    \volta 2 {
      g2
        % create a slur and move it to a new position
```

```
        % the <> is just an empty chord to carry the slur termination
        -\tweak control-points
                  #'((-2 . 3.8) (-1 . 3.9) (0 . 4) (1 . 3.4)) ( <> )
      f,
    }
    \volta 3 {
      e'2
        % create a slur and move it to a new position
        -\tweak control-points
                  #'((-2 . 3) (-1 . 3.1) (0 . 3.2) (1 . 2.4)) ( <> )
      f,
    }
  }
}
```



Problèmes connus et avertissements

Lorsque plusieurs liaisons, quelle qu'en soit la nature, commencent au même moment, jouer sur la propriété `control-points` est impossible, et la commande `\tweak` inefficace. Vous pouvez néanmoins influer sur la propriété `tie-configuration` de l'objet `TieColumn` pour déterminer la ligne de départ et l'orientation.

Voir aussi

Référence des propriétés internes : Section "TieColumn" dans *Référence des propriétés internes*.

Annexes

Annexe A Commandes pour *markups*

A.1 Commandes pour *markup*

Les commandes suivantes peuvent être utilisées dans un bloc `\markup { }`.

The following commands can all be used inside `\markup { }`.

A.1.1 Font markup

`\abs-fontsize` *size* (number) *arg* (markup)

Use *size* as the absolute font size (in points) to display *arg*.

This function adjusts the `baseline-skip` and `word-space` properties accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size text font size 16 text font size 12

Used properties:

- `baseline-skip` (3)
- `word-space` (0.6)

`\bold` *arg* (markup)

Print *arg* with a bold face.

```
\markup {
  default
  \hspace #2
  \bold bold
}
```

default **bold**

The code `\markup \bold ...` is a shorthand for `\markup \override #'(font-series . bold) ...` – using the more verbose form, it is possible to obtain nuances such as semi-bold, if the text font has such variants. Refer to the documentation for the `font-series` properties (Section “User backend properties” dans *Référence des propriétés internes*).

`\box` *arg* (markup)

Draw a box around *arg*.

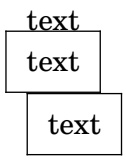
This function looks at the `thickness`, `box-padding`, and `font-size` properties to determine the line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box \line { V. S. }
}
```

V. S.

Note that the box does not horizontally displace its argument. Use markup commands like `\left-align` or `\table` to make LilyPond realign it.

```
\markup {
  \override #'(box-padding . 1.5)
  \column {
    "text"
    \box "text"
    \left-align \box "text"
  }
}
```



Used properties:

- `box-padding` (`0.2`)
- `font-size` (`0`)
- `thickness` (`1`)

`\caps` *arg* (markup)

Print *arg* in (fake) small caps.

This function is a copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

Use `\fontCaps` for real small caps (if the font provides it).

`\dynamic` *arg* (markup)

Print *arg* using the (music) font for dynamics.

This font only contains letters **f**, **m**, **n**, **p**, **r**, **s**, and **z**. When producing phrases like 'più **f**', the normal words (like 'più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

sfzp

`\figured-bass` *arg* (markup)

Set *arg* as small numbers for figured bass.

Specially slashed digits can be achieved with a trailing backslash (for numbers 6, 7, and 9) or a trailing plus (for numbers 2, 4, and 5).[1]

[1] Internally, this works by activating the 'dlig' OpenType feature of the Emmentaler font.

The use of a backslash is in analogy to `\figuremode` (voir Section 15.3.2 [Saisie de la basse chiffrée], page 541). Note that to get a backslash character in markup it must be escaped by doubling it. Additionally, it must be put into double quotes.

```
\markup {
  \figured-bass {
    2 3 4+ 7 "9\\"
  }
}
```



`\finger` *arg* (markup)
: Set *arg* as small numbers for fingering instructions.

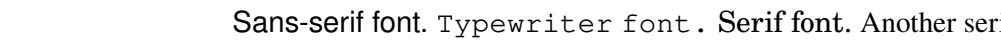
```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```



`\font-select` *defs* (pair) *arg* (markup)
: Output the markup *arg* using fonts specified in *defs*.

defs is either a single pair (*family* . *font-name*) or a non-empty list of such pairs. Here, *family* is usually `serif`, `sans` or `typewriter`; *font-name* is a string. Font families not mentioned in the call to `\font-select` remain unchanged.

```
\markup {
  \font-select #'((sans . "Liberation Sans")
                  (typewriter . "Liberation Mono"))
  {
    \sans { Sans-serif font. }
    \typewriter { Typewriter font. }
    Serif font.
    \font-select #'(serif . "Liberation Serif") {
      Another serif font.
    }
  }
}
```



`\fontCaps` *arg* (markup)
: Print *arg* in small caps.

This command sets the `font-variant` property to `small-caps`.

Unlike `\smallCaps`, which merely uses capital letters at a smaller font size, this uses the actual variant of the font for small caps. (As a consequence, if you configure a custom text font, this command has no effect if that font does not have a small caps variant.)

```
\markup \fontCaps "Small caps"
```



`\fontsize` *increment* (number) *arg* (markup)
: Increase current font size by *increment* to print *arg*.

This function adjusts the `baseline-skip` and `word-space` properties accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5 smaller
}
```

default **smaller**

Used properties:

- `baseline-skip` (2)
- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to value 2 to print *arg*.

```
\markup {
  default
  \hspace #2
  \huge huge
}
```

default huge

`\italic` *arg* (markup)
Print *arg* in italics.

This command sets the `font-shape` property to `italic`.

```
\markup {
  default
  \hspace #2
  \italic italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to value 1 to print *arg*.

```
\markup {
  default
  \hspace #2
  \large large
}
```

default large

`\larger` *arg* (markup)
Increase current font size by 1 to print *arg*.

This function adjusts the `baseline-skip` and `word-space` properties accordingly.

```
\markup {
  default
  \hspace #2
  \larger larger
}
```

default larger

\magnify *sz* (number) *arg* (markup)
: Magnify current font by factor *sz* to print *arg*.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

Note that magnification only works if a font name is explicitly selected. Use \fontsize otherwise.

\normal-size-sub *arg* (markup)
: Set *arg* in subscript with a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default subscript in standard size

Used properties:

- font-size (0)

\normal-size-super *arg* (markup)
: Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default superscript in standard size

Used properties:

- font-size (0)

\normal-text *arg* (markup)
: Print *arg* with default text font.

  This resets all font-related properties (except the size), no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \fontCaps {
    huge bold sans caps
    \hspace #2
    \normal-text {
      huge normal
    }
    \hspace #2
    as before
  }
}
```

HUGE BOLD SANS CAPS huge normal **AS BEFORE**

\normal-weight *arg* (markup)
: Switch to normal weight (in contrast to bold) to print *arg*.

  This command sets the `font-series` property to `normal`.

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \normal-weight {
      normal font series
    }
    \hspace #2
    bold again
  }
}
```

some bold text normal font series **bold again**

\normalsize *arg* (markup)
: Set font size to default (i.e., to value 0) to print *arg*.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small normal size teeny again

\number *arg* (markup)
: Print *arg* using the (music) font for numbers.

  This font also contains symbols for figured bass, some punctuation, spaces of various widths, some letters and text variants of accidentals, the common time, and the cut time symbol. Use `\dynamic` to access the (very small number of) letters. For

accidentals you might use \number in combination with Unicode characters to access the text representation forms of accidental glyphs, as the following table shows.

Unicode value	Unicode character
U+266D	♭
U+266E	♮
U+266F	♯
U+1D12A	𝄪
U+1D12B	𝄫
U+1D134	𝄴
U+1D135	𝄵

Examples:

```
\number ♭ → ♭
\number { \char ##x266F } → ♯
```

To get accidentals protected against overrides of font-name it is preferable to use \text-doubleflat, \text-flat, \text-natural, \text-sharp, \text-doublesharp, or the general \text-accidental for the text variants of accidentals. For the time signature symbols, use \text-common-time and \text-cut-time.

The appearance of digits in the Emmentaler font can be controlled with four OpenType features: 'tnum', 'cv47', 'ss01', and 'kern', which can be arbitrarily combined.

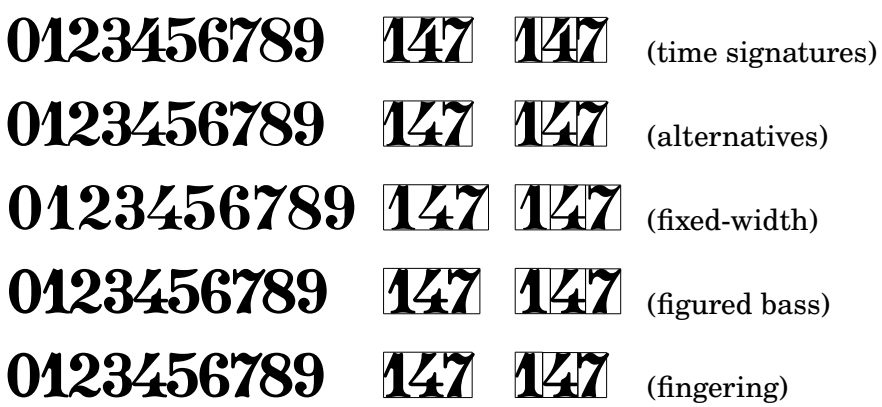
tnum — If off (which is the default), glyphs 'zero' to 'nine' have no left and right side bearings. If on, the glyphs all have the same advance width by making the bearings non-zero.

cv47 — If on, glyphs 'four' and 'seven' have shorter vertical strokes. Default is off.

ss01 — If on, glyphs 'zero' to 'nine' have a fatter design, making them more readable at small sizes. Default is off.

kern — If on (which is the default), provide pairwise kerning between (most) glyphs.

```
\markuplist
  \number
  \fontsize #4.5
  \override #'((padding . 2)
               (baseline-skip . 4)
               (box-padding . 0)
               (thickness . 0.1))
  \table #'(-1 -1 -1 -1) {
      0123456789 \box 147 \concat { \box 1 \box 4 \box 7 }
    \normal-text \normalsize "(time signatures)"
    \override #'(font-features .("cv47")) {
      0123456789 \box 147 \concat { \box 1 \box 4 \box 7 } }
    \normal-text \normalsize "(alternatives)"
    \override #'(font-features .("tnum" "cv47" "-kern")) {
      0123456789 \box 147 \concat { \box 1 \box 4 \box 7 } }
    \normal-text \normalsize "(fixed-width)"
    \override #'(font-features . ("tnum" "cv47" "ss01")) {
      0123456789 \box 147 \concat { \box 1 \box 4 \box 7 } }
    \normal-text \normalsize "(figured bass)"
    \override #'(font-features . ("cv47" "ss01")) {
      0123456789 \box 147 \concat { \box 1 \box 4 \box 7 } }
    \normal-text \normalsize "(fingering)"
  }
```



See also the markup commands `\figured-bass` and `\finger`, which set the font features accordingly.

`\overtie` *arg* (markup)

Print a tie over *arg*.

```
\markup \line {
  \overtie "overtied"
  \override #'((offset . 5) (thickness . 1))
  \overtie "overtied"
  \override #'((offset . 1) (thickness . 5))
  \overtie "overtied"
}
```



Used properties:

- `shorten-pair` (`(0 . 0)`)

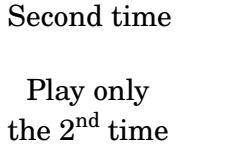
- `height-limit` (0.7)
- `direction` (1)
- `offset` (2)
- `thickness` (1)

`\replace` *replacements* (list) *arg* (markup)

Use *replacements* to replace strings in *arg*.

Each (*key* . *value*) pair of the *replacements* alist specifies what should be replaced; *key* gets replaced by *value*. Note the quasi-quoting syntax with a backquote in the second example.

```
\markup \replace #'(("2nd" . "Second"))
  "2nd time"
\markup \replace
  #`(("2nd" . ,#{ \markup \concat { 2 \super nd } #}))
  \center-column {
    \line { Play only }
    \line { the 2nd time }
  }
```



Used properties:

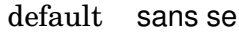
- `replacement-alist`

`\sans` *arg* (markup)

Print *arg* with a sans-serif font.

This command sets the `font-family` property to `sans`.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```



`\serif` *arg* (markup)

Print *arg* with a serif font.

This command sets the `font-family` property to `serif`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \serif {
      text in serif font
    }
    \hspace #2
    return to sans
```

```
  }
}
```

sans serif, bold text in serif font return to sans

`\simple` *str* (string)
: Print string *str*.

  `\markup \simple "x"` is equivalent to `\markup "x"`. This command was previously used internally, but no longer is, and is being kept for backward compatibility only.

`\small` *arg* (markup)
: Set font size to value -1 to print *arg*.

```
\markup {
  default
  \hspace #2
  \small small
}
```

default small

`\smallCaps` *arg* (markup)
: Print *arg* in (fake) small caps.

  Unlike `\fontCaps`, which uses the actual small caps variant of the current font, this fakes small caps by using capital letters at a smaller font size. It can thus be used for fonts that don't have a small caps variant.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\smaller` *arg* (markup)
: Decrease current font size by 1 to print *arg*.

  This function adjusts the `baseline-skip` and `word-space` properties accordingly.

```
\markup {
  \fontsize #3.5 {
    large text
    \hspace #2
    \smaller { smaller text }
    \hspace #2
    large text
  }
}
```

large text smaller text large text

`\sub` *arg* (markup)
: Set *arg* in subscript.

```
\markup { \concat { H \sub 2 O } }
```

H_2O

See also \super.

Used properties:

- font-size (0)

\super *arg* (markup)
Set *arg* in superscript.

```
\markup { E = \concat { mc \super 2 } }
```

$E = mc^2$

See also \sub.

Used properties:

- font-size (0)

\teeny *arg* (markup)
Set font size to value -3 to print *arg*.

```
\markup {
  default
  \hspace #2
  \teeny teeny
}
```

default teeny

\tie *arg* (markup)
Add a horizontal bow at the bottom or top of *arg*.

This function uses make-tie-stencil to create the bow; it looks at the thickness and offset properties to determine the line thickness and vertical offset, respectively. The added bow fits the extent of *arg*; use the shorten-pair property to modify this. The direction property may be set explicitly using override or direction modifiers, or implicitly by using voiceOne, etc.

```
\markup {
  \override #'(direction . 1)
  \tie "above"
  \override #'(direction . -1)
  \tie "below"
}
```

above below

See also \undertie and \overtie, which are shorthands for this function.

Used properties:

- shorten-pair ((0 . 0))
- height-limit (0.7)
- direction (1)
- offset (2)
- thickness (1)

\tiny *arg* (markup)
Set font size to value -2 to print *arg*.

```
\markup {
  default
  \hspace #2
  \tiny tiny
}
```

default tiny

\typewriter *arg* (markup)
Print *arg* in typewriter.

This command sets the `font-family` property to `typewriter`, also switching off the 'liga' OpenType feature to disable ligatures like 'ff' or 'fi'.

```
\markup {
  "default fi ff"
  \hspace #2
  \typewriter "typewriter fi ff"
}
```

default fi ff typewriter fi ff

\underline *arg* (markup)
Underline *arg*.

This function looks at the property `thickness` to determine the line thickness, at `offset` to determine the line's vertical offset from *arg*, and at `underline-skip` to determine the distance of additional lines from the others.

The `underline-shift` property is used to make subsequent calls work correctly. Overriding it makes little sense since it would end up adding the provided value to the one of `offset`.

```
\markup \justify-line {
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(thickness . 1)
  \underline "underlined"
  \override #'(offset . 1)
  \override #'(thickness . 5)
  \underline "underlined"
  \override #'(offset . 5)
  \override #'(underline-skip . 4)
  \underline \underline \underline "underlined thrice"
}
```

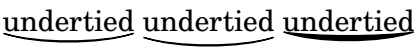
underlined underlined underlined underlined thrice

Used properties:

- `underline-skip` (2)
- `underline-shift` (0)
- `offset` (2)
- `thickness` (1)

\undertie *arg* (markup)
Print a tie under *arg*.

```
\markup \line {
  \undertie "undertied"
  \override #'((offset . 5) (thickness . 1))
  \undertie "undertied"
  \override #'((offset . 1) (thickness . 5))
  \undertie "undertied"
}
```



Used properties:

- `shorten-pair` (`(0 . 0)`)
- `height-limit` (`0.7`)
- `direction` (`1`)
- `offset` (`2`)
- `thickness` (`1`)

`\upright` *arg* (markup)

Print *arg* upright.

This command is the opposite of `\italic`; it sets the `font-shape` property to `upright`.

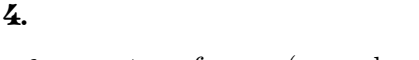
```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```



`\volta-number` *arg* (markup)

Set *arg* in a font appropriate for volta numbers.

```
\markup \volta-number "4."
```



`\with-string-transformer` *transformer* (procedure) *arg* (markup)

Apply string transformer function *transformer* to *arg*.

Whenever a string is interpreted inside *arg*, function *transformer* is called first, and its result is then interpreted. The arguments passed to *transformer* are the output definition, the property alist chain, and the markup *arg*. See Section “New markup command definition” dans *Extension de LilyPond* about the two first arguments.

```
\markup \with-string-transformer
  #(lambda (layout props str)
     (string-upcase str))
  \concat { "abc" \larger "def" }
```



A.1.2 Markup for text alignment

`\abs-hspace` *amount* (number)
: Create an invisible object taking up absolute horizontal space of *amount* points.

```
\markup {
  one
  \abs-hspace #20
  two
  \abs-hspace #40
  three
}
```

one two three

: See also `\hspace`.

`\abs-vspace` *amount* (number)
: Create an invisible object taking up absolute vertical space of *amount* points.

```
\markup {
    \center-column {
    one
    \abs-vspace #20
    two
    \abs-vspace #40
    three
  }
}
```

one

two

three

: See also `\vspace`.

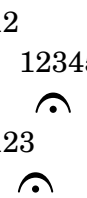
`\align-on-other` *axis* (non-negative, exact integer) *other-dir* (boolean-or-number) *other* (markup) *self-dir* (boolean-or-number) *self* (markup)
: Align markup *self* on markup *other* along *axis*.

  This function uses *self-dir* and *other-dir* for mutual alignment of *self* and *other*, respectively, translating *self* as requested relative to its surroundings. *other* is not printed.

  If *self-dir* or *other-dir* is `#f`, use the reference point of *self* or *other*, respectively.

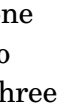
```
\markup \column {
  12
  \align-on-other #X #RIGHT 12
                     #LEFT 12345
  \align-on-other #X #RIGHT 123
                     #LEFT \fermata
  123
  \align-on-other #X #RIGHT 123
```

```
                                ##f \fermata
}
```



`\center-align` *arg* (markup)
: Align *arg* to its X center.

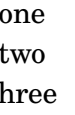
```
\markup {
  \column {
    one
    \center-align two
    three
  }
}
```



`\center-column` *args* (markup list)
: Put *args* into a centered column.

  See also `\column`.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```



  Used properties:

  - `baseline-skip`

`\column` *args* (markup list)
: Stack the markups in *args* vertically.

  The property `baseline-skip` determines the space between markups in *args* (to be more precise, the space between the baselines of the markups).

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

The baseline of the output of `\column` is the baseline of its first line.

Used properties:

- `baseline-skip`

`\combine` *arg1* (markup) *arg2* (markup)

Print *arg1*, then print *arg2* on top of it.

Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; for this purpose use `\overlay` instead.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```



`\concat` *args* (markup list)

Concatenate *args* in a horizontal line, without spaces in between.

Strings are concatenated on the input level, allowing ligatures. For example, `\concat { "f" "i" }` is equivalent to `"fi"`.

```
\markup {
  \concat {
    one two three
  }
}
```

onetwothree

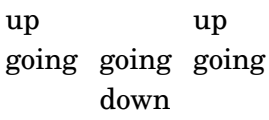
`\dir-column` *args* (markup list)

Make a column of *args*.

Depending on the setting of the `direction` layout property, the arguments are stacked upwards or downwards.

```
\markup {
  \override #`(direction . ,UP)
  \dir-column {
    going up
  }
  \hspace #1
  \dir-column {
    going down
  }
  \hspace #1
  \override #'(direction . 1)
  \dir-column {
    going up
  }
```

```
}
```



The baseline of the output of `\dir-column` is the baseline of its first line.

Used properties:

- `baseline-skip`
- `direction`

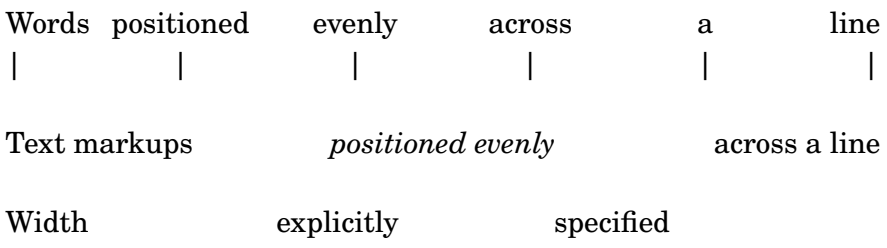
`\fill-line` *args* (markup list)

Put markups *args* into a horizontal line at fixed positions.

If there is a single argument, it gets centered. If there are two arguments, they get aligned to the left and right, respectively. Otherwise, if there are n arguments, the markups are aligned to n equally spaced positions, with the first markup left-aligned, the last markup right-aligned, and the remaining markups centered at the respective positions. If there are no arguments, return an empty stencil.

The width of the horizontal line can be modified by overriding the `line-width` property. The space between arguments is at least as large as the value of the `word-space` property, at the cost of lengthening the line if necessary.

```
\markup {
  \column {
    \fill-line { Words positioned evenly across a line }
    \fill-line { | | | | | | }
    \null
    \fill-line {
      \line { Text markups }
      \line { \italic { positioned evenly } }
      \line { across a line }
    }
    \null
    \override #'(line-width . 50)
    \fill-line { Width explicitly specified }
  }
}
```



See also `\justify-line`.

Used properties:

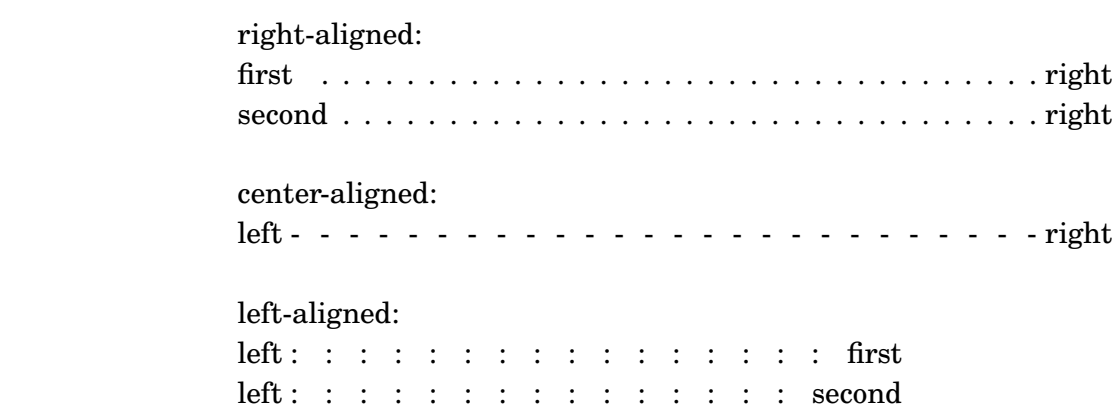
- `line-width` (`#f`)
- `word-space` (`0.6`)
- `text-direction` (`1`)

`\fill-with-pattern` *space* (number) *dir* (direction) *pattern* (markup) *left* (markup) *right* (markup)

Put *left* and *right* at the start and end of a line, respectively, and fill the space in between with repeated *pattern* markups.

Patterns are spaced apart by *space* and aligned to direction *dir*. The width of the line is given by the `line-width` property.

```
\markup \column {
  "right-aligned:"
  \fill-with-pattern #1 #RIGHT . first right
  \fill-with-pattern #1 #RIGHT . second right
  \null
  "center-aligned:"
  \fill-with-pattern #1.5 #CENTER - left right
  \null
  "left-aligned:"
  \override #'(line-width . 50) {
    \fill-with-pattern #2 #LEFT : left first
    \fill-with-pattern #2 #LEFT : left second
  }
}
```



Used properties:

- `line-width`
- `word-space`

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)

Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT two
    three
    \null
    one
    \general-align #X #CENTER two
    three
    \null
    \line {
      one
      \general-align #Y #UP two
```

```
      three
    }
    \null
    \line {
      one
      \general-align #Y #3.2 two
      three
    }
  }
}
```

one
two
three

one
two
three

one two three

one three
two

\halign *dir* (number) *arg* (markup)
: Print *arg* with horizontal alignment set to *dir*.

  If *dir* is -1, *arg* is left-aligned, while +1 makes it right-aligned. Values in between interpolate the alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT two
    three
    \null
    one
    \halign #CENTER two
    three
    \null
    one
    \halign #RIGHT two
    three
    \null
    one
    \halign #-5 two
    three
  }
}
```

one
two
three

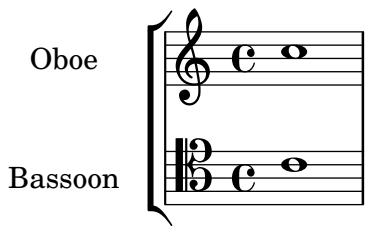
one
two
three

one
two
three

one
two
three

\hcenter-in *length* (number) *arg* (markup)
Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12 Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12 Bassoon
    }
    \clef tenor
    c'1
  }
>>
```



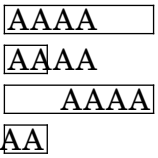
\hspace *amount* (number)
Create an invisible object taking up *amount* horizontal space.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

amount can be also a negative value, which can be best visualized as if the current drawing point gets moved to the left.

```
\markup \concat {
  \hspace #4
  \column {
    \box \concat { AAAA \hspace #4 }
    \box \concat { AAAA \hspace #-4 }
    \box \concat { \hspace #4 AAAA }
    \box \concat { \hspace #-4 AAAA }
  }
}
```



See also `\abs-hspace`.

`\justify` *args* (markup list)
: Print *args* as lines aligned both at the left and the right.

  Like `\wordwrap`, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

The baseline of the output of `\justify` is the baseline of its first line.

Used properties:

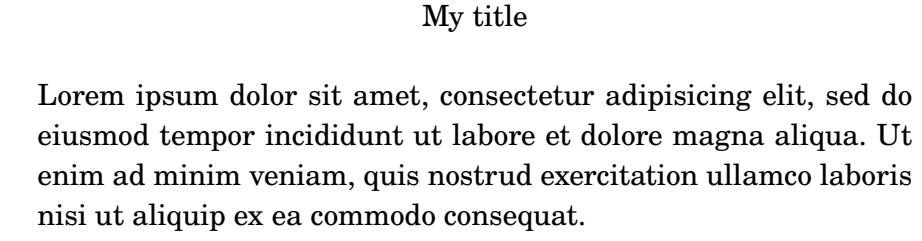
- `text-direction` (`1`)
- `word-space`
- `line-width` (`#f`)
- `baseline-skip`

`\justify-field` *symbol* (symbol)
: Justify the data that has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:myText
    }
  }
}

\markup {
  \null
}
```



`\justify-line` *args* (markup list)

Put markups *args* into a horizontal line, equally spaced.

If there is a single argument, it gets centered. If there are two arguments, they get aligned to the left and right, respectively. Otherwise, the markups are spread to fill the entire line, separated by equally large spaces. If there are no arguments, return an empty stencil.

The width of the horizontal line can be modified by overriding the `line-width` property. The space between arguments is at least as large as the value of the `word-space` property, at the cost of lengthening the line if necessary.

```
\markup {
  \justify-line {
    Constant space between neighboring words
  }
}
```



See also `\fill-line`.

Used properties:

- `line-width` (`#f`)

- word-space (0.6)
- text-direction (1)

\justify-string *arg* (string)

Print string *arg* as lines aligned both at the left and the right.

Paragraphs are indicated by double newlines. Use \override #'(line-width . *X*) to set the line width; *X* is the number of staff spaces.

```
\markup {
  \override #'(line-width . 40)
  \justify-string "Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor incididunt ut
      labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident, sunt
      in culpa qui officia deserunt mollit anim id est
      laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

The baseline of the output of \justify-string is the baseline of its first line.

Used properties:

- text-direction (1)
- word-space
- line-width
- baseline-skip

\left-align *arg* (markup)

Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align two
    three
  }
}
```

one
two
three

`\left-column` *args* (markup list)
Put *args* into a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (markup list)
Put *args* into a horizontal line.

The property `word-space` determines the space between markups in *args*. For right-to-left scripts like Hebrew, `text-direction` should be set to -1.

```
\markup
  \override #'(word-space . 3)
  \column {
    \line { "A B" "C D" "E F" }
    \override #'(text-direction . -1)
    \line { "A B" "C D" "E F" }
  }
```

A B C D E F
E F C D A B

Used properties:

- `text-direction` (1)
- `word-space`

`\lower` *amount* (number) *arg* (markup)
Lower *arg* by the distance *amount*.

A negative *amount* indicates raising; see also `\raise`.

The argument to `\lower` is the vertical displacement amount, measured in (global) staff spaces, which is independent of the markup's current font size. If you need vertical movement that takes the font size into account, use `\translate-scaled` instead.

This function is normally used to move one element inside of a markup relative to the other elements. When using it on the whole markup, bear in mind that spacing mechanisms that place the markup itself on the page could cancel this shift. Consider using grob properties such as `padding`, `Y-offset`, or `extra-offset`, or spacing variables such as `markup-system-spacing`.

```
\markup {
  one
  \lower #3 two
  three
}
```

one three
two

\overlay *args* (markup list)
Take a list of markups *args* and combine them.

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \overlay {
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
    \translate #'(0 . 4) \arrow-head #Y #UP ##f
  }
}
```



\pad-around *amount* (number) *arg* (markup)
Add padding *amount* all around *arg*.
Identical to function \pad-markup.

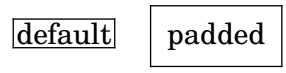
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```

default padded

\pad-markup *amount* (number) *arg* (markup)
Add padding *amount* all around *arg*.
Identical to function \pad-around.

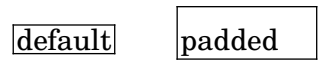
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-markup #1 {
      padded
```

```
    }
  }
}
```



\pad-to-box *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



\pad-x *amount* (number) *arg* (markup)
Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```



\pad-x-left *amount* (number) *arg* (markup)
Add padding *amount* to the left of *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x-left #2 {
      padded
    }
  }
}
```



\pad-x-right *amount* (number) *arg* (markup)
: Add padding *amount* to the right of *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x-right #2 {
      padded
    }
  }
}
```



\put-adjacent *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
: Put *arg2* next to *arg1* along *axis* to the *dir* side, without moving *arg1*.

```
\markup \column {
  text
  \put-adjacent #X #LEFT text *
  text
}
```



\raise *amount* (number) *arg* (markup)
: Raise *arg* by the distance *amount*.

  A negative *amount* indicates lowering, see also \lower.

  The argument to \raise is the vertical displacement amount, measured in (global) staff spaces, which is independent of the markup's current font size. If you need vertical movement that takes the font size into account, use \translate-scaled instead.

  This function is normally used to move one element inside of a markup relative to the other elements. When using it on the whole markup, bear in mind that spacing mechanisms that place the markup itself on the page could cancel this shift. Consider using grob properties such as padding, Y-offset, or extra-offset, or spacing variables such as markup-system-spacing.

```
\markup { C \small \bold \raise #1.0 9/7+ }
```



\right-align *arg* (markup)
: Align *arg* on its right edge.

```
\markup {
  \column {
    one
    \right-align two
    three
  }
}
```

one
two
three

\right-column *args* (markup list)
: Put *args* into a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

\rotate *ang* (number) *arg* (markup)
: Rotate *arg* by *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45°
  }
}
```

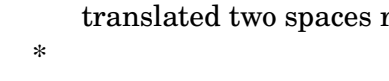
default rotated 45°

\translate *offset* (pair of numbers) *arg* (markup)
: Translate *arg* relative to its surroundings.

offset is a pair of numbers representing the displacement in the X and Y axes. See also `\translate-scaled`.

This function is normally used to move one element inside of a markup relative to the other elements. When using it on the whole markup, bear in mind that spacing mechanisms that place the markup itself on the page could cancel this shift. Consider using grob properties such as `padding`, `X-offset`, `Y-offset` or `extra-offset`, or spacing variables such as `markup-system-spacing`.

```
\markup {
  *
  \translate #'(2 . 3)
    \line { translated two spaces right, three up }
}
```

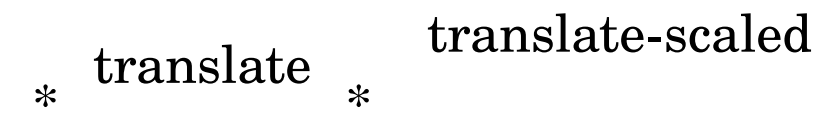


\translate-scaled *offset* (pair of numbers) *arg* (markup)
: Translate *arg* by *offset*, scaling the offset by the font size.

  This function is normally used to move one element inside of a markup relative to the other elements. When using it on the whole markup, bear in mind that spacing mechanisms that place the markup itself on the page could cancel this shift. Consider using grob properties such as `padding`, `X-offset`, `Y-offset` or `extra-offset`, or spacing variables such as `markup-system-spacing`.

  See also `\translate`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```

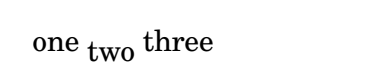


  Used properties:

  - `font-size` (0)

\vcenter *arg* (markup)
: Align *arg* to its Y center.

```
\markup {
  one
  \vcenter two
  three
}
```



\vspace *amount* (number)
: Create an invisible object taking up vertical space of *amount* multiplied by 3.

```
\markup {
    \center-column {
    one
    \vspace #1
    two
    \vspace #3
    three
  }
}
```

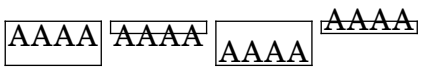
one

two

three

amount can be also a negative value, which can be best visualized as if the current drawing point gets moved up.

```
\markup {
  \vspace #1
  \box \column { AAAA \vspace #0.4 }
  \box \column { AAAA \vspace #-0.4 }
  \box \column { \vspace #0.4 AAAA }
  \box \column { \vspace #-0.4 AAAA }
}
```



See also `\abs-vspace`.

`\wordwrap` *args* (markup list)
Print *args* as left-aligned lines.

This function provides simple word-wrap. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore
    magna aliqua.  Ut enim ad minim veniam, quis nostrud
    exercitation ullamco laboris nisi ut aliquip ex ea
    commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

The baseline of the output of `\wordwrap` is the baseline of its first line.

Used properties:

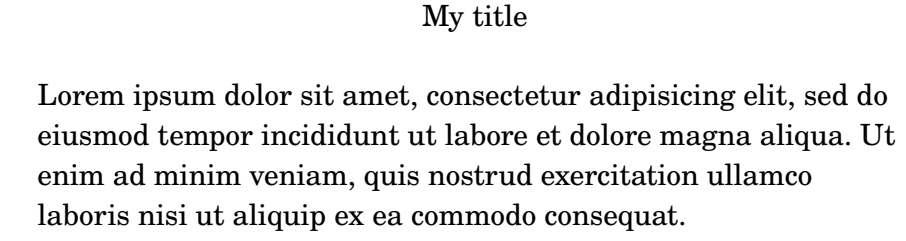
- `text-direction` (`1`)
- `word-space`
- `line-width` (`#f`)
- `baseline-skip`

`\wordwrap-field` *symbol* (symbol)
Word-wrap the data that has been assigned to *symbol*.

```
\header {
  title = "My title"
  myText = "Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut
    labore et dolore magna aliqua.  Ut enim ad minim
    veniam, quis nostrud exercitation ullamco laboris nisi
    ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:myText
    }
  }
}

\markup {
  \null
}
```



`\wordwrap-string` *arg* (string)

Print string *arg* as left-aligned lines.

Paragraphs are indicated by double newlines. Use `\override #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string "Lorem ipsum dolor sit amet,
      consectetur adipisicing elit, sed do eiusmod tempor
      incididunt ut labore et dolore magna aliqua.


      Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea commodo
      consequat.


      Excepteur sint occaecat cupidatat non proident,
      sunt in culpa qui officia deserunt mollit anim id
      est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et
dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

The baseline of the output of `\wordwrap-string` is the baseline of its first line.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

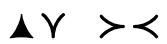
A.1.3 Graphical markup

`\arrow-head` *axis* (integer) *dir* (direction) *filled* (boolean)

Print an arrow head along *axis* in direction *dir*.

Fill the head if *filled* is set to `#t`.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
}
```



`\beam` *width* (number) *slope* (number) *thickness* (number)

Draw a beam with given *width*, *slope*, and *thickness*.

```
\markup {
  \beam #5 #1 #2
}
```



`\bracket` *arg* (markup)

Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note {2.} #UP
  }
}
```

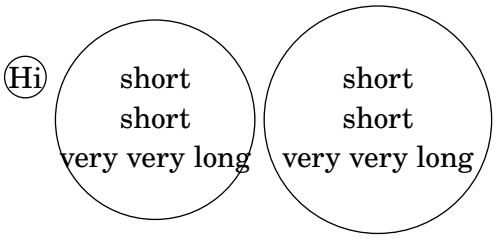


\circle *arg* (markup)

Draw a circle around *arg*.

Use properties `thickness`, `circle-padding`, and `font-size` to set the line thickness and padding around the markup. If `bbox` is set to `#t`, make the circle enclose the bounding box of *arg*, otherwise use either the width or the height of *arg* (whatever is larger) as the diameter.

```
\markup {
  \circle {
    Hi
  }
  \circle {
    \center-column { "short" "short" "very very long" }
  }
  \override #'(bbox . #t) \circle {
    \center-column { "short" "short" "very very long" }
  }
}
```



Note that the circle does not horizontally displace its argument. Use markup commands like `\left-align` or `\table` to make LilyPond realign it.

Used properties:

- `bbox` (`#f`)
- `circle-padding` (`0.2`)
- `font-size` (`0`)
- `thickness` (`1`)

\draw-circle *radius* (number) *thickness* (number) *filled* (boolean)

Draw a circle with given *radius* and *thickness*.

Fill the circle if *filled* is set to `#t`.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```



\draw-dashed-line *dest* (pair of numbers)

Draw a dashed line along vector *dest*.

Properties on and off give the length of a dash and the space between two dashes, respectively; phase shortens the first dash by the given amount.

If the full-length property is set to #t (which is the default), the value of property off (and on under some boundary conditions) gets adjusted so that there is neither whitespace at the end of the line nor the last dash truncated.

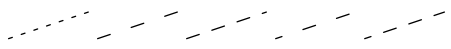
```
\markup {
  \override #'((on . 0.3) (off . 0.5))
  \draw-dashed-line #'(6 . 2)

  \draw-dashed-line #'(6 . 2)

  \override #'(full-length . #f)
  \draw-dashed-line #'(6 . 2)

  \override #'(phase . 0.5)
  \draw-dashed-line #'(6 . 2)

  \override #'((full-length . #f) (phase . 0.5))
  \draw-dashed-line #'(6 . 2)
}
```



Used properties:

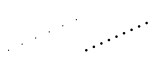
- full-length (#t)
- phase (0)
- off (1)
- on (1)
- thickness (1)

\draw-dotted-line *dest* (pair of numbers)

Draw a dotted line along vector *dest*.

Property off gives the space between two dots; its value gets adjusted so that the first and last dot exactly start and end the line, respectively. phase shifts all dots along the vector by the given amount.

```
\markup {
  \draw-dotted-line #'(5.1 . 2.3)
  \override #'((thickness . 2) (off . 0.2))
  \draw-dotted-line #'(5.1 . 2.3)
}
```



Used properties:

- phase (0)
- off (1)
- thickness (1)

\draw-hline

Draw a horizontal line.

The property `span-factor` sets the length of the line as a multiple of the `line-width` property.

```
\markup {
  \column {
    \draw-hline
    \override #'(span-factor . 1/3)
    \draw-hline
  }
}
```

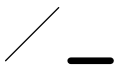


Used properties:

- `span-factor` (1)
- `line-width`
- `thickness` (1)

`\draw-line` *dest* (pair of numbers)
Draw a line along vector *dest*.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```



Used properties:

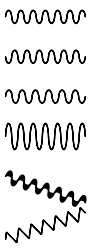
- `thickness` (1)

`\draw-squiggle-line` *sq-length* (number) *dest* (pair of numbers) *eq-end?* (boolean)
Draw a squiggled line along vector *dest*.

sq-length is the length of the first bow; this value gets always adjusted so that an integer number of squiggles is printed. If *eq-end?* is set to `#t`, the squiggled line ends with a bow in the same direction as the starting one.

The appearance of the squiggled line may be customized by overriding the `thickness`, `angularity`, `height`, and `orientation` properties.

```
\markup
  \column {
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(orientation . -1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \draw-squiggle-line #0.5 #'(6 . 0) ##f
    \override #'(height . 1)
    \draw-squiggle-line #0.5 #'(6 . 0) ##t
    \override #'(thickness . 5)
    \draw-squiggle-line #0.5 #'(6 . -2) ##t
    \override #'(angularity . 2)
    \draw-squiggle-line #0.5 #'(6 . 2) ##t
  }
```



Used properties:

- `orientation` (`1`)
- `height` (`0.5`)
- `angularity` (`0`)
- `thickness` (`0.5`)

`\ellipse` *arg* (markup)

Draw an ellipse around *arg*.

Use properties `thickness`, `x-padding`, `y-padding`, and `font-size` to set the line thickness and padding around the markup.

This is the same as function `\oval` but with different padding defaults.

```
\markup {
  \ellipse {
    Hi
  }
}
```



Note that the ellipse does not horizontally displace its argument. Use markup commands like `\left-align` or `\table` to make LilyPond realign it.

Used properties:

- `y-padding` (`0.2`)
- `x-padding` (`0.2`)
- `font-size` (`0`)
- `thickness` (`1`)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an image *file-name*, scaled along *axis* to *size*.

See `\image` for details on this command; calling

```
\markup \epsfile axis size file-name
```

is the same as

```
\markup
  \override #'(background-color . #f)
  \image axis size file-name
```

`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)

Draw a box of dimensions *xext* and *yext*, with rounded corners given by *blot*.

For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e., sharp corners).

```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \combine
    \filled-box #'(1 . 8) #'(0 . 7) #0.2
    \with-color #white
      \filled-box #'(3.6 . 5.6) #'(3.5 . 5.5) #0.7
}
```



`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
    \line {
      one two three
    }
  }
}
```

one two three

`\image` *axis* (number) *size* (number) *file-name* (string)

Inline an image *file-name*, scaled along *axis* to *size*.

The image format is determined based on the extension of *file-name*, which should be `.png` for a PNG image, or `.eps` for an EPS image (`.PNG` and `.EPS` are allowed as well).

EPS files are only supported in the PostScript backend – for all output formats –, and in the Cairo backend – when creating PostScript or EPS output.

If the image has transparency, it will appear over a colored background with the color specified by the `background-color` property, which defaults to `"white"`.

To disable the colored background, set `background-color` to `#f`. For EPS images, this always works (where EPS images work in the first place). On the other hand, for PNG images, it works in the Cairo backend (which can output all supported formats), as well as in the SVG backend, but *not* in the PostScript backend, which is the default. See Voir Section “Advanced command-line options for LilyPond” dans *Utilisation des programmes* for how to activate the Cairo backend.

Use `\withRelativeDir` as a prefix to *file-name* if the file should be found relative to the input file.

Used properties:

- `background-color` (`"white"`)

`\oval` *arg* (markup)

Draw an oval around *arg*.

Use properties `thickness`, `x-padding`, `y-padding`, and `font-size` to set the line thickness and padding around the markup.

This is the same as function `\ellipse` but with different padding defaults.

```
\markup {
  \oval {
    Hi
  }
}
```



Note that the oval does not horizontally displace its argument. Use markup commands like `\left-align` or `\table` to make LilyPond realign it.

Used properties:

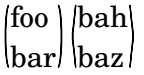
- `y-padding` (`0.75`)
- `x-padding` (`0.75`)
- `font-size` (`0`)
- `thickness` (`1`)

`\parenthesize` *arg* (markup)

Draw parentheses around *arg*.

This is useful for parenthesizing a column containing several lines of text.

```
\markup {
  \parenthesize
  \column {
    foo
    bar
  }
  \override #'(angularity . 2)
  \parenthesize
  \column {
    bah
    baz
  }
}
```



Used properties:

- `width` (`0.25`)
- `line-thickness` (`0.1`)
- `thickness` (`1`)
- `size` (`1`)
- `padding`
- `angularity` (`0`)

`\path` *thickness* (number) *commands* (list)

Draw a path with line *thickness* according to the directions given in *commands*.

commands is a list of lists where the `car` of each sublist is a drawing command and the `cdr` comprises the associated arguments for each command.

There are seven commands available to use in *commands*: `moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `rcurveto`, and `closepath`. Note that the commands that begin

with 'r' are the relative variants of the other three commands. You may also use the standard SVG single-letter equivalents: `moveto` = `M`, `lineto` = `L`, `curveto` = `C`, `closepath` = `Z`. The relative commands are written lowercase: `rmoveto` = `r`, `rlineto` = `l`, `rcurveto` = `c`.

The commands `moveto`, `rmoveto`, `lineto`, and `rlineto` take 2 arguments, namely the X and Y coordinates of the destination point.

The commands `curveto` and `rcurveto` create cubic Bézier curves, and take 6 arguments; the first two are the X and Y coordinates for the first control point, the second two are the X and Y coordinates for the second control point, and the last two are the X and Y coordinates for the destination point.

The `closepath` command takes zero arguments and closes the current subpath in the active path.

Line-cap styles and line-join styles may be customized by overriding the `line-cap-style` and `line-join-style` properties, respectively. Available line-cap styles are `butt`, `round`, and `square`. Available line-join styles are `miter`, `round`, and `bevel`.

The property `filled` specifies whether or not the path is filled with color.

```
samplePath =
  #'((lineto -1 1)
     (lineto 1 1)
     (lineto 1 -1)
     (curveto -5 -5 -5 5 -1 0)
     (closepath))

\markup \scale #'(2 . 2) {
  \path #0.25 #samplePath

  \override #'(line-join-style . miter)
  \path #0.25 #samplePath

  \override #'(filled . #t)
  \path #0.25 #samplePath
}
```



Used properties:

- `filled` (`#f`)
- `line-join-style` (`round`)
- `line-cap-style` (`round`)

`\polygon` *points* (list of number pairs)

A polygon delimited by the list of *points*.

Property `extroversion` defines how the shape of the polygon is adapted to its thickness: if it is 0, the polygon is traced as-is. If it is -1, the outer side of the line is just on the given points. If set to 1, the line has its inner side on the points. The `thickness` property controls the thickness of the line; for filled polygons, this means the diameter of the blot.

```
regularPentagon =
  #'((1 . 0) (0.31 . 0.95) (-0.81 . 0.59)
     (-0.81 . -0.59) (0.31 . -0.95))

\markup \scale #'(2 . 2) {
  \polygon #'((-1 . -1) (0 . -3) (2 . 2) (1 . 2))
  \override #'(filled . #f)
    \override #'(thickness . 2)
      \combine
        \with-color #(universal-color 'blue)
          \polygon #regularPentagon
        \with-color #(universal-color 'vermillion)
          \override #'(extroversion . 1)
            \polygon #regularPentagon
}
```



Used properties:

- `thickness` (1)
- `filled` (#t)
- `extroversion` (0)

`\postscript` *str* (string)

Insert *str* directly into the output as a PostScript command string.

This command is meant as a *last resort*. Almost all needs are better fulfilled by other markup commands (see, for example, `\path` and `\draw-line`). If you do use this command, keep the following points in mind:

- `\postscript` does not work in SVG output.
- Only a subset of the PostScript language is supported during the conversion from PostScript to PDF.
- There are no stability guarantees on the details of how LilyPond produces its own output (i.e., the context into which the PostScript code is inserted). They may change substantially across versions.
- LilyPond cannot understand the shape of the drawing, leading to suboptimal spacing. Usually, it is necessary to explicitly set up dimensions with a command like `\with-dimensions`.
- Depending on how you install LilyPond, the version of the PostScript interpreter (Ghostscript) can vary, and some of its features may be disabled.

str is processed with the following constraints.

- The string is embedded into the (intermediate) output file with the PostScript commands

  ```
  gsave currentpoint translate 0.1 setlinewidth
  ```

  before and

  ```
  grestore
  ```
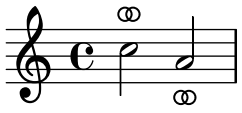
  after it.

- After these preceding commands (i.e., `currentpoint translate`) the origin of the current transformation is the reference point of `\postscript`. Scale and rotation of the current transformation reflect the global staff line distance and (if applied) other transformation markup commands (e.g., `\scale` and `\rotate`) encapsulating the `\postscript` command.
- The current point is set to the coordinate (0, 0).
- If an unwanted line appears at the beginning of your PostScript code, you are probably missing a call to `newpath`.

```
ringsps = "
  0.15 setlinewidth
  0.9 0.6 moveto
  0.4 0.6 0.5 0 361 arc
  stroke
  1.0 0.6 0.5 0 361 arc
  stroke
  "

rings = \markup {
  \with-dimensions #'(-0.2 . 1.6) #'(0 . 1.2)
  \postscript #ringsps
}

\relative c'' {
  c2^\rings
  a2_\rings
}
```
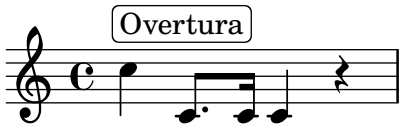


`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*.

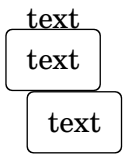
This function looks at properties `thickness`, `box-padding`, and `font-size` to determine the line thickness and padding around the *arg*. The `corner-radius` property defines the radius of the round corners (default value is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```



Note that the box does not horizontally displace its argument. Use markup commands like `\left-align` or `\table` to make LilyPond realign it.

```
\markup {
  \override #'(box-padding . 1.5)
  \column {
    "text"
    \rounded-box "text"
    \left-align \rounded-box "text"
  }
}
```



Used properties:

- `box-padding` (`0.5`)
- `font-size` (`0`)
- `corner-radius` (`1`)
- `thickness` (`1`)

`\scale` *factor-pair* (pair of numbers) *arg* (markup)

Scale *arg*.

factor-pair is a pair of numbers representing the scaling factor of the X and Y axes. Negative values may be used to produce mirror images.

```
\markup {
  \line {
    \scale #'(2 . 1)
    stretched
    \scale #'(1 . -1)
    mirrored
  }
}
```



`\triangle` *filled* (boolean)

Draw a triangle.

Fill the triangle if *filled* is set to `#t`.

The appearance can be controlled with properties `extroversion`, `font-size`, and `thickness`.

```
\markup {
  \triangle ##t
  \triangle ##f
  \override #'(font-size . 5)
  \override #'(thickness . 5) {
    \override #'(extroversion . 1)
    \triangle ##f
    \override #'(extroversion . -1)
    \triangle ##f
  }
}
```



Used properties:

- `thickness` (1)
- `font-size` (0)
- `extroversion` (0)

`\with-url` *url* (string) *arg* (markup)
Add a link to URL *url* around *arg*.

This only works in the PDF backend.[2]

```
\markup {
  \with-url "https://lilypond.org/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

A.1.4 Markup for music and musical symbols

`\accidental` *alteration* (an exact rational number)
Select an accidental glyph for *alteration*, given as a rational number.

Use `\text-accidental` instead if you need glyph representation forms that fit and align well with text.

```
\markup {
  text
  \tiny { \accidental #1/2 \accidental #-1/2 }
  text
  \tiny { \text-accidental #1/2 \text-accidental #-1/2 }
  text
}
```

text ♯𝄲 text ♯𝄲 text

Used properties:

- `alteration-glyph-name-alist`

`\bar-line` *strg* (string)
Print a bar line in markup.

The allowed characters for input string *strg* are ‘`;|.:!S[]{}`’, having the same meaning as with the `\bar` command. The additional characters ‘`{`’ and ‘`}`’ denote left and right braces, respectively.

The output is vertically centered.

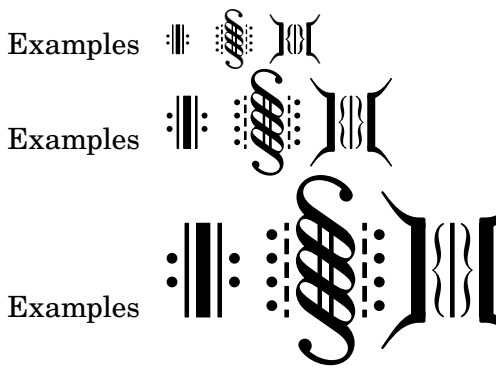
Changes of `font-size` are respected.

The default of `height` is 4 staff-space units. Apart from the bracket tips of a bracket bar line and the segno bar line all other bar lines are scaled with `height`. We don’t scale bracket tips and segno to meet the behavior of `SystemStartBracket` and the segno bar line.

[2] Due to technical limitations the link doesn’t work here in the Notation Reference.

\bar-line is further customizable by overriding dot-count and dash-count for dotted and dashed bar lines. The values for hair-thickness, kern and thick-thickness are customizable as well; defaults are the same as the values of the corresponding BarLine grob properties.

```
\markup {
   \override #'(word-space . 2)
   \column {
      \line {
        Examples
        \fontsize #-5 \translate-scaled #'(0 . 2) {
          \bar-line ":|.|:"
          \bar-line ";!S!;"
          \bar-line "]{|}["
        }
      }
      \line {
        Examples
        \fontsize #0 \translate-scaled #'(0 . 2) {
          \bar-line ":|.|:"
          \bar-line ";!S!;"
          \bar-line "]{|}["
        }
      }
      \line {
        Examples
        \fontsize #5 \translate-scaled #'(0 . 2) {
          \bar-line ":|.|:"
          \bar-line ";!S!;"
          \bar-line "]{|}["
        }
      }
   }
}
```



Used properties:

- thick-thickness (6.0)
- kern (3.0)
- hair-thickness (1.9)
- dash-count (5)
- dot-count (4)

- `height` (4)
- `font-size` (0)

`\coda` Draw a coda sign.

```
\markup {
  \coda
}
```

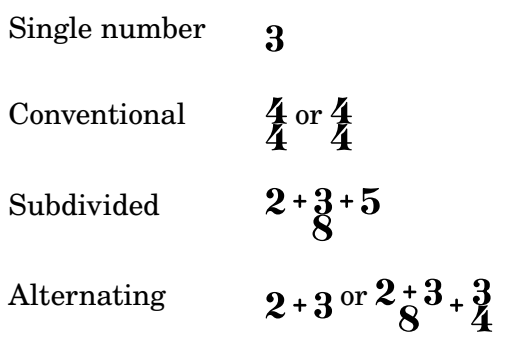


`\compound-meter` *time-sig* (number or pair)

Draw a numeric time signature based on *time-sig*.

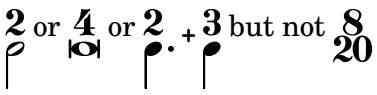
time-sig can be a single number, a pair of numbers, a simple list, or a list of lists, as the following example demonstrates.

```
\markuplist {
  \override #'(baseline-skip . 4.5)
  \override #'(padding . 4.5)
  \table #'(-1 -1) {
    "Single number"  \compound-meter #3
    "Conventional"   \line {
                       \compound-meter #'(4 . 4) or
                       \compound-meter #'(4 4)
                     }
    "Subdivided"     \compound-meter #'(2 3 5 8)
    "Alternating"    \line {
                       \compound-meter #'((2) (3)) or
                       \compound-meter #'((2 3 8) (3 4))
                     }
  }
}
```



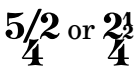
Setting the `denominator-style` property to `note` prints denominators as a note and dots when exact representation is possible. Example:

```
\markup {
  \override #'(denominator-style . note)
  \line {
    \compound-meter #'(2 2) or
    \compound-meter #'(4 1/2) or
    \compound-meter #'((2 8/3) (3 4)) but not
    \compound-meter #'(8 20)
  }
}
```



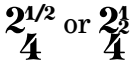
The `nested-fraction-mixed` property controls whether fractional parts are printed as mixed numbers or as common fractions. Example:

```
\markup {
  \override #'(nested-fraction-mixed . #f)
  \compound-meter #'(5/2 4) or
  \override #'(nested-fraction-mixed . #t)
  \compound-meter #'(5/2 4)
}
```



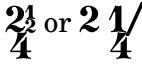
The `nested-fraction-orientation` property controls how nested fractions are arranged. Supported values are `horizontal` and `vertical`. Example:

```
\markup {
  \override #'(nested-fraction-orientation . horizontal)
  \compound-meter #'(5/2 4) or
  \override #'(nested-fraction-orientation . vertical)
  \compound-meter #'(5/2 4)
}
```



The `nested-fraction-relative-font-size` property controls the size of the numerals in nested fractions. Recommended values are -5.5 and 0. Using large numerals may take precedence over related properties. Example:

```
\markup {
  \override #'(nested-fraction-relative-font-size . -5.5)
  \compound-meter #'(5/2 4) or
  \override #'(nested-fraction-relative-font-size . 0)
  \compound-meter #'(5/2 4)
}
```



Used properties:

- `note-staff-position` (-2)
- `note-head-style` (())
- `note-flag-style` (())
- `note-dots-direction` (0)
- `nested-fraction-relative-font-size` (())
- `nested-fraction-orientation` (default)
- `nested-fraction-mixed` (#t)
- `font-size` (0)
- `denominator-style` (default)

\customTabClef *num-strings* (integer) *staff-space* (number)

Draw a clef in sans-serif style for a tablature with *num-strings* lines spaced by *staff-space*.

This markup command is used to implement `\clef moderntab` within a `TabStaff` context.

```
\markup {
  \customTabClef #4 #1
}
```



\doubleflat

Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```



\doublesharp

Draw a double sharp symbol.

```
\markup {
  \doublesharp
}
```



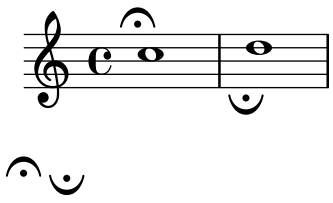
\fermata Create a fermata glyph.

If property `direction` is `DOWN`, use an inverted glyph.

Note that within music, one would normally use the `\fermata` articulation instead of a markup.

```
{ c''1^\markup \fermata d''1_\markup \fermata }

\markup { \fermata \override #`(direction . ,DOWN) \fermata }
```



Used properties:

- `direction` (1)

\flat Draw a flat symbol.

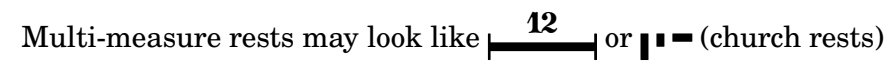
```
\markup {
  \flat
}
```



\multi-measure-rest-by-number *length* (non-negative, exact integer)
Return a multi-measure rest symbol for *length* measures.

If the number of measures is greater than the number given by `expand-limit` a thick horizontal line is printed. For every multi-measure rest lasting more than one measure a number is printed on top. However, if property `multi-measure-rest-number` is set to `#t`, this number gets suppressed.

```
\markup {
  Multi-measure rests may look like
  \multi-measure-rest-by-number #12
  or
  \override #'(multi-measure-rest-number . #f)
    \multi-measure-rest-by-number #7
  (church rests)
}
```



Used properties:

- `multi-measure-rest-number` (`#t`)
- `width` (`8`)
- `expand-limit` (`10`)
- `hair-thickness` (`2.0`)
- `thick-thickness` (`6.6`)
- `word-space`
- `style` (`()`)
- `font-size` (`0`)

\musicglyph *glyph-name* (string)
Print music symbol *glyph-name*.

See Section “The Emmentaler font” dans *Manuel de notation* for a complete listing of the possible glyph names.

```
\markup {
  \musicglyph "f"
  \musicglyph "rests.2"
  \musicglyph "clefs.G_change"
}
```



\natural Draw a natural symbol.

```
\markup {
  \natural
}
```



`\note` *duration* (duration) *dir* (number)

Draw a note with a given duration and a stem.

The note duration is specified with *duration* (setting both the note head type and the number of augmentation dots). The stem direction and length is given by *dir*.

This function is wrapper around `\note-by-number`; its documentation gives more details on the available properties.

```
\markup {
  \note {4..} #UP
  \hspace #2
  \override #'(style . cross)
    \note {4..} #0.75
  \hspace #2
  \note {\breve} #0
}
```



Used properties:

- `style` (`default`)
- `dots-direction` (`0`)
- `flag-style` (`()`)
- `font-size` (`0`)

`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)

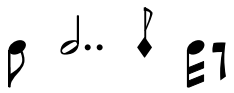
Draw a note with a given length, a number of dots, and a stem.

The note length is specified with *log*, the number of dots with *dot-count*, and the stem direction and length with *dir*. Fractional values for *dir* can be used to control the length of the stem. Value 0 suppresses the stem completely; this is useful for note head styles that don't take stems or come with built-in stems (like Kievan).

Ancient note head styles (via the `style` property, voir Section B.9 [Styles de tête de note], page 937) get mensural-style flags by default; use the `flag-style` property to override this. Supported flag styles are `default`, `old-straight-flag`, `modern-straight-flag`, `flat-flag`, `stacked`, `mensural`, and `neomensural`. The last flag style is the same as `mensural` and provided for convenience.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
  \hspace #2
  \override #'(style . petrucci)
    \note-by-number #3 #0 #UP
  \hspace #2
  \override #'(flag-style . modern-straight-flag)
    \note-by-number #4 #0 #DOWN
  \override #'(style . kievan)
```

```
    \note-by-number #2 #0 #0
}
```



See also function \note.

Used properties:

- style (default)
- dots-direction (0)
- flag-style (())
- font-size (0)

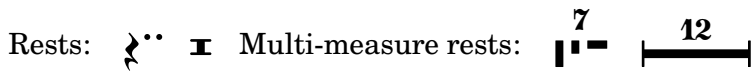
\rest *duration* (duration)

Return a rest symbol with length *duration*.

If the multi-measure-rest property is set to #t, a multi-measure rest symbol may be returned. In this case the duration needs to be entered as { 1**N* } to get a multi-measure rest for *N* bars. Actually, only the scaling factor (i.e., the number after '*') determines the length; the basic duration is disregarded.

See also functions \rest-by-number and \multi-measure-rest-by-number for more information on the used properties.

```
\markup {
  Rests:
  \hspace #2
  \rest { 4.. }
  \hspace #2
  \rest { \breve }
  \hspace #2
  Multi-measure rests:
  \override #'(multi-measure-rest . #t)
  {
    \hspace #2
    \rest { 1*7 }
    \hspace #2
    \rest { 1*12 }
  }
}
```



Used properties:

- multi-measure-rest-number (#t)
- width (8)
- expand-limit (10)
- hair-thickness (2.0)
- thick-thickness (6.6)
- word-space
- style (())
- font-size (0)

- `style` (())
- `ledgers` ((-1 0 1))
- `font-size` (0)

`\rest-by-number` *log* (integer) *dot-count* (integer)

Draw a rest of length *log*, with *dot-count* dots.

For duration logs that appear in the `ledgers` property, rest symbols with ledger lines are selected.

```
\markup {
  \rest-by-number #3 #2
  \hspace #2
  \rest-by-number #0 #1
  \hspace #2
  \rest-by-number #-1 #0
  \hspace #2
  \override #'(ledgers . ())
  \rest-by-number #-1 #0
}
```
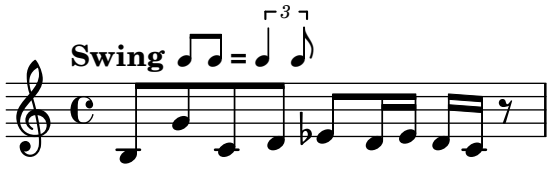


Used properties:

- `style` (())
- `ledgers` ((-1 0 1))
- `font-size` (0)

`\rhythm` *music* (music)

Draw embedded rhythmic pattern as specified by *music*.

```
\relative {
  \tempo \markup {
    Swing
    \hspace #0.4
    \rhythm { 8[ 8] } = \rhythm { \tuplet 3/2 { 4 8 } }
  }
  b8 g' c, d ees d16 ees d c r8
}
```



Within `\rhythm`, there is no time signature and no division in measures (as with `\cadenzaOn`, voir Section 2.3.5 [Musique sans métrique], page 94). Beaming must be added explicitly with the syntax explained in Section 2.4.3 [Barres de ligature manuelles], page 117.

```
\markup {
  The rhythmic pattern \rhythm { 16[ 8 16] } is
  a type of syncopation.
}
```

The rhythmic pattern  is a type of syncopation.

\stemDown can be used to flip the stems.

```
\markup \rhythm { \stemDown 8 16 8 }
```



\rhythm works by creating a StandaloneRhythmVoice context, which is enclosed in a StandaloneRhythmStaff context, which is enclosed in a StandaloneRhythmScore context. It is possible to apply global tweaks to the output by using a \layout block.

```
\layout {
  \context {
    \StandaloneRhythmVoice
    \xNotesOn
  }
}

\markup \rhythm { 8 16 8 }
```



Used properties:

- font-size (-2)

\score *score* (score)
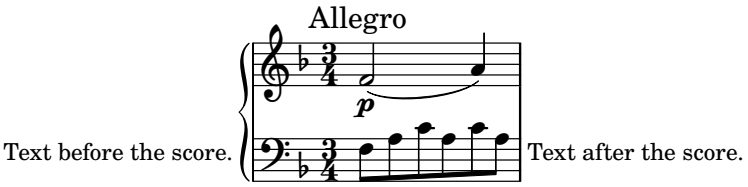
Inline an image of music as specified by *score*.

The reference point (usually the middle staff line) of the lowest staff in the top system is placed on the baseline.

No page breaks and no MIDI output, i.e., both \pageBreak commands and \midi{} blocks get ignored.

```
\markup {
  Text before the score.
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
      }
    >>
```

```
    \layout {
      indent = 0.0\cm
    }
  }
  Text after the score.
}
```



Used properties:

- `baseline-skip`

`\segno` Draw a segno symbol.

```
\markup {
  \segno
}
```


`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```


`\semisharp`

Draw a semisharp symbol.

```
\markup {
  \semisharp
}
```


`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```


`\sesquisharp`

Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```



\sharp Draw a sharp symbol.

```
\markup {
  \sharp
}
```

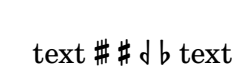


\text-accidental *alteration* (an exact rational number)
Get an accidental for *alteration* that aligns well with text.

alteration must be a rational number; the function uses the `alteration-glyph-name-alist` property to map this number to an accidental glyph.

If no text variant glyph for the accidental is found, markup is constructed to fittingly scale and position the (non-text) accidental glyph.

```
\markup {
  text
  \tiny { \text-accidental #3/4 \text-accidental #1/2
          \text-accidental #-1/4 \text-accidental #-1/2 }
  text
}
```



Used properties:

- `alteration-glyph-name-alist`

\text-common-time
Draw a common time symbol for text.

```
\markup {
  \text-common-time
}
```



\text-cut-time
Draw a cut time symbol for text.

```
\markup {
  \text-cut-time
}
```



\text-doubleflat
Draw a double flat symbol for text.

```
\markup {
  \text-doubleflat
}
```



`\text-doublesharp`
: Draw a double sharp symbol for text.

```
\markup {
  \text-doublesharp
}
```



`\text-flat`
: Draw a flat symbol for text.

```
\markup {
  \text-flat
}
```



`\text-natural`
: Draw a natural symbol for text.

```
\markup {
  \text-natural
}
```



`\text-sharp`
: Draw a sharp symbol for text.

```
\markup {
  \text-sharp
}
```



`\tied-lyric` *str* (string)
: Replace '~' tilde symbols with tie characters in *str*.

```
\markup \column {
  \tied-lyric
    "Siam navi~all'onde~algenti Lasciate~in abbandono"
  \tied-lyric
    "Impetuosi venti I nostri~affetti sono"
  \tied-lyric
    "Ogni diletto~e scoglio Tutta la vita~e~un mar."
}
```

Siam navi‿all'onde‿algenti Lasciate‿in abbandono
Impetuosi venti I nostri‿affetti sono
Ogni diletto‿e scoglio Tutta la vita‿e‿un mar.

Used properties:

- `word-space`

`\varcoda` Draw a varcoda sign.

```
\markup {
  \varcoda
}
```

𝍬

A.1.5 Conditional markup

`\if` *condition?* (procedure) *argument* (markup)

Test *condition?*, and only insert *argument* if it is true.

The condition is provided as a procedure taking an output definition and a property alist chain. The procedure is applied, and its result determines whether to print the markup. This command is most useful inside `oddHeaderMarkup` or similar. Here is an example printing page numbers in bold:

```
\paper {
  oddHeaderMarkup =
    \markup \fill-line {
      ""
      \if #print-page-number
           \bold \fromproperty #'page:page-number-string
    }
  evenHeaderMarkup =
    \markup \fill-line {
      \if #print-page-number
           \bold \fromproperty #'page:page-number-string
      ""
    }
}
```

`\unless` *condition?* (procedure) *argument* (markup)

Test *condition?*, and only insert *argument* if it is false.

This function is similar to `\if`; the following example shows how to print the copyright notice on all pages but the last instead of just the first page.

```
\paper {
  oddFooterMarkup = \markup {
    \unless #on-last-page-of-part \fill-line {
      \fromproperty #'header:copyright
    }
  }
}

\header {
  copyright = "© LilyPond Authors. License: GFDL."
  tagline = "© LilyPond Authors.  Documentation placed
under the GNU Free Documentation License
version 1.3."
}
```

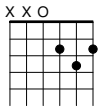
A.1.6 Instrument-specific markup

`\fret-diagram` *definition-string* (string)

Make a (guitar) fret diagram based on *definition-string*.

For example, say

```
\markup \fret-diagram "s:1.25;6-x;5-x;4-o;3-2;2-3;1-2;"
```



for fret spacing 5/4 of staff space, D chord diagram.

Syntax rules for *definition-string*:

– Diagram items are separated by semicolons.

– Possible items:

- `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
- `t:`*number* – Set the line thickness (relative to normal line thickness). Default: 0.5.
- `h:`*number* – Set the height of the diagram in frets. Default: 4.
- `w:`*number* – Set the width of the diagram in strings. Default: 6.
- `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
- `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
- `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
- `c:`*string1-string2-fret* – Include a barre mark from *string1* to *string2* on *fret*.
- *string-fret* – Place a dot on *string* at *fret*. If *fret* is '`o`', *string* is identified as open. If *fret* is '`x`', *string* is identified as muted.
- *string-fret-fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.

– Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (`0.5`)
- `fret-diagram-details`
- `size` (`1.0`)
- `align-dir` (`-0.4`)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

For example,

```
\markup \fret-diagram-terse "x;x;o;2;3;2;"
```



displays a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.

- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a '`-`' followed by the finger indicator, e.g., '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

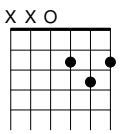
- `thickness` (`0.5`)
- `fret-diagram-details`
- `size` (`1.0`)
- `align-dir` (`-0.4`)

`\fret-diagram-verbose` *marking-list* (pair)

Make a fret diagram containing the symbols indicated in *marking-list*.

The following example produces a standard D chord diagram without fingering indications.

```
\markup \scale #'(1.5 . 1.5)
  \fret-diagram-verbose
    #'((mute 6) (mute 5) (open 4)
       (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```



Possible elements in *marking-list*:

`(mute` *`string-number`*`)`

Place a small 'x' at the top of string *string-number*.

`(open` *`string-number`*`)`

Place a small 'o' at the top of string *string-number*.

`(barre` *`start-string end-string fret-number`*`)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(capo` *`fret-number`*`)`

Place a capo indicator (a large solid bar) across the entire fretboard at fret location *fret-number*. Also, set fret *fret-number* to be the lowest fret on the fret diagram.

`(place-fret` *`string-number fret-number`* `[`*`finger-value`*`] [`*`color-modifier`*`] [`*`color`*`] ['parenthesized ['default-paren-color]])`

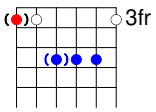
Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*, an optional color modifier *color-modifier*, an optional color *color*, an optional parenthesis `'parenthesized` and an optional parenthesis color `'default-paren-color`.

By default, the fret playing indicator is a solid dot. This can be globally changed by setting the value of the property `dot-color` or for a single dot by setting the value of *color*. The dot can be parenthesized by adding `'parenthesized`. By default the color for the parenthesis is taken from

the dot. Adding `'default-paren-color` will take the parenthesis color from the global `dot-color` property; as a fallback black will be used. Setting *color-modifier* to `inverted` inverts the dot color for a specific fingering.

The values for *string-number*, *fret-number*, and the optional *finger* should be entered first in that order. The order of the other optional arguments does not matter. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

```
\markup \scale #'(1.5 . 1.5)
  \fret-diagram-verbose #'(
    (place-fret 6 3 1 red parenthesized default-paren-color)
    (place-fret 5 3 1 inverted)
    (place-fret 4 5 2 blue parenthesized)
    (place-fret 3 5 3 blue)
    (place-fret 2 5 4 blue)
    (place-fret 1 3 1 inverted)
  )
```



Used properties:

- `thickness` (`0.5`)
- `fret-diagram-details`
- `size` (`1.0`)
- `align-dir` (`-0.4`)

`\harp-pedal` *definition-string* (string)

Make a harp pedal diagram containing the symbols indicated in *definition-string*.

Possible elements in *definition-string*:

`^`	pedal is up
`-`	pedal is neutral
`v`	pedal is down
`|`	vertical divider line
`o`	the following pedal should be circled (indicating a change)

```
\markup \harp-pedal "^-v|--ov^"
```



The function also checks whether the string has the typical form of three pedals, then the divider, and then the remaining four pedals, printing a warning otherwise (without preventing the non-standard order).

Use the `size` property to control the overall size, and the `thickness` property for the line thickness of the horizontal line and the divider.

The remaining configuration is done via the `harp-pedal-details` property; it contains the following elements:

`box-offset`
: vertical shift of the box center for up/down pedals

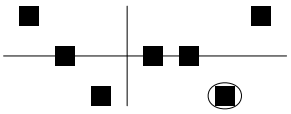
`box-width`
: box width

`box-height`
: box height

`space-before-divider`
: spacing between two boxes before the divider

`space-after-divider`
: spacing between two boxes after the divider

```
\markup {
  \override #'((size . 1.5)
               (harp-pedal-details . ((box-width . 1)
                                      (box-offset . 2))))
  \harp-pedal "^-v|--ov^"
}
```



For global modification of `harp-pedal-details`, i.e., outside of the current `\markup` block, you can also use code similar to

```
\override Voice.TextScript.harp-pedal-details.box-width = 1
```

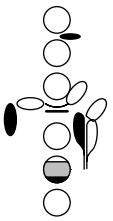
Used properties:

- `thickness` (`0.5`)
- `harp-pedal-details` (`()`)
- `size` (`1.2`)

`\woodwind-diagram` *instrument* (symbol) *user-draw-commands* (list)
Make a woodwind-instrument diagram for *instrument* using *user-draw-commands*.

user-draw-commands is a list of alists, specifying the left-hand keys, the elements on the central column, and the right-hand keys. For example, this diagram

```
\markup \woodwind-diagram
  #'oboe #'((lh . (d ees))
            (cc . (five3qT1q))
            (rh . (gis)))
```



shows an oboe with the left-hand d key, left-hand ees key, and right-hand gis key depressed, while the five-hole of the central column effectuating a trill between 1/4 and 3/4 is closed.

The following instruments are supported:

– piccolo
– flute
– oboe
– clarinet
– bass-clarinet
– saxophone
– bassoon
– contrabassoon

To see all of the callable keys for a given instrument, include the function call `(print-keys '`*instrument*`)` in your `.ly` file, where *instrument* is the instrument whose keys you want to print.

Certain keys allow for special configurations. The entire gamut of configurations possible is as follows:

`1q`	1/4 covered
`1h`	1/2 covered
`3q`	3/4 covered
`R`	ring depressed
`F`	fully covered; the default if no state put

Additionally, these configurations can be used in trills. So, for example, `three3qTR` effectuates a trill between 3/4 full and ring depressed on the three hole. As another example, `threeRT` effectuates a trill between R and open, whereas `threeTR` effectuates a trill between open and shut. To see all of the possibilities for all of the keys of a given instrument, invoke `(print-keys-verbose '`*instrument*`)`.

Lastly, substituting an empty list for the pressed-key alist results in a diagram with all of the keys drawn but none filled, for example

```
\markup \woodwind-diagram #'flute #'()
```



Used properties:

- `woodwind-diagram-details` (`()`)
- `font-size` (`0`)
- `graphical` (`#t`)
- `thickness` (`0.1`)
- `size` (`1`)

A.1.7 Accordion registers

`\discant` *name* (string)

Generate a discant accordion register symbol for *name*.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The register names in the default `\discant` register set have been modeled after the numeric Swiss notation (as depicted in `http://de.wikipedia.org/wiki/Register_%28Akkordeon%29`), omitting the slashes and dropping leading zeros.

The string *name* is basically a three-digit number with the lowest digit specifying the number of 16' reeds, the tens the number of 8' reeds, and the hundreds specifying the number of 4' reeds. Without modification, the specified number of reeds in 8' is centered in the symbol. Newer instruments may have registrations where 8' can be used either within or without a tone chamber, 'cassotto'. Notationally, the central dot then indicates use of cassotto. One can suffix the tens' digits '`1`' and '`2`' with '`+`' or '`-`' to indicate clustering the dots at the right or left, respectively, rather than centered.

Some examples are



```
\discant "1"    \discant "1+0"
```



```
\discant "120"   \discant "131"
```

Used properties:

- `font-size` (`0`)

`\freeBass` *name* (string)

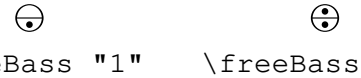
Generate a free bass/converter accordion register symbol for the usual two-reed layout as given by *name*.

To make it available,

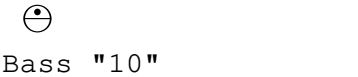
```
#(use-modules (lily accreg))
```

is required near the top of your input file.

Available registrations are



```
\freeBass "1"    \freeBass "11"
```



```
\freeBass "10"
```

Used properties:

- `font-size` (`0`)

`\stdBass` *name* (string)

Generate a standard bass accordion register symbol for *name*.

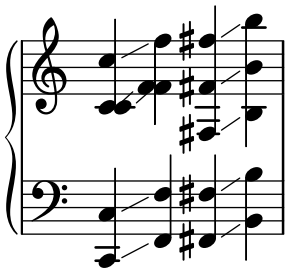
To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The default bass register definitions have been modeled after the article `http://www.accordions.com/index/art/stradella.shtml` originally appearing in *Accord Magazine*.

The underlying register model is



This kind of overlapping arrangement is common for Italian instruments though the exact location of the octave breaks differ.

When not composing for a particular target instrument, using the five-reed definitions makes more sense than using a four-reed layout: in that manner, the '`Master`' register is unambiguous. This is rather the rule in literature bothering about bass registrations at all.

Available registrations are



`\stdBass "Soprano"`



`\stdBass "Soft Bass"`



`\stdBass "Alto"`



`\stdBass "Soft Tenor"`



`\stdBass "Tenor"`



`\stdBass "Bass/Alto"`



`\stdBass "Master"`

Used properties:

- `font-size` (0)

`\stdBassIV` *name* (string)
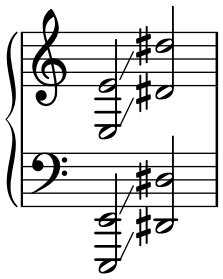
Generate a standard bass accordion register symbol for *name*.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The main use is for four-reed standard bass instruments with reedbank layout



Notable instruments are Morino models with MIII (the others are five-reed instead) and the Atlantic IV. Most of those models have three register switches. Some newer Morinos with MIII might have five or even seven.

The prevalent three-register layout uses the middle three switches '`Tenor`', '`Master`', '`Soft Bass`'. Note that the sound is quite darker than the same registrations of '`c,`'-based instruments.

Available registrations are



`\stdBassIV "Soprano"`



`\stdBassIV "Soft Bass"`



`\stdBassIV "Alto"`



`\stdBassIV "Bass/Alto"`



`\stdBassIV "Tenor"`



`\stdBassIV "Soft Bass/Alto"`



`\stdBassIV "Master"`



`\stdBassIV "Soft Tenor"`

Used properties:

- `font-size` (0)

`\stdBassV` *name* (string)
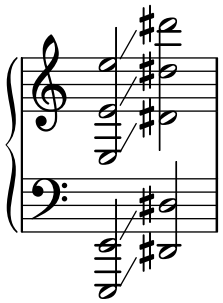
Generate a standard bass accordion register symbol for *name*.

To make it available,

```
#(use-modules (lily accreg))
```

is required near the top of your input file.

The main use is for five-reed standard bass instruments with reedbank layout



This tends to be the bass layout for Hohner's Morino series without converter or MIII manual.

With the exception of the rather new 7-register layout, the highest two chord reeds are usually sounded together. Older instruments offer 5 or 3 bass registers. The Tango VM offers an additional '`Solo Bass`' setting that mutes the chord reeds. The symbol on the register buttons of the Tango VM would actually match the physical five-octave layout reflected here, but it is not used in literature.

Composers should likely prefer the five-reed versions of these symbols. The mismatch of a four-reed instrument with five-reed symbols is easier to resolve for the player than the other way round.

Available registrations are



Used properties:

- `font-size` (0)

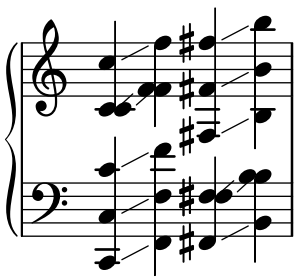
`\stdBassVI` *name* (string)
: Generate a standard bass accordion register symbol for six-reed basses as given by *name*.

  To make it available,

  ```
  #(use-modules (lily accreg))
  ```

  is required near the top of your input file.

  This is primarily the register layout for the Hohner « Gola » model. The layout is



The registers are effectively quite similar to that of \stdBass. An additional bass reed at alto pitch is omitted for aesthetical reasons from the 'Master' setting, so the symbols are almost the same except for the 'Alto/Soprano' register with bass notes at Alto pitch and chords at Soprano pitch.

Available registrations are



\stdBassVI "Soprano"



\stdBassVI "Alto/Soprano"



\stdBassVI "Alto"



\stdBassVI "Bass/Alto"



\stdBassVI "Soft Tenor"



\stdBassVI "Soft Bass"



\stdBassVI "Master"

Used properties:

- font-size (0)

A.1.8 Other markup commands

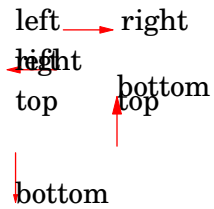
\annotate-moving *arg* (markup)

Indicate \vspace and \hspace movement with an arrow.

The arrow changes its size and thickness depending on the printed length; the maximum size of the arrow head can be controlled with the size property. If size exceeds a third of the length of the final arrow, it falls back to that third.

Note that the arrows do not reflect the actual extents of the objects created by \vspace and \hspace; you might use \box for that.

```
\markup
  \column {
    \line { left \annotate-moving \hspace #4 right }
    \line { left \annotate-moving \hspace #-4 right }
    \line {
      \column {
        top
        \override #'(size . 0.6)
          \annotate-moving \vspace #4/3 bottom
      }
      \column {
        top
        \override #'(size . 2.0)
          \annotate-moving \vspace #-4/3 bottom
      }
    }
  }
```



Used properties:

- `size` (`1`)
- `color` (`"red"`)

`\append-to-tag` *tag* (symbol) *more* (markup) *arg* (markup)

Append *more* to all markup in var *arg* tagged with *tag*.

It works similar to `\appendToTag` for music, but only with markups.

```
tagged = \markup {
  \tag #'foo A
  \tag #'bar B
}

\markup { \append-to-tag #'foo postfoo \tagged }
```

A postfoo B

Used properties:

- `tags-with-appends-alist` (`()`)

`\auto-footnote` *mkup* (markup) *note* (markup)

Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \auto-footnote a b
  \override #'(padding . 0.2)
  \auto-footnote c d
}
```

a[1] c[2]

[1] b
[2] d

The footnote will be annotated automatically.

Used properties:

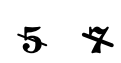
- `padding` (`0.0`)

- `raise` (`0.5`)

`\backslashed-digit` *num* (integer)
: Print number *num* with the Emmentaler font, crossed through with a backslash.

  This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```



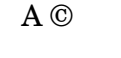
  Used properties:

  - `thickness` (`1.6`)
  - `font-size` (`0`)

`\char` *num* (integer)
: Produce a single Unicode character with code *num*.

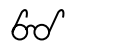
  Characters encoded in hexadecimal format require the prefix `#x`.

```
\markup {
  \char #65 \char ##x00a9
}
```



`\eyeglasses`
: Prints out eyeglasses, indicating strongly to look at the conductor.

```
\markup { \eyeglasses }
```



`\first-visible` *args* (markup list)
: Use the first markup in *args* that yields a non-empty stencil and ignore the rest.

```
\markup {
  \first-visible {
    \fromproperty #'header:composer
    \italic Unknown
  }
}
```



`\footnote` *mkup* (markup) *note* (markup)
: Have footnote *note* act as an annotation to the markup *mkup*.

```
\markup {
  \footnote a b
  \override #'(padding . 0.2)
  \footnote c d
}
```

a c

b
d

The footnote will not be annotated automatically.

`\fraction` *arg1* (markup) *arg2* (markup)
Make a fraction of markups *arg1* and *arg2*.

```
\markup {
  π ≈ \fraction 355 113
}
```

$$\pi \approx \frac{355}{113}$$

Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
Read *symbol* from the property settings and produce a stencil from the markup contained within.

If *symbol* is not defined or is not a markup, return an empty markup.

Currently, the following properties can be accessed.

- Within a `\paper` block defining titles, headers, or footers, or within a `\header` block: all fields from the `\header` block (that produce markup) are available, with `header:` as a name prefix.
- Within a `\paper` block defining headers or footers: the current page number (symbol `page:page-number-string`).
- Within the `tocItemMarkup` paper variable (or in custom-made Scheme code that uses function `add-toc-item!`) defining a table of contents entry: the entry's text and page number are available as `toc:text` and `toc:page`, respectively. An entry's indentation markup is available as `toc:indent`.

```
\header {
  myTitle = "myTitle"
  title = \markup {
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
```

```
  \null
}
```

from *myTitle*

\keep-with-tag *tags* (symbol list or symbol) *arg* (markup)

Keep markup from *arg* that is untagged or tagged with *tags*.

All other parts of *arg* that are using a different tag are replaced with empty stencils. It works similar to \keepWithTag for music, but only with markups.

```
tagged = \markup {
  untagged
  \tag #'foo A
  \tag #'bar B
}

\markup { \keep-with-tag #'bar \tagged }
\markup { \keep-with-tag #'foo \tagged }
```

untagged B

untagged A

Used properties:

- tags-to-keep (())

\left-brace *size* (number)

Print a brace from the music font, of height *size* (in points).

```
\markup {
  \left-brace #35
  \hspace #2
  \left-brace #45
}
```



\lookup *glyph-name* (string)

Print a brace glyph with name *glyph-name*.

This is a historical command; \left-brace (which looks up the glyph by absolute size and is independent of the font size) is recommended instead.

```
\markup \lookup "brace200"
```



\markalphabet *num* (integer)

Make a markup letter for *num*.

The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

H Z

\markletter *num* (integer)
: Make a markup letter for *num*.

  The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

H AA

\null
: An empty markup with extents of a single point.

```
\markup {
  \null
}
```

\on-the-fly *procedure* (procedure) *arg* (markup)
: Apply the *procedure* markup command to *arg*.

  procedure takes the same arguments as `interpret-markup` and returns a stencil.

\override *new-prop* (pair) *arg* (markup)
: Add the argument *new-prop* to the property list for printing *arg*.

  In general, any property may be overridden that is part of `font-interface` (Section "font-interface" dans *Référence des propriétés internes*), `text-interface` (Section "text-interface" dans *Référence des propriétés internes*), or `instrument-specific-markup-interface` (Section "instrument-specific-markup-interface" dans *Référence des propriétés internes*). Additionally, various markup commands listen to other properties, too, as described in a markup function's documentation.

  new-prop is either a single alist pair or a non-empty list of alist pairs.

```
\markup {
  \undertie "undertied"
  \override #'(offset . 15)
  \undertie "offset undertied"
  \override #'((offset . 15) (thickness . 3))
  \undertie "offset thick undertied"
}
```

undertied offset undertied offset thick undertied

\page-link *page-number* (number) *arg* (markup)
: Add a link to a score's page *page-number* around *arg*.

  This only works in the PDF backend.

```
\markup {
  \page-link #2 { \italic { This links to page 2... } }
}
```

This links to page 2...

\page-ref *label* (symbol) *gauge* (markup) *default* (markup)
: Print a page number reference.

  label is the label set on the referenced page (using \label or \tocItem), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

  If the current book or book part is set to use roman numerals for page numbers, the reference will be formatted accordingly – in which case the *gauge*'s width may require additional tweaking.

  Used properties:

  - x-align (1)

\pattern *count* (non-negative, exact integer) *axis* (non-negative, exact integer) *space* (number) *pattern* (markup)
: Print a *pattern* markup *count* times.

  Patterns are spaced apart by *space* (defined as for \hspace or \vspace, respectively) and distributed on *axis*.

```
\markup \column {
  "Horizontally repeated:"
  \pattern #7 #X #2 \flat
  \null
  "Vertically repeated:"
  \pattern #3 #Y #0.5 \flat
}
```

Horizontally repeated:
♭ ♭ ♭ ♭ ♭ ♭ ♭

Vertically repeated:
♭
♭
♭

\property-recursive *symbol* (symbol)
: Print out a warning when header field markup in *symbol* contains some recursive markup definition.

\push-to-tag *tag* (symbol) *more* (markup) *arg* (markup)
: Prepend *more* to all markup in *arg* tagged with *tag*.

  It works similar to \pushToTag for music, but only with markups.

```
tagged = \markup {
  \tag #'foo A
  \tag #'bar B
}

\markup { \push-to-tag #'foo prefoo \tagged }
```

prefoo A B

Used properties:

- `tags-with-pushes-alist` (`()`)

`\qr-code` *width* (non-negative number) *str* (string)

Insert a QR code for string *str*, usually a URL, with a given *width*.

```
\markup \vcenter {
  \center-column { Engraved with LilyPond }
  \hspace #1.5
  \qr-code #10.0 "https://lilypond.org"
}
```

Engraved with LilyPond 

The `error-correction-level` property can be set to one of the symbols `low`, `medium`, `quarter`, or `high`. The higher the level of error correction is, the more the QR code contains redundancy, potentially helping detectors, e.g., in poor lighting conditions; however, a higher correction level also makes the code denser.

```
\markup \vcenter {
  \center-column { Engraved with LilyPond }
  \hspace #1.5
  \override #'(error-correction-level . high)
    \qr-code #10.0 "https://lilypond.org"
}
```

Engraved with LilyPond 

The `quiet-zone-size` property specifies the width of the « quiet zone », namely the white area around the QR code. It is expressed as a multiple of the width of one little square inside the QR code. Use at least 4 for best results.

Used properties:

- `quiet-zone-size` (`4`)
- `error-correction-level` (`low`)

`\remove-with-tag` *tags* (symbol list or symbol) *arg* (markup)

Remove markup from *arg* that is tagged with *tags*.

The removed markup is replaced with empty stencils. It works similar to `\removeWithTag` for music, but only with markups.

```
tagged = \markup {
  \tag #'foo A
  \tag #'bar B
}

\markup { \remove-with-tag #'foo \tagged }
\markup { \remove-with-tag #'bar \tagged }
```

B

A

Used properties:

- `tags-to-remove` (())

`\replace-with-tag` *tag* (symbol) *replacement* (markup) *arg* (markup)
Replace tagged markups in *arg*.

Everything tagged with *tag* (including the tagging itself) gets replaced with *replacement*.

It works similar to `\replaceWithTag` for music, but only with markups.

```
tagged = \markup {
  \tag #'foo A
  \tag #'bar B
}

\markup { \replace-with-tag #'foo C \tagged }
```

C B

Used properties:

- `tags-with-replacement-alist` (())

`\right-brace` *size* (number)
A music brace in point size *size*, rotated 180 degrees.

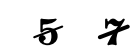
```
\markup {
  \right-brace #45
  \hspace #2
  \right-brace #35
}
```



`\slashed-digit` *num* (integer)
Print number *num* with the Emmentaler font, crossed through with a slash.

This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \slashed-digit #7
}
```



Used properties:

- `thickness` (`1.6`)
- `font-size` (`0`)

`\stencil` *stil* (stencil)
: Use stencil *stil* as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```



`\strut`
: Create a box of the same height as the space in the current font.

`\tag` *tags* (symbol list or symbol) *arg* (markup)
: Tag markup *arg* with *tag*.

tag can be one or multiple tags. This allows later on to reference *arg*; for example, to remove it or to add markup before or after the tagged markup. It works similar to `\tag` for music, but only with markups.

```
tagged = \markup {
  \tag #'foo A
  \tag #'bar B
}

\markup { \keep-with-tag #'bar \tagged }
\markup { \keep-with-tag #'foo \tagged }
```





Used properties:

- `tags-with-replacement-alist` (`()`)
- `tags-with-appends-alist` (`()`)
- `tags-with-pushes-alist` (`()`)
- `tags-to-remove` (`()`)
- `tags-to-keep` (`()`)

`\transparent` *arg* (markup)
: Make *arg* transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

\verbatim-file *name* (string)
: Read the contents of file *name* and include it verbatim.

```
\markup {
  \verbatim-file "en/included/simple.ly"
}
```

```
% A simple piece in LilyPond, a scale.
\version "2.19.21"
\relative {
  c' d e f g a b c
}
```

Use `\withRelativeDir` as a prefix to *name* if the file should be found relative to the input file.

\whiteout *arg* (markup)
: Provide a white background for *arg*.

The shape of the white background is determined by the `style` property. The default is `box` which produces a rectangle. `rounded-box` produces a rounded rectangle, and `outline` approximates the outline of the markup.

The color of the background can be controlled with the `color` property, defaulting to `"white"`.

```
\markup {
  \combine
    \filled-box #'(-1 . 62) #'(-3 . 4) #1
    \override #'(line-width . 60)
      \fill-line {
        \override #'(thickness . 1.5)
          \whiteout box
        \override #'((style . rounded-box) (thickness . 3))
          \whiteout rounded-box
        \override #'((style . outline) (thickness . 3))
          \whiteout outline
        \override #'((color . "red") (style . outline))
          \whiteout red-outline
      }
}
```



Used properties:

- `color` (`"white"`)
- `thickness` (`()`)

- `style` (box)

`\with-color` *col* (color) *arg* (markup)
Use color *col* to draw *arg*.

Voir Section 7.1.5 [Coloration d'objets], page 298, for valid color specifications.

```
\markup {
  \with-color #red red
  \hspace #2
  \with-color #green green
  \hspace #2
  \with-color "#0000ff" blue
}
```

red green blue

`\with-dimension` *axis* (integer) *val* (pair of numbers) *arg* (markup)
Set the dimension of *arg* along *axis* to *val*.

If *axis* is equal to `X`, set the horizontal dimension. If *axis* is equal to `Y`, set the vertical dimension.

`\with-dimension-from` *axis* (integer) *arg1* (markup) *arg2* (markup)
Print *arg2* but replace the dimension along *axis* with the one from *arg1*.

If *axis* is set to `X`, replace the horizontal dimension. If *axis* is set to `Y`, replace the vertical dimension.

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
Set the horizontal and vertical dimensions of *arg* to *x* and *y*.

`\with-dimensions-from` *arg1* (markup) *arg2* (markup)
Print *arg2* with the horizontal and vertical dimensions of *arg1*.

`\with-link` *label* (symbol) *arg* (markup)
Add a link to the page holding label *label* around *arg*.

This only works in the PDF backend.

```
\markup {
  \with-link #'label {
    \italic { This links to the page
              containing the label... }
  }
}
```

`\with-outline` *outline* (markup) *arg* (markup)
Print *arg* with the outline and dimensions of *outline*.

The outline is used by skylines to resolve collisions (not for whiteout).

`\with-true-dimension` *axis* (integer) *arg* (markup)
Give *arg* its actual dimension (extent) on *axis*.

Sometimes, the extents of a markup's printed ink differs from the default extents. The main case is if glyphs are involved. By default, the extents of a glyph are based on the glyph's *metrics* (i.e., a default vertical and horizontal size for the glyph), which, for various reasons, are often not identical to its *bounding box* (i.e., the smallest rectangle that completely encompasses the glyph's outline) – in most cases, the outline protrudes the box spanned up by the metrics.

```
\markup {
  text
  \fontsize #10
  \override #'((box-padding . 0) (thickness . 0.2))
  \box
    \musicglyph "scripts.trill"
  text
}
```



For purposes other than setting text, this behavior may not be wanted. You can use `\with-true-dimension` in order to give the markup its actual printed extent.

```
\markup {
  text
  \fontsize #10
  \override #'((box-padding . 0) (thickness . 0.2))
  \box
    \with-true-dimension #X
    \musicglyph "scripts.trill"
  text
}
```



`\with-true-dimensions` *arg* (markup)
: Give *arg* its actual dimensions (extents).

  Calling

  ```
  \markup \with-true-dimensions arg
  ```

  is short for

  ```
  \markup
    \with-true-dimension #X
    \with-true-dimension #Y
    arg
  ```

  i.e., `\with-true-dimensions` has the effect of `\with-true-dimension` on both axes.

A.2 Commandes pour liste de *markups*

Les commandes suivantes peuvent être utilisées dans un bloc `\markuplist { }`.

`\column-lines` *args* (markup list)
: Stack the markups in *args* vertically.

  Like `\column`, but return a list of lines instead of a single markup. The property `baseline-skip` determines the space between each markup in *args*.

  Used properties:

  - `baseline-skip`

`\justified-lines` *args* (markup list)
: Print *args* as lines aligned both at the left and the right.

Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

Used properties:

- `text-direction` (`1`)
- `word-space`
- `line-width` (`#f`)
- `baseline-skip`

`\override-lines` *new-prop* (pair) *args* (markup list)

Add the argument *new-prop* to the property list for printing *args*.

Like `\override` but for markup lists.

`\score-lines` *score* (score)

Inline an image of music as specified by *score*.

Like `\score` but return a list of lines instead of a single markup.

Used properties:

- `tags-with-replacement-alist` (`()`)
- `tags-with-appends-alist` (`()`)
- `tags-with-pushes-alist` (`()`)
- `tags-to-remove` (`()`)
- `tags-to-keep` (`()`)

`\string-lines` *str* (string)

Split string *str* into lines.

The character to split at is specified by the property `split-char`, defaulting to `#\newline`. Surrounding whitespace is removed from every resulting string. The returned list of markups is ready to be formatted by other markup or markup list commands like `\column`, `\line`, etc.

```
\markup {
  \column
    \string-lines
     "foo, foo,
     bar, bar,
     buzz, buzz!"
}
```

foo, foo,
bar, bar,
buzz, buzz!

Used properties:

- `split-char` (`#\newline`)

`\table` *column-align* (number list) *lst* (markup list)

Print a table.

column-align specifies how each column is aligned; possible values are -1, 0, and 1. The number of elements in *column-align* determines how many columns will be printed.

The entries to print are given by *lst*, a markup list. If needed, the last row is filled up with `point-stencils`.

Override the padding property to increase the horizontal distance between columns. Override baseline-skip to increase the vertical distance between rows.

```
% A markup command to print a fixed-width number.
\markup fwnum =
  \markup \override #'(font-features . ("ss01" "-kern"))
    \number \etc

\markuplist {
  \override #'(padding . 2)
  \table #'(0 1 0 -1) {
    \underline { center-aligned right-aligned
                 center-aligned left-aligned }
    one      \fwnum    1 thousandth \fwnum 0.001
    eleven   \fwnum   11 hundredth  \fwnum 0.01
    twenty   \fwnum   20 tenth      \fwnum 0.1
    thousand \fwnum 1000 one        \fwnum 1.0
  }
}
```

center-aligned	right-aligned	center-aligned	left-aligned
one	**1**	thousandth	**0.001**
eleven	**11**	hundredth	**0.01**
twenty	**20**	tenth	**0.1**
thousand	**1000**	one	**1.0**

Used properties:

- baseline-skip
- padding (0)

\table-of-contents

Print a table of contents.

This function uses the paper variable tocTitleMarkup for the title; it then prints \tocItem entries line by line.

Voir Section 21.7 [Table des matières], page 641, for a complete discussion.

Used properties:

- baseline-skip

\tag-list *tags* (symbol list or symbol) *arg* (markup list)

Tag markup list *arg* with *tag*.

tag can be one or multiple tags. This allows later on to reference *arg*; for example, to remove it or to add markup before or after the tagged markup list.

It works like the \tag command for markups but with markup lists. You will need it if you have to reference a whole list; for example, to use \push-to-tag and \append-to-tag without pushing or appending before or after every single item of the list, but before or after the whole list instead.

```
tagged = \markuplist {
  \tag-list #'foo { foo bar }
}

\markup { \push-to-tag #'foo test \tagged }
```

test foo bar

Used properties:

- `tags-with-replacement-alist` (`()`)
- `tags-with-appends-alist` (`()`)
- `tags-with-pushes-alist` (`()`)
- `tags-to-remove` (`()`)
- `tags-to-keep` (`()`)

`\wordwrap-lines` *args* (markup list)
Print *args* as left-aligned lines.

Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where X is the number of staff spaces.

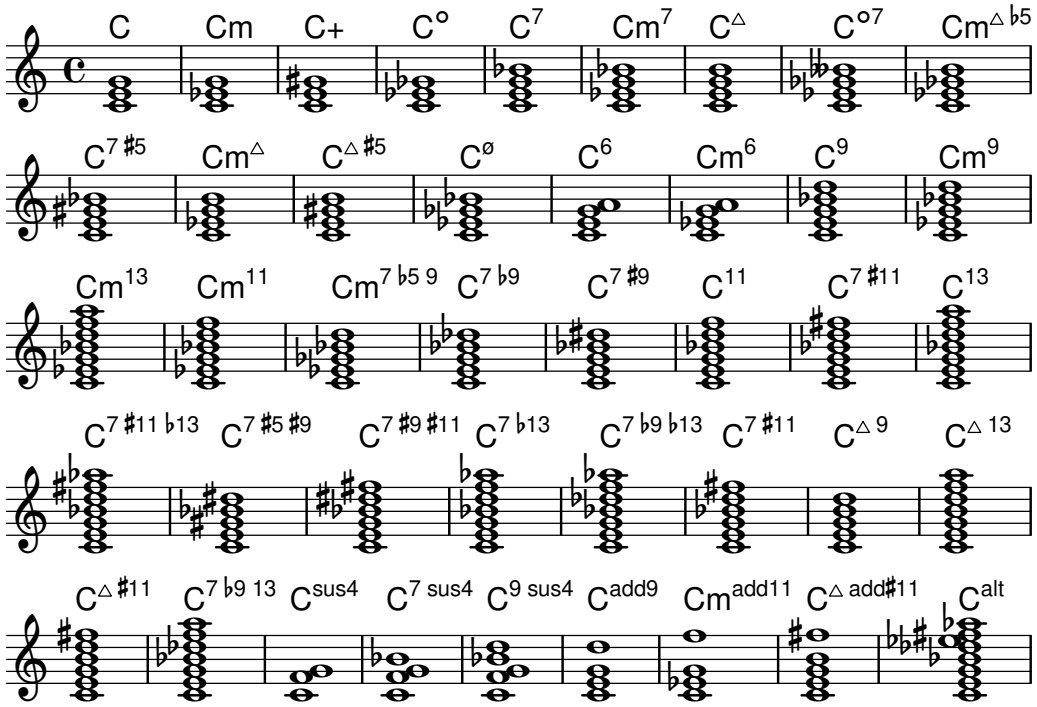
Used properties:

- `text-direction` (`1`)
- `word-space`
- `line-width` (`#f`)
- `baseline-skip`

Annexe B Tables du manuel de notation

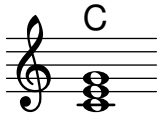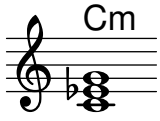
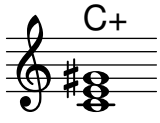
B.1 Table des noms d'accord
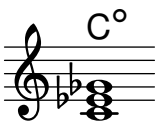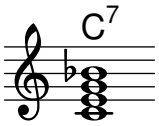
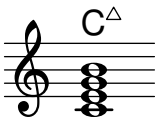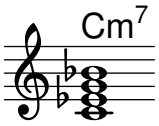
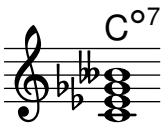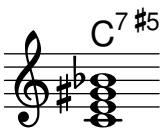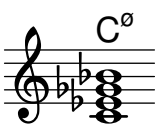
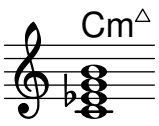
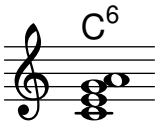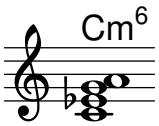
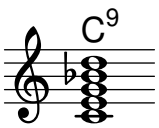
La liste suivante répertorie les noms des accords selon la dénomination standard de LilyPond ainsi que ce qu'ils représentent. D'autres systèmes de nommage, non pris en charge, sont indiqués dans le morceau choisi « Noms d'accords alternatifs » (*chord-names-alternative*) dans Section "Accords" dans *Morceaux choisis*, qui inclut une notation inspirée par Harald Banter (1982) utilisée par défaut dans LilyPond jusqu'à sa version 1.7.

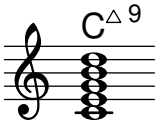
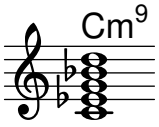
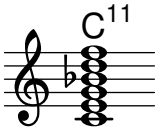
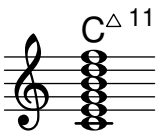
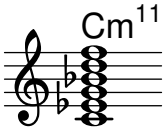
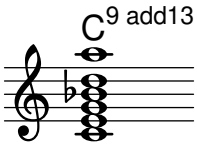
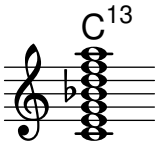
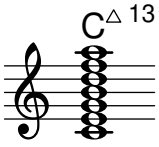
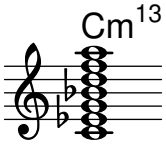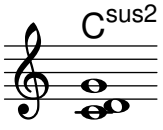


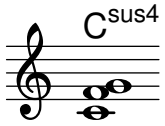
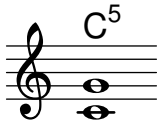
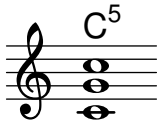
B.2 Modificateurs d'accord usuels

Le tableau suivant indique les différents modificateurs qui permettent d'obtenir les structures habituelles d'un certain nombre d'accords.

Type	**Intervalle**	**Modificateur**	**Exemple**	**Résultat**
Accord parfait majeur (triade majeure)	Tierce majeure et quinte juste	(rien)	`c1`	
Accord parfait mineur (triade mineure)	Tierce mineure et quinte juste	`m` ou `m5`	`c1:m`	
Triade augmentée	Tierce majeure et quinte augmentée	`aug`	`c1:aug`	

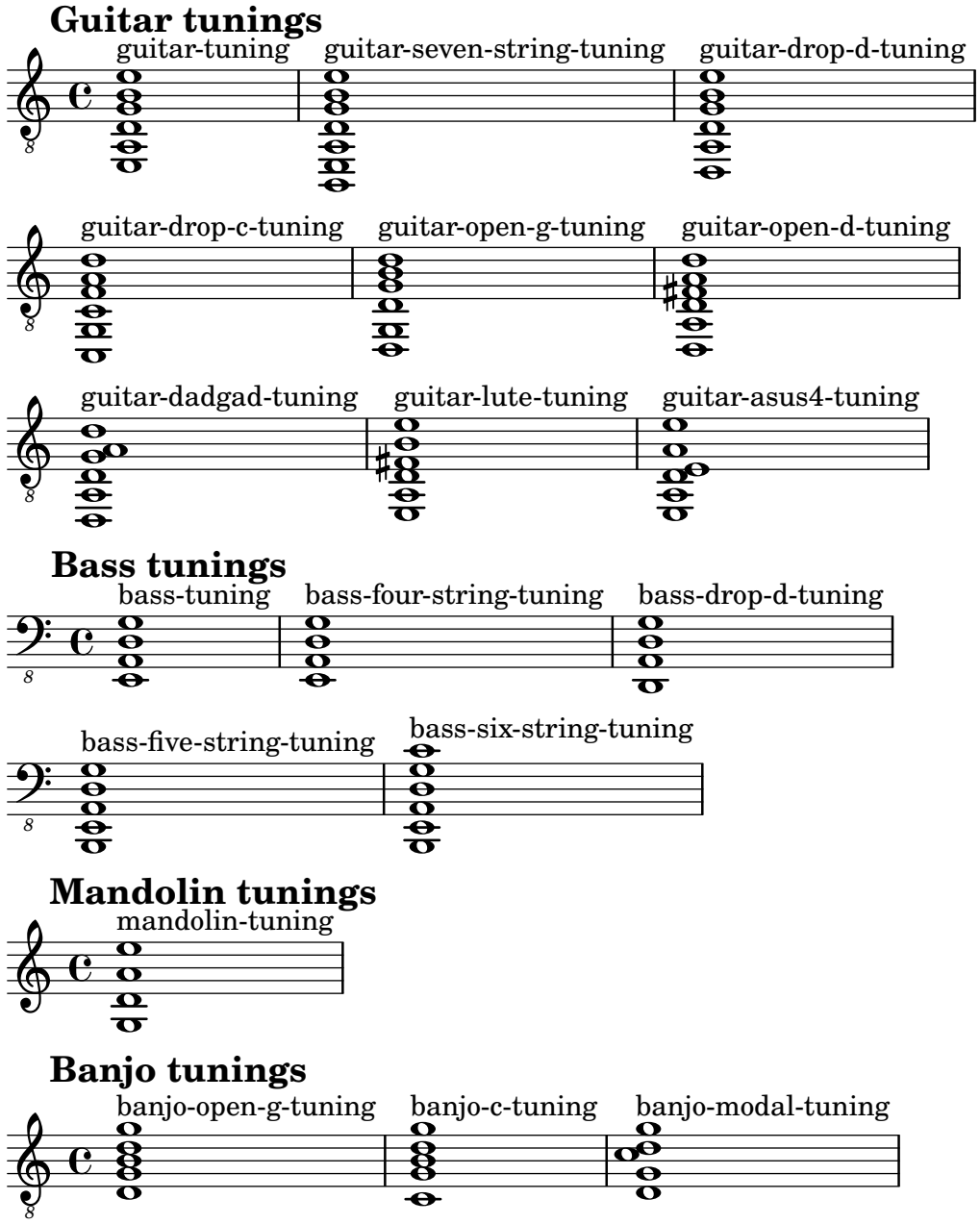
Triade diminuée	Tierce mineure et quinte diminuée	`dim`	`c1:dim`	
Septième de dominante	Triade majeure et septième mineure	`7`	`c1:7`	
Septième majeure	Triade majeure et septième majeure	`maj7` ou `maj`	`c1:maj7`	
Septième mineure	Triade mineure et septième mineure	`m7`	`c1:m7`	
Septième diminuée	Triade diminuée et septième diminuée	`dim7`	`c1:dim7`	
Septième augmentée	Triade augmentée et septième mineure	`aug7`	`c1:aug7`	
Septième semi-diminuée	Triade diminuée et septième mineure	`m7.5-`	`c1:m7.5-`	
Accord mineur avec septième majeure	Triade mineure et septième majeure	`m7+`	`c1:m7+`	
Sixte ajoutée (majeur)	Triade majeure et sixte majeure	`6`	`c1:6`	
Sixte ajoutée (mineur)	Triade mineure et sixte majeure	`m6`	`c1:m6`	
Neuvième de dominante	Septième de dominante et neuvième majeure	`9`	`c1:9`	

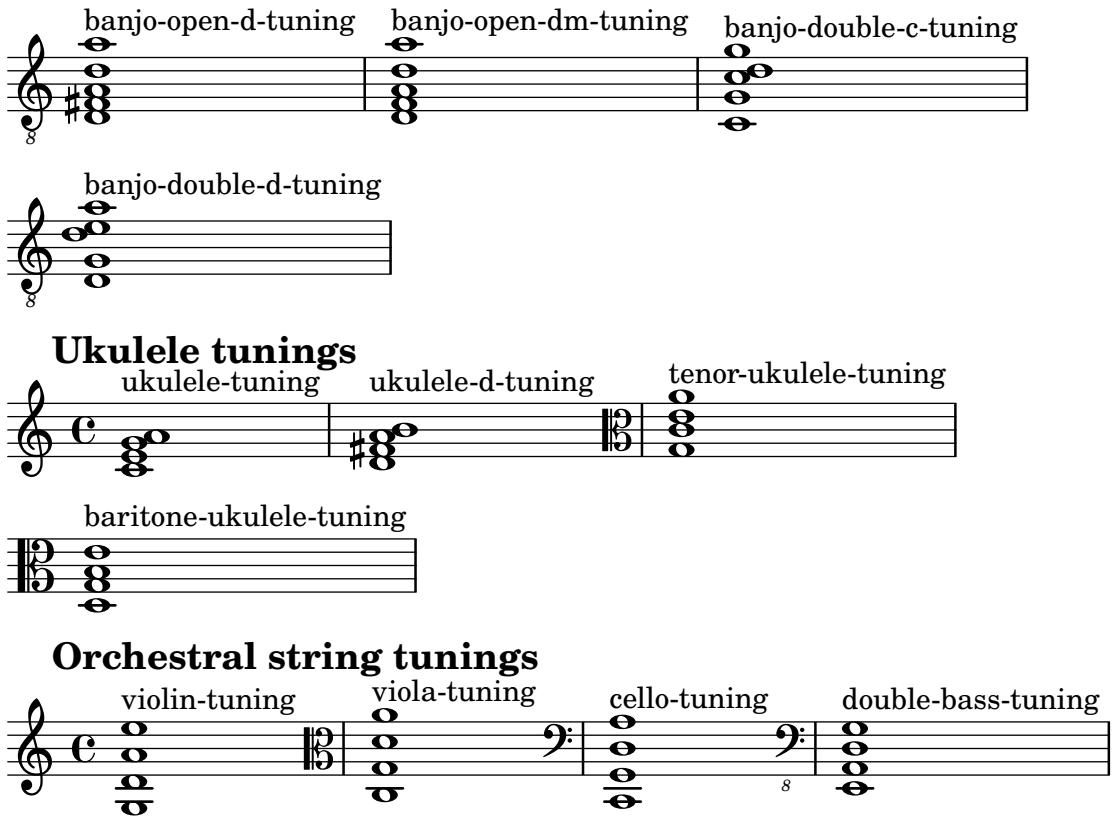
Neuvième (majeur)	Septième majeure et neuvième majeure	`maj9`	`c1:maj9`	
Neuvième (mineur)	Septième mineure et neuvième majeure	`m9`	`c1:m9`	
Onzième de dominante	Neuvième de dominante et onzième juste	`11`	`c1:11`	
Onzième (majeur)	Septième majeure, neuvième majeure et onzième juste	`maj11`	`c1:maj11`	
Onzième (mineur)	Septième mineure, neuvième majeure et onzième juste	`m11`	`c1:m11`	
Treizième de dominante	Neuvième de dominante et treizième majeure	`13`	`c1:13`	
Treizième de dominante	Onzième de dominante et treizième majeure	`13.11`	`c1:13.11`	
Treizième (majeur)	Septième majeure, neuvième, onzième et treizième majeures	`maj13.11`	`c1:maj13.11`	
Treizième (mineur)	Septième mineure, neuvième, onzième et treizième majeures	`m13.11`	`c1:m13.11`	
Seconde suspendue	Seconde majeure et quinte juste	`sus2`	`c1:sus2`	

Quarte suspendue	Quarte juste et quinte juste	`sus4`	`c1:sus4`	
Power chord (deux voix)	Quinte juste	`1.5`	`c1:5`	
Power chord (trois voix)	Quinte juste et octave	`1.5.8`	`c1:5.8`	

B.3 Accordages prédéfinis

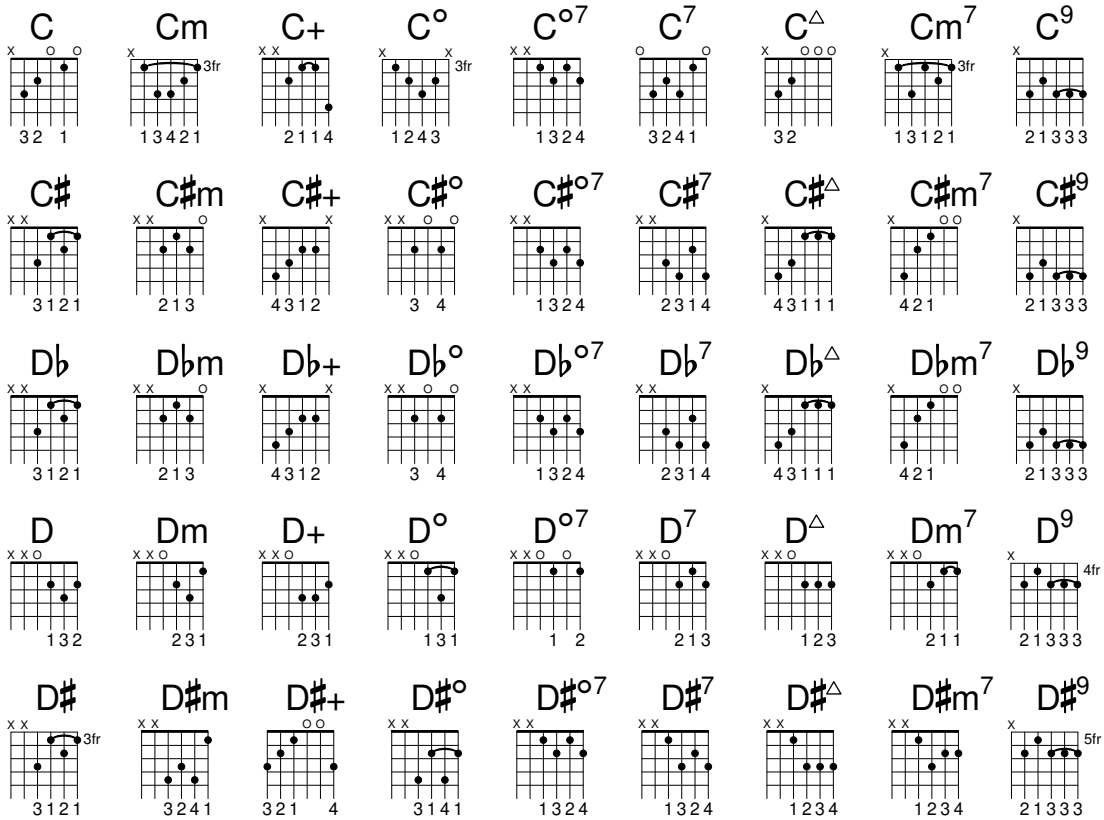
La liste suivante répertorie les différents accordages dont LilyPond dispose.

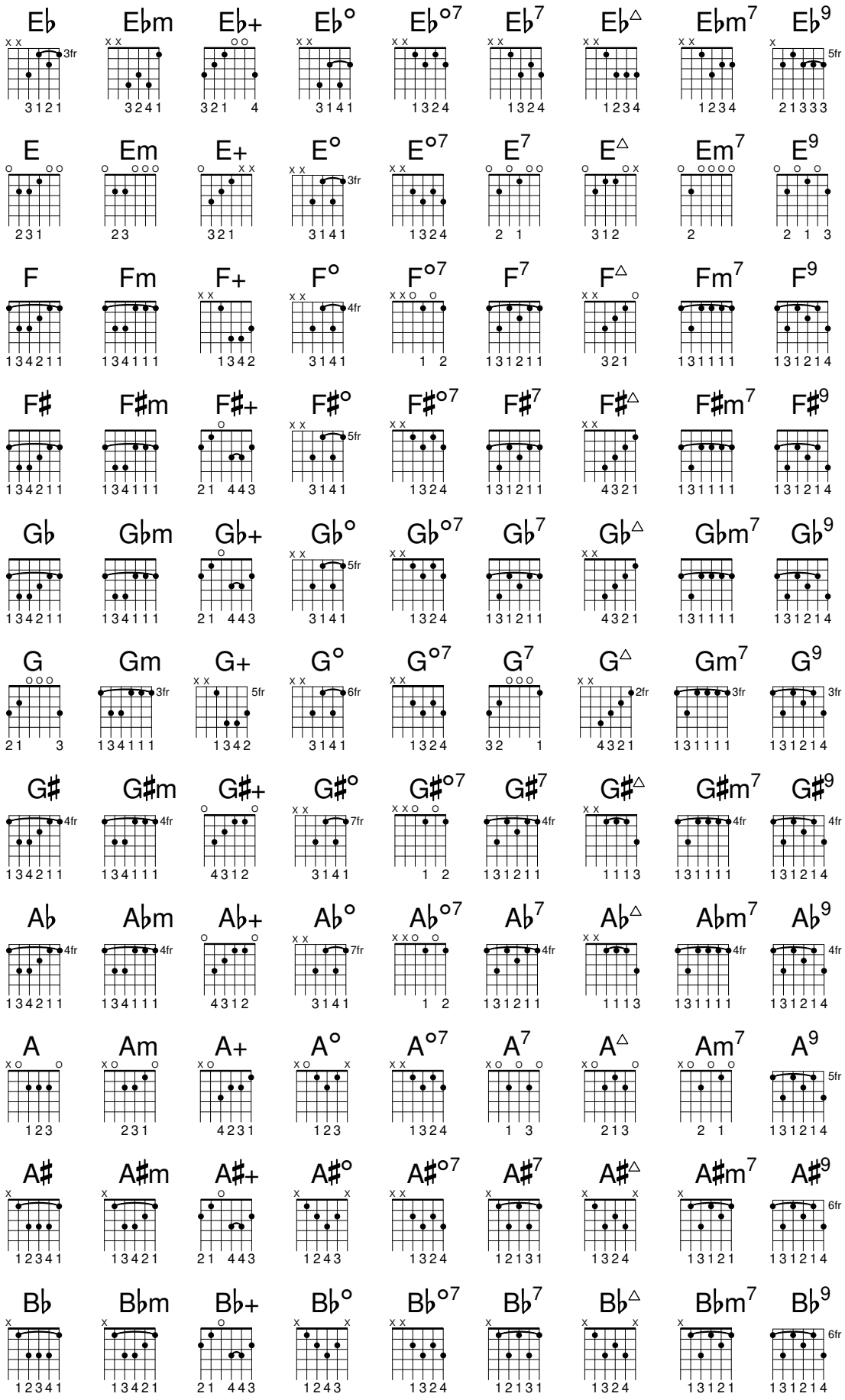


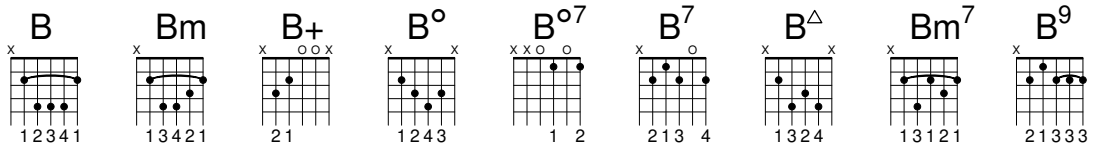


B.4 Diagrammes d'accord prédéfinis

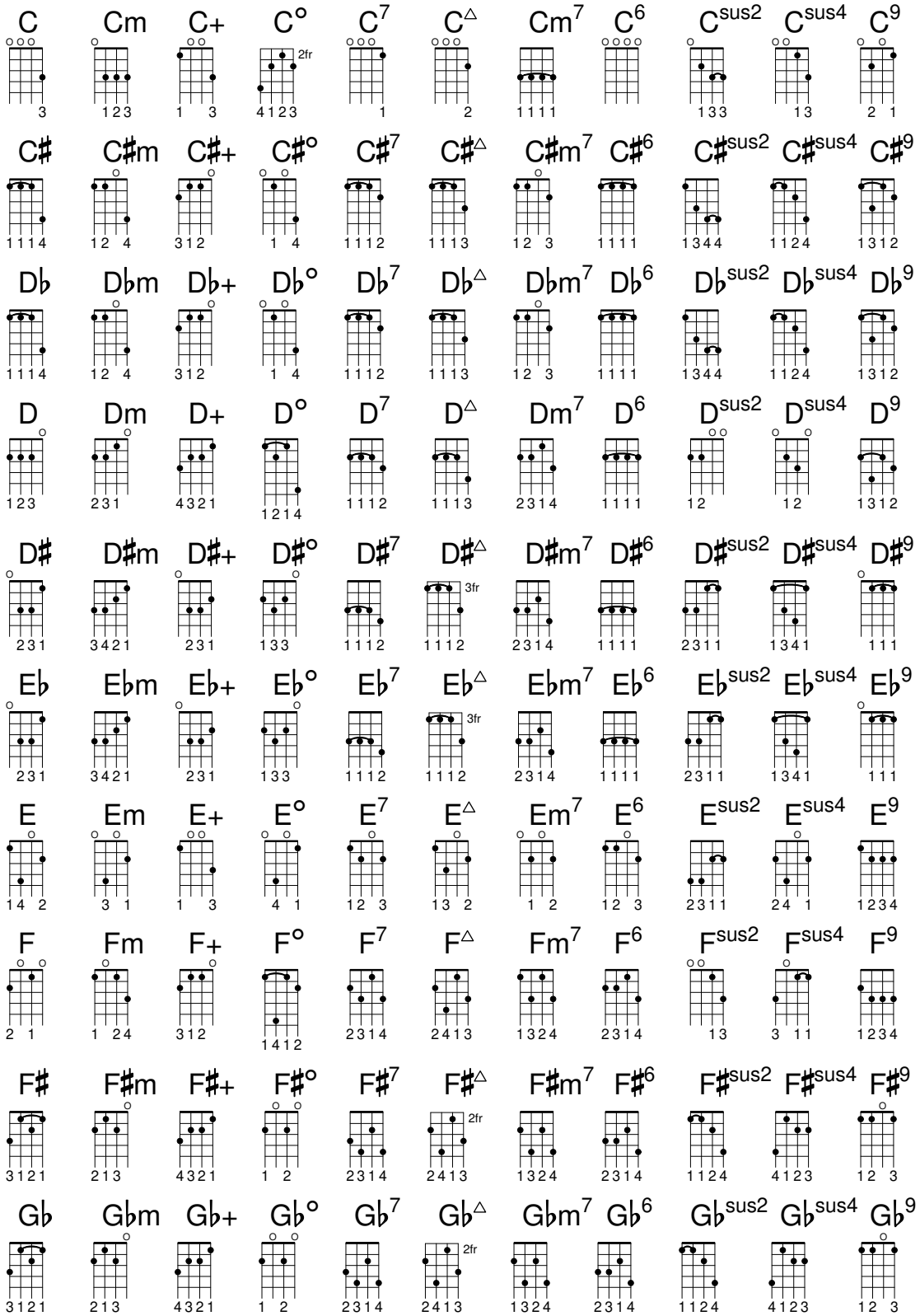
B.4.1 Diagrammes pour guitare

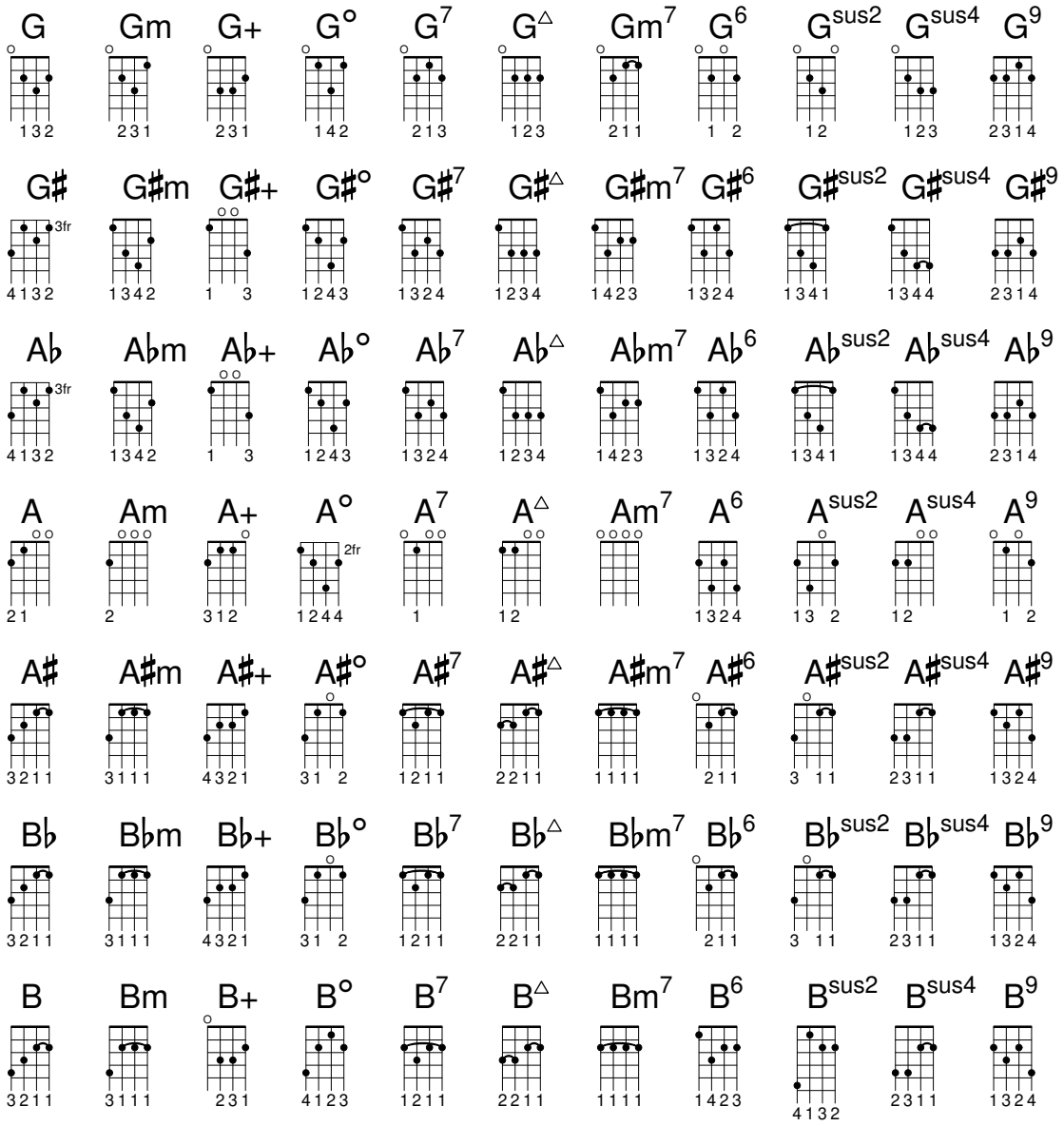



E♭ E♭m E♭+ E♭° E♭°7 E♭7 E♭△ E♭m7 E♭9
E Em E+ E° E°7 E7 E△ Em7 E9
F Fm F+ F° F°7 F7 F△ Fm7 F9
F♯ F♯m F♯+ F♯° F♯°7 F♯7 F♯△ F♯m7 F♯9
G♭ G♭m G♭+ G♭° G♭°7 G♭7 G♭△ G♭m7 G♭9
G Gm G+ G° G°7 G7 G△ Gm7 G9
G♯ G♯m G♯+ G♯° G♯°7 G♯7 G♯△ G♯m7 G♯9
A♭ A♭m A♭+ A♭° A♭°7 A♭7 A♭△ A♭m7 A♭9
A Am A+ A° A°7 A7 A△ Am7 A9
A♯ A♯m A♯+ A♯° A♯°7 A♯7 A♯△ A♯m7 A♯9
B♭ B♭m B♭+ B♭° B♭°7 B♭7 B♭△ B♭m7 B♭9

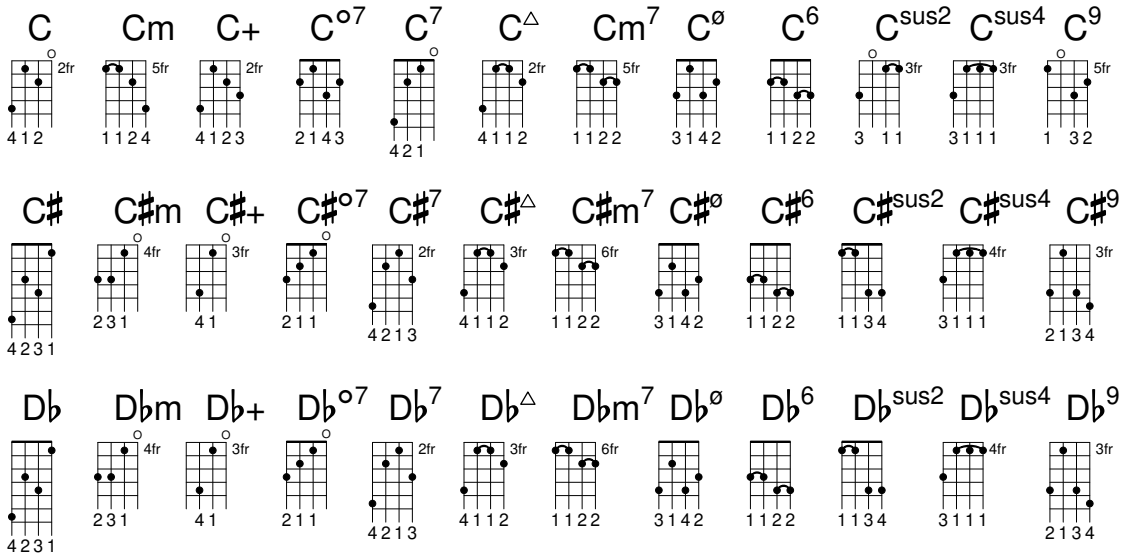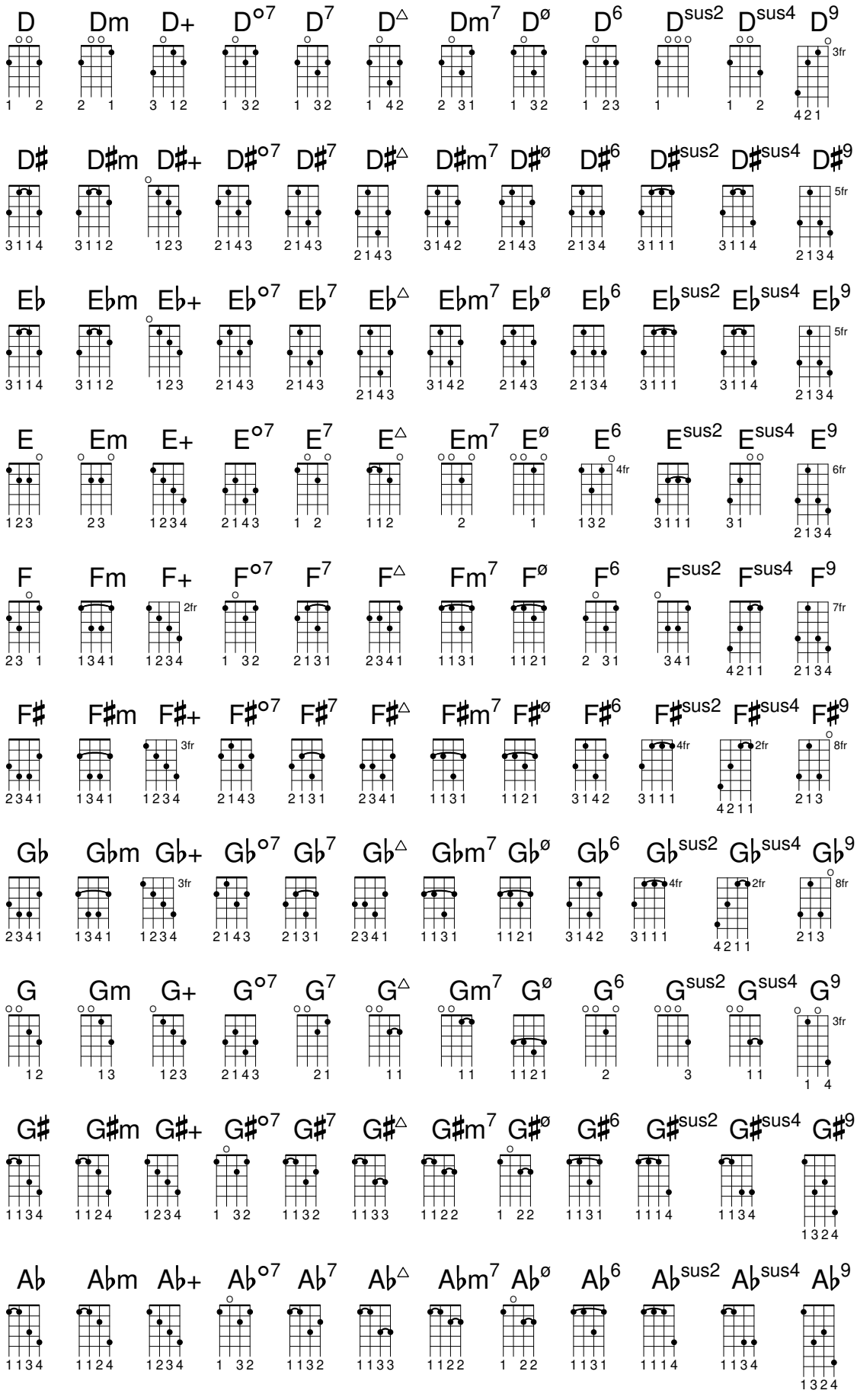


B.4.2 Diagrammes pour ukulele

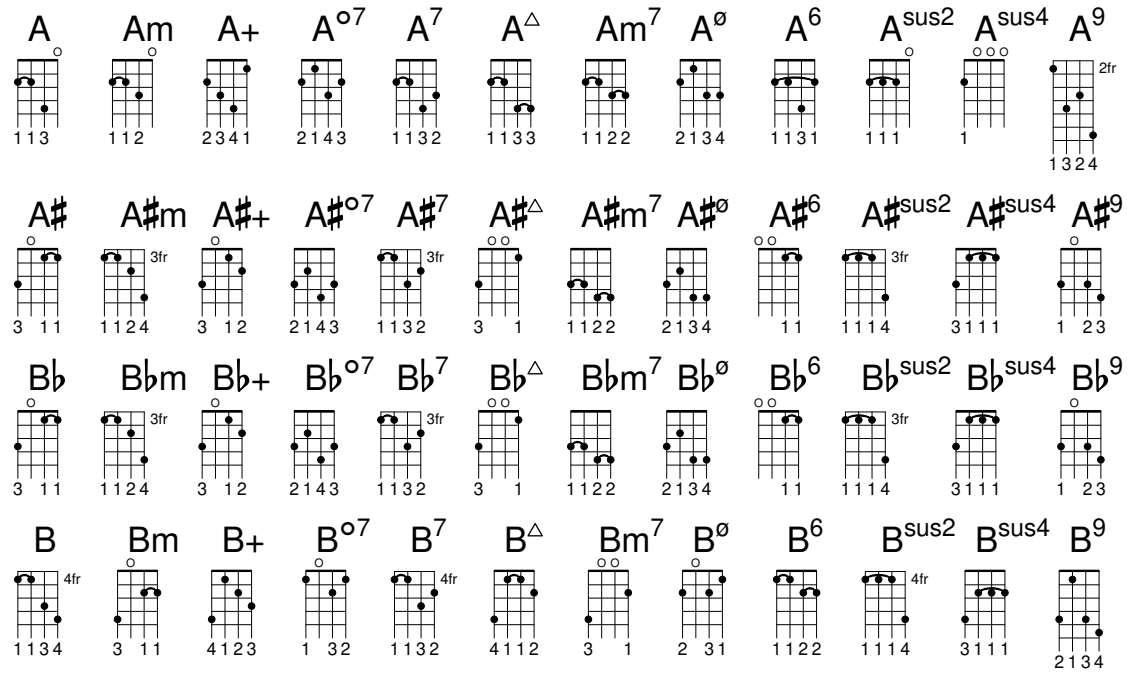




B.4.3 Diagrammes pour mandoline




D Dm D+ D°7 D7 D△ Dm7 Dø D6 Dsus2 Dsus4 D9
D♯ D♯m D♯+ D♯°7 D♯7 D♯△ D♯m7 D♯ø D♯6 D♯sus2 D♯sus4 D♯9
E♭ E♭m E♭+ E♭°7 E♭7 E♭△ E♭m7 E♭ø E♭6 E♭sus2 E♭sus4 E♭9
E Em E+ E°7 E7 E△ Em7 Eø E6 Esus2 Esus4 E9
F Fm F+ F°7 F7 F△ Fm7 Fø F6 Fsus2 Fsus4 F9
F♯ F♯m F♯+ F♯°7 F♯7 F♯△ F♯m7 F♯ø F♯6 F♯sus2 F♯sus4 F♯9
G♭ G♭m G♭+ G♭°7 G♭7 G♭△ G♭m7 G♭ø G♭6 G♭sus2 G♭sus4 G♭9
G Gm G+ G°7 G7 G△ Gm7 Gø G6 Gsus2 Gsus4 G9
G♯ G♯m G♯+ G♯°7 G♯7 G♯△ G♯m7 G♯ø G♯6 G♯sus2 G♯sus4 G♯9
A♭ A♭m A♭+ A♭°7 A♭7 A♭△ A♭m7 A♭ø A♭6 A♭sus2 A♭sus4 A♭9



B.5 Formats de papier prédéfinis

Les formats de page sont définis dans le fichier `scm/paper.scm`.

La série A « ISO 216 »

`"a10"`	(26 x 37 mm)
`"a9"`	(37 x 52 mm)
`"a8"`	(52 x 74 mm)
`"a7"`	(74 x 105 mm)
`"a6"`	(105 x 148 mm)
`"a5"`	(148 x 210 mm)
`"a4"`	(210 x 297 mm)
`"a3"`	(297 x 420 mm)
`"a2"`	(420 x 594 mm)
`"a1"`	(594 x 841 mm)
`"a0"`	(841 x 1189 mm)

Deux tailles étendues, définies par la « DIN 476 »

`"4a0"`	(1682 x 2378 mm)
`"2a0"`	(1189 x 1682 mm)

La série B « ISO 216 »

`"b10"`	(31 x 44 mm)
`"b9"`	(44 x 62 mm)
`"b8"`	(62 x 88 mm)
`"b7"`	(88 x 125 mm)
`"b6"`	(125 x 176 mm)
`"b5"`	(176 x 250 mm)
`"b4"`	(250 x 353 mm)
`"b3"`	(353 x 500 mm)
`"b2"`	(500 x 707 mm)
`"b1"`	(707 x 1000 mm)
`"b0"`	(1000 x 1414 mm)

La série C standard « ISO 269 »

`"c10"`	(28 x 40 mm)
`"c9"`	(40 x 57 mm)
`"c8"`	(57 x 81 mm)
`"c7"`	(81 x 114 mm)
`"c6"`	(114 x 162 mm)
`"c5"`	(162 x 229 mm)
`"c4"`	(229 x 324 mm)
`"c3"`	(324 x 458 mm)
`"c2"`	(458 x 648 mm)
`"c1"`	(648 x 917 mm)
`"c0"`	(917 x 1297 mm)

Formats nord américains

`"junior-legal"`	(8.0 x 5.0 in)
`"legal"`	(8.5 x 14.0 in)
`"ledger"`	(17.0 x 11.0 in)
`"letter"`	(8.5 x 11.0 in)
`"tabloid"`	(11.0 x 17.0 in)
`"11x17"`	(11.0 x 17.0 in)
`"17x11"`	(17.0 x 11.0 in)

Government-letter, défini par le *IEEE Printer Working Group*, à l'usage des enfants

`"government-letter"`	(8 x 10.5 in)
`"government-legal"`	(8.5 x 13.0 in)
`"philippine-legal"`	(8.5 x 13.0 in)

Formats ANSI

`"ansi a"`	(8.5 x 11.0 in)
`"ansi b"`	(17.0 x 11.0 in)
`"ansi c"`	(17.0 x 22.0 in)
`"ansi d"`	(22.0 x 34.0 in)
`"ansi e"`	(34.0 x 44.0 in)
`"engineering f"`	(28.0 x 40.0 in)

Formats nord américains pour l'architecture

`"arch a"`	(9.0 x 12.0 in)
`"arch b"`	(12.0 x 18.0 in)
`"arch c"`	(18.0 x 24.0 in)
`"arch d"`	(24.0 x 36.0 in)
`"arch e"`	(36.0 x 48.0 in)
`"arch e1"`	(30.0 x 42.0 in)

Formats anciens, toujours en vigueur au Royaume Uni

`"statement"`	(5.5 x 8.5 in)
`"half letter"`	(5.5 x 8.5 in)
`"quarto"`	(8.0 x 10.0 in)
`"octavo"`	(6.75 x 10.5 in)
`"executive"`	(7.25 x 10.5 in)
`"monarch"`	(7.25 x 10.5 in)
`"foolscap"`	(8.27 x 13.0 in)
`"folio"`	(8.27 x 13.0 in)
`"super-b"`	(13.0 x 19.0 in)
`"post"`	(15.5 x 19.5 in)
`"crown"`	(15.0 x 20.0 in)
`"large post"`	(16.5 x 21.0 in)

"demy"	(17.5 x 22.5 in)
"medium"	(18.0 x 23.0 in)
"broadsheet"	(18.0 x 24.0 in)
"royal"	(20.0 x 25.0 in)
"elephant"	(23.0 x 28.0 in)
"double demy"	(22.5 x 35.0 in)
"quad demy"	(35.0 x 45.0 in)
"atlas"	(26.0 x 34.0 in)
"imperial"	(22.0 x 30.0 in)
"antiquarian"	(31.0 x 53.0 in)

Formats de base PA4

"pa0"	(840 x 1120 mm)
"pa1"	(560 x 840 mm)
"pa2"	(420 x 560 mm)
"pa3"	(280 x 420 mm)
"pa4"	(210 x 280 mm)
"pa5"	(140 x 210 mm)
"pa6"	(105 x 140 mm)
"pa7"	(70 x 105 mm)
"pa8"	(52 x 70 mm)
"pa9"	(35 x 52 mm)
"pa10"	(26 x 35 mm)

Format utilisé en Asie du Sudest et en Australie

"f4"	(210 x 330 mm)

Format spécifique aux courts exemples `@lilypond` de la documentation, basé sur un A8 à l'italienne.

"a8landscape"	(74 x 52 mm)

B.6 Instruments MIDI

La liste suivante répertorie les différentes dénominations que vous pouvez affecter à la propriété `midiInstrument`. Est affecté à chaque instrument son numéro dans l'ordre des 128 programmes du standard *General MIDI*.

1	acoustic grand	2	bright acoustic	3	electric grand
4	honky-tonk	5	electric piano 1	6	electric piano 2
7	harpsichord	8	clav	9	celesta
10	glockenspiel	11	music box	12	vibraphone
13	marimba	14	xylophone	15	tubular bells
16	dulcimer	17	drawbar organ	18	percussive organ
19	rock organ	20	church organ	21	reed organ
22	accordion	23	harmonica	24	concertina
25	acoustic guitar (nylon)	26	acoustic guitar (steel)	27	electric guitar (jazz)
28	electric guitar (clean)	29	electric guitar (muted)	30	overdriven guitar
31	distorted guitar	32	guitar harmonics	33	acoustic bass
34	electric bass (finger)	35	electric bass (pick)	36	fretless bass
37	slap bass 1	38	slap bass 2	39	synth bass 1
40	synth bass 2	41	violin	42	viola
43	cello	44	contrabass	45	tremolo strings
46	pizzicato strings	47	orchestral harp	48	timpani
49	string ensemble 1	50	string ensemble 2	51	synthstrings 1
52	synthstrings 2	53	choir aahs	54	voice oohs
55	synth voice	56	orchestra hit	57	trumpet
58	trombone	59	tuba	60	muted trumpet
61	french horn	62	brass section	63	synthbrass 1
64	synthbrass 2	65	soprano sax	66	alto sax

67	tenor sax	68	baritone sax	69	oboe
70	english horn	71	bassoon	72	clarinet
73	piccolo	74	flute	75	recorder
76	pan flute	77	blown bottle	78	shakuhachi
79	whistle	80	ocarina	81	lead 1 (square)
82	lead 2 (sawtooth)	83	lead 3 (calliope)	84	lead 4 (chiff)
85	lead 5 (charang)	86	lead 6 (voice)	87	lead 7 (fifths)
88	lead 8 (bass+lead)	89	pad 1 (new age)	90	pad 2 (warm)
91	pad 3 (polysynth)	92	pad 4 (choir)	93	pad 5 (bowed)
94	pad 6 (metallic)	95	pad 7 (halo)	96	pad 8 (sweep)
97	fx 1 (rain)	98	fx 2 (soundtrack)	99	fx 3 (crystal)
100	fx 4 (atmosphere)	101	fx 5 (brightness)	102	fx 6 (goblins)
103	fx 7 (echoes)	104	fx 8 (sci-fi)	105	sitar
106	banjo	107	shamisen	108	koto
109	kalimba	110	bagpipe	111	fiddle
112	shanai	113	tinkle bell	114	agogo
115	steel drums	116	woodblock	117	taiko drum
118	melodic tom	119	synth drum	120	reverse cymbal
121	guitar fret noise	122	breath noise	123	seashore
124	bird tweet	125	telephone ring	126	helicopter
127	applause	128	gunshot		

Certains kits de percussions sont aussi disponibles.

1	standard kit	9	room kit	17	power kit
25	electronic kit	26	tr-808 kit	33	jazz kit
41	brush kit	49	orchestra kit	57	sfx kit
128	mt-32 kit/cm-64 kit				

B.7 Liste des couleurs

Couleurs de base

La syntaxe appropriée à la gestion des couleurs est traitée au chapitre Section 7.1.5 [Coloration d'objets], page 298.

black	white	red	green
blue	cyan	magenta	yellow
grey	darkred	darkgreen	darkblue
darkcyan	darkmagenta	darkyellow	

Noms de couleur CSS

Les noms de couleur CSS peuvent s'utiliser tels quels en argument textuel.

aliceblue	antiquewhite	aqua
aquamarine	azure	beige
bisque	black	blanchedalmond
blue	blueviolet	brown
burlywood	cadetblue	chartreuse
chocolate	coral	cornflowerblue
cornsilk	crimson	cyan
darkblue	darkcyan	darkgoldenrod
darkgray	darkgreen	darkgrey
darkkhaki	darkmagenta	darkolivegreen
darkorange	darkorchid	darkred
darksalmon	darkseagreen	darkslateblue
darkslategray	darkslategrey	darkturquoise
darkviolet	deeppink	deepskyblue
dimgray	dimgrey	dodgerblue
firebrick	floralwhite	forestgreen



La définition CSS diffère des noms de couleur X pour les couleurs suivantes : `green`, `grey`, `maroon`, `purple`.

Noms de couleur X

Les noms de couleur X 11 (`https://en.wikipedia.org/wiki/X11_color_names`) offrent un choix plus large que les noms CSS. Ils peuvent s'employer de différentes manières.

- Un nom de couleur peut s'écrire sous la forme d'un mot composé et sans espace auquel chaque élément prend une initiale capitalisée (par ex. `LightSlateBlue`). Il peut aussi s'exprimer sous la forme d'une suite de mots, auquel cas les initiales ne sont pas capitalisées (par ex. `light slate blue`).
- Le *gris* accepte aussi bien l'orthographe « grey » que « gray » (par ex. `DarkSlateGray`).
- Certains noms peuvent prendre un suffixe numérique, comme `LightSalmon4`.

Ils sont accessibles par la fonction Scheme `x11-color`.

Les listes suivantes présentent tous les noms de couleur disponibles, sans adjonction de suffixe numérique, puis celles acceptant un suffixe.

Noms de couleur sans suffixe numérique

Les couleurs suivantes peuvent s'utiliser sans suffixe numérique.

aliceblue
antiquewhite
aquamarine
azure
beige
bisque
black
blanchedalmond
blue
blueviolet
brown
burlywood
cadetblue
chartreuse
chocolate
coral
cornflowerblue
cornsilk
cyan
darkblue
darkcyan
darkgoldenrod
darkgray
darkgreen
darkgrey
darkkhaki
darkmagenta
darkolivegreen
darkorange
darkorchid
darkred
darksalmon
darkseagreen
darkslateblue
darkslategray
darkslategrey
darkturquoise
darkviolet
deeppink
deepskyblue
dimgray
dimgrey
dodgerblue
firebrick
floralwhite
forestgreen
gainsboro
ghostwhite
gold
goldenrod
gray
green
greenyellow
grey
honeydew
hotpink
indianred
ivory
khaki
lavender
lavenderblush
lawngreen
lemonchiffon
lightblue
lightcoral
lightcyan
lightgoldenrod
lightgoldenrodyellow
lightgray
lightgreen
lightgrey
lightpink
lightsalmon
lightseagreen
lightskyblue
lightslateblue
lightslategray
lightslategrey
lightsteelblue
lightyellow
limegreen
linen
magenta
maroon
mediumaquamarine
mediumblue
mediumorchid
mediumpurple
mediumseagreen
mediumslateblue
mediumspringgreen
mediumturquoise
mediumvioletred
midnightblue
mintcream
mistyrose
moccasin
navajowhite
navy
navyblue
oldlace
olivedrab
orange
orangered
orchid
palegoldenrod
palegreen
paleturquoise
palevioletred
papayawhip
peachpuff
peru
pink
plum
powderblue
purple
red
rosybrown
royalblue
saddlebrown
salmon
sandybrown
seagreen
seashell
sienna
skyblue
slateblue
slategray
slategrey
snow
springgreen
steelblue
tan
thistle
tomato
turquoise
violet
violetred
wheat
white
whitesmoke
yellow
yellowgreen

Noms de couleur avec suffixe numérique

Les couleurs suivantes acceptent un suffixe entier numérique N compris entre 1 et 4, allant du plus clair au plus foncé. Il est à noter que, dans le cas de ces couleurs, la teinte `toto` est identique à `toto1`.

antiquewhite1	antiquewhite2	antiquewhite3	antiquewhite4
aquamarine1	aquamarine2	aquamarine3	aquamarine4
azure1	azure2	azure3	azure4
bisque1	bisque2	bisque3	bisque4
blue1	blue2	blue3	blue4
brown1	brown2	brown3	brown4
burlywood1	burlywood2	burlywood3	burlywood4
cadetblue1	cadetblue2	cadetblue3	cadetblue4
chartreuse1	chartreuse2	chartreuse3	chartreuse4
chocolate1	chocolate2	chocolate3	chocolate4
coral1	coral2	coral3	coral4
cornsilk1	cornsilk2	cornsilk3	cornsilk4
cyan1	cyan2	cyan3	cyan4
darkgoldenrod1	darkgoldenrod2	darkgoldenrod3	darkgoldenrod4
darkolivegreen1	darkolivegreen2	darkolivegreen3	darkolivegreen4
darkorange1	darkorange2	darkorange3	darkorange4
darkorchid1	darkorchid2	darkorchid3	darkorchid4
darkseagreen1	darkseagreen2	darkseagreen3	darkseagreen4
darkslategray1	darkslategray2	darkslategray3	darkslategray4
deeppink1	deeppink2	deeppink3	deeppink4
deepskyblue1	deepskyblue2	deepskyblue3	deepskyblue4
dodgerblue1	dodgerblue2	dodgerblue3	dodgerblue4
firebrick1	firebrick2	firebrick3	firebrick4
gold1	gold2	gold3	gold4
goldenrod1	goldenrod2	goldenrod3	goldenrod4
green1	green2	green3	green4
honeydew1	honeydew2	honeydew3	honeydew4
hotpink1	hotpink2	hotpink3	hotpink4
indianred1	indianred2	indianred3	indianred4
ivory1	ivory2	ivory3	ivory4
khaki1	khaki2	khaki3	khaki4
lavenderblush1	lavenderblush2	lavenderblush3	lavenderblush4
lemonchiffon1	lemonchiffon2	lemonchiffon3	lemonchiffon4
lightblue1	lightblue2	lightblue3	lightblue4
lightcyan1	lightcyan2	lightcyan3	lightcyan4
lightgoldenrod1	lightgoldenrod2	lightgoldenrod3	lightgoldenrod4
lightpink1	lightpink2	lightpink3	lightpink4
lightsalmon1	lightsalmon2	lightsalmon3	lightsalmon4
lightskyblue1	lightskyblue2	lightskyblue3	lightskyblue4
lightsteelblue1	lightsteelblue2	lightsteelblue3	lightsteelblue4
lightyellow1	lightyellow2	lightyellow3	lightyellow4
magenta1	magenta2	magenta3	magenta4
maroon1	maroon2	maroon3	maroon4
mediumorchid1	mediumorchid2	mediumorchid3	mediumorchid4
mediumpurple1	mediumpurple2	mediumpurple3	mediumpurple4
mistyrose1	mistyrose2	mistyrose3	mistyrose4

navajowhite1	navajowhite2	navajowhite3	navajowhite4
olivedrab1	olivedrab2	olivedrab3	olivedrab4
orange1	orange2	orange3	orange4
orangered1	orangered2	orangered3	orangered4
orchid1	orchid2	orchid3	orchid4
palegreen1	palegreen2	palegreen3	palegreen4
paleturquoise1	paleturquoise2	paleturquoise3	paleturquoise4
palevioletred1	palevioletred2	palevioletred3	palevioletred4
peachpuff1	peachpuff2	peachpuff3	peachpuff4
pink1	pink2	pink3	pink4
plum1	plum2	plum3	plum4
purple1	purple2	purple3	purple4
red1	red2	red3	red4
rosybrown1	rosybrown2	rosybrown3	rosybrown4
royalblue1	royalblue2	royalblue3	royalblue4
salmon1	salmon2	salmon3	salmon4
seagreen1	seagreen2	seagreen3	seagreen4
seashell1	seashell2	seashell3	seashell4
sienna1	sienna2	sienna3	sienna4
skyblue1	skyblue2	skyblue3	skyblue4
slateblue1	slateblue2	slateblue3	slateblue4
slategray1	slategray2	slategray3	slategray4
snow1	snow2	snow3	snow4
springgreen1	springgreen2	springgreen3	springgreen4
steelblue1	steelblue2	steelblue3	steelblue4
tan1	tan2	tan3	tan4
thistle1	thistle2	thistle3	thistle4
tomato1	tomato2	tomato3	tomato4
turquoise1	turquoise2	turquoise3	turquoise4
violetred1	violetred2	violetred3	violetred4
wheat1	wheat2	wheat3	wheat4
yellow1	yellow2	yellow3	yellow4

Échelle de gris

Une échelle de gris, compris entre 0 et 100, est aussi disponible.

gray0	gray1	gray2	gray3	gray4
gray5	gray6	gray7	gray8	gray9
gray10	gray11	gray12	gray13	gray14
gray15	gray16	gray17	gray18	gray19
gray20	gray21	gray22	gray23	gray24
gray25	gray26	gray27	gray28	gray29
gray30	gray31	gray32	gray33	gray34
gray35	gray36	gray37	gray38	gray39
gray40	gray41	gray42	gray43	gray44
gray45	gray46	gray47	gray48	gray49
gray50	gray51	gray52	gray53	gray54
gray55	gray56	gray57	gray58	gray59
gray60	gray61	gray62	gray63	gray64
gray65	gray66	gray67	gray68	gray69
gray70	gray71	gray72	gray73	gray74

gray75 gray76 gray77 gray78 gray79
gray80 gray81 gray82 gray83 gray84
gray85 gray86 gray87 gray88 gray89
gray90 gray91 gray92 gray93 gray94
gray95 gray96 gray97 gray98 gray99
gray100

Couleurs in-ambiguës

La fonction Scheme `universal-color` procure un jeu de huit couleurs (`https://jfly.uni-koeln.de/color`) conçu pour lever toute ambiguïté aux personnes souffrant de daltonisme.

black orange skyblue bluegreen
yellow blue vermillion redpurple

B.8 La fonte Emmentaler

La fonte Emmentaler est constituée de deux *jeux* de glyphes : « Feta » est utilisé pour la notation classique, et « Parmesan » pour la notation ancienne.

Les différents symboles – ou glyphes – de la fonte Emmentaler peuvent s'inclure directement dans un objet *markup*. Pour ce faire, il suffit d'employer le nom du glyphe (voir les tables ci-après) comme ceci :

```
g^\markup { \musicglyph "scripts.segno" }
```

ou

```
\markup { \musicglyph "five" }
```

Pour de plus amples informations, reportez-vous au chapitre Section 8.2 [Mise en forme du texte], page 329.

B.8.1 Tables de glyphes modernes

Tous les glyphes du sous-ensemble « Feta ».

Glyphes de clef

`clefs.C`  `clefs.C_change` 

Glyphes de métrique

`timesig.C44`  `timesig.C44.text` 

Glyphes de chiffre

`plus` + `comma` ,

Glyphes de symbole pour la basse chiffrée

`figbass.twoplus`  `figbass.fourplus` 

`figbass.fiveplus`  `figbass.sixstroked` 

`figbass.sevenstroked` `figbass.ninestroked` 

Glyphes d'altération

accidentals.sharp 

accidentals.sharp.figbass 

Glyphes de tête de note par défaut

noteheads.uM2 

noteheads.dM2 

Glyphes de tête de note spéciale

noteheads.sM1double 

noteheads.s0diamond 

Glyphes de tête de note à forme variable

noteheads.s0do

noteheads.s1do

Glyphes de silence

rests.0

rests.1

Glyphes de crochet de croche

flags.u3 

flags.u4 

Glyphes de point

dots.dot

Glyphes de nuance

space

f 

Glyphes de script

scripts.ufermata 

scripts.dfermata 

Glyphes de flèche

arrowheads.open.01

arrowheads.open.0M1

Glyphes d'extrémité d'accolade

brackettips.up 

brackettips.down 

Glyphes de pédale

pedal.* 

pedal.M

Glyphes d'accordéon

accordion.discant 

accordion.dot

Glyphes de liaison

ties.lyric.short

ties.lyric.default

B.8.2 Tables des glyphes anciens

Tous les glyphes du sous-ensemble « Parmesan ».

Glyphes de style vaticana

clefs.vaticana.do

clefs.vaticana.do_change

Glyphes de style medicaea

clefs.medicaea.do 

clefs.medicaea.do_change 

Glyphes de style Hufnagel

clefs.hufnagel.do

clefs.hufnagel.do_change

Glyphes de style mensural

rests.M3mensural

rests.M2mensural

Glyphes de style néomensural

rests.M3neomensural 

rests.M2neomensural

Glyphes de style Petrucci

clefs.petrucci.c1 

clefs.petrucci.c1_change 

Glyphes de style Solesmes

noteheads.ssolesmes .incl.parvum

noteheads .ssolesmes.auct.asc
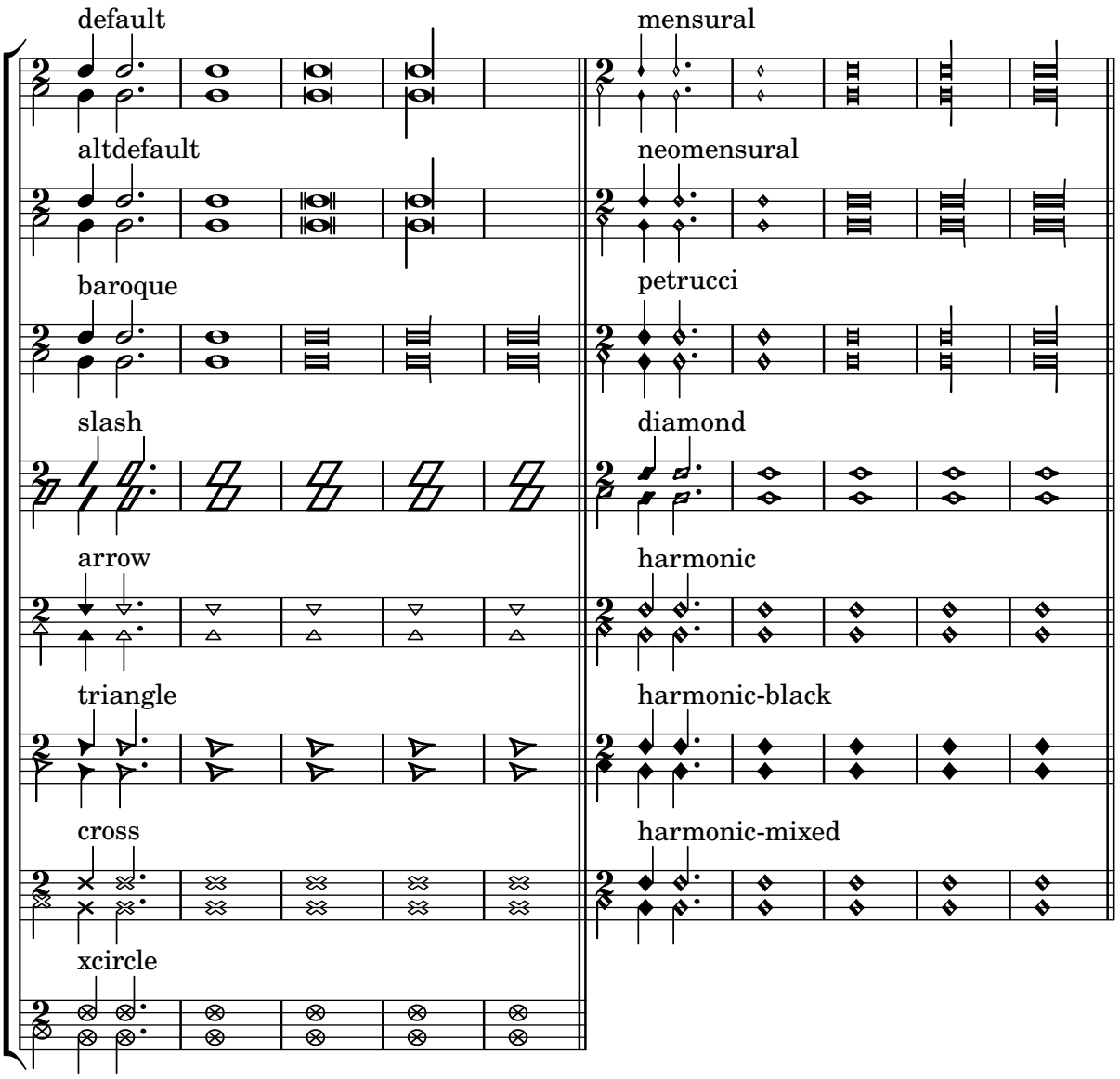
Glyphes de style kiévien

clefs.kievan.do

clefs.kievan.do_change

B.9 Styles de tête de note
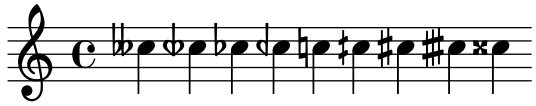
Voici les différents styles de tête de note disponibles.
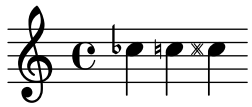


B.10 Jeux de glyphes d'altération

Voici les différents jeux de glyphes d'altération.
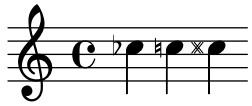
`standard-alteration-glyph-name-alist`
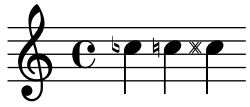


`alteration-hufnagel-glyph-name-alist`
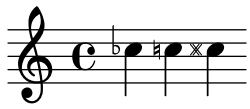


`alteration-medicaea-glyph-name-alist`
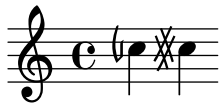


`alteration-vaticana-glyph-name-alist`



`alteration-mensural-glyph-name-alist`
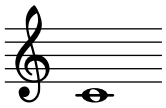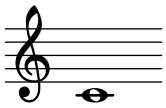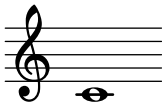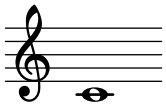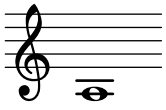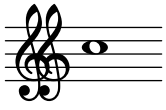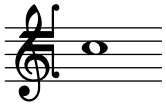


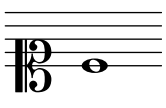
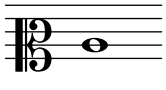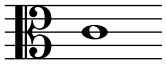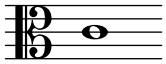
`alteration-kievan-glyph-name-alist`
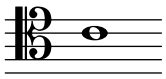


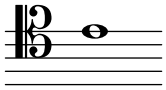
B.11 Styles de clef

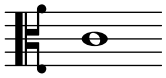
Le tableau suivant répertorie tous les styles de clef disponibles ainsi que la position du *do médium* par rapport à la clef. Certaines modifications, comme les indications d'octaviation, sont mentionnées dans Section 1.3.1 [Clefs], page 20.

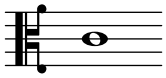
B.11.1 Clefs standards
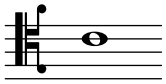
Exemple	**Résultat**	**Exemple**	**Résultat**
`\clef G`		`\clef "G2"`	
`\clef treble`		`\clef violin`	
`\clef french`		`\clef GG`	
`\clef tenorG`			
`\clef soprano`		`\clef mezzosoprano`	
`\clef C`		`\clef alto`	

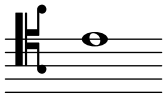
```
\clef tenor
```
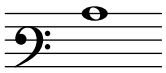

```
\clef baritone
```
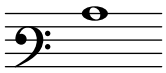

```
\clef varC
```
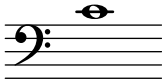

```
\clef altovarC
```
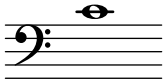

```
\clef tenorvarC
```

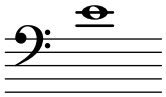

```
\clef baritonevarC
```


```
\clef varbaritone
```


```
\clef baritonevarF
```

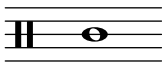

```
\clef F
```

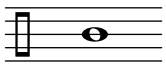

```
\clef bass
```


```
\clef subbass
```


B.11.2 Clefs pour portée de percussions
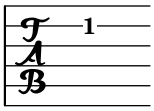
Exemple **Résultat** **Exemple** **Résultat**

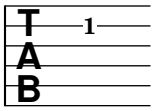
```
\clef percussion
```


```
\clef varpercussion
```


B.11.3 Clefs pour tablatures

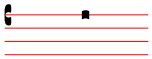
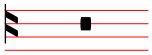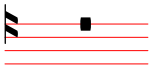
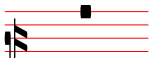
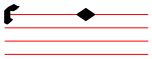
Exemple **Résultat** **Exemple** **Résultat**

```
\new TabStaff {
  \clef tab
}
```

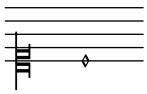
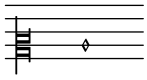
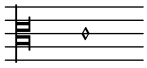
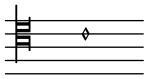
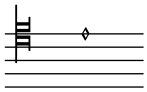
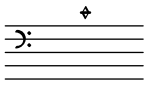

```
\new TabStaff {
  \clef moderntab
}
```


Clefs de musique ancienne

Grégorien

Exemple	Résultat	Exemple	Résultat
`\clef "vaticana-do1"`		`\clef "vaticana-do2"`	
`\clef "vaticana-do3"`		`\clef "vaticana-fa1"`	
`\clef "vaticana-fa2"`			
`\clef "medicaea-do1"`		`\clef "medicaea-do2"`	
`\clef "medicaea-do3"`		`\clef "medicaea-fa1"`	
`\clef "medicaea-fa2"`			
`\clef "hufnagel-do1"`		`\clef "hufnagel-do2"`	
`\clef "hufnagel-do3"`		`\clef "hufnagel-fa1"`	
`\clef "hufnagel-fa2"`		`\clef "hufnagel-do-fa"`	

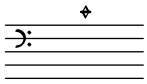
Mensural

Exemple	Résultat	Exemple	Résultat
`\clef "mensural-c1"`		`\clef "mensural-c2"`	
`\clef "mensural-c3"`		`\clef "mensural-c4"`	
`\clef "mensural-c5"`			
`\clef "mensural-f"`		`\clef "mensural-f2"`	

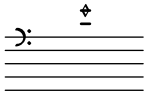
\clef "mensural-f3"



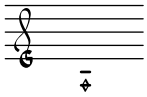
\clef "mensural-f4"



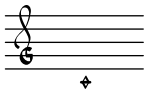
\clef "mensural-f5"



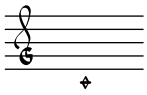
\clef "mensural-g1"



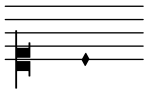
\clef "mensural-g2"



\clef "mensural-g"



\clef "blackmensural-c1"



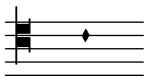
\clef "blackmensural-c2"



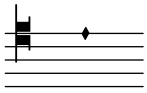
\clef "blackmensural-c3"



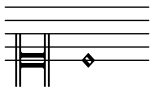
\clef "blackmensural-c4"
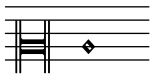


\clef "blackmensural-c5"



\clef "neomensural-c1"
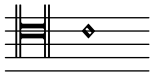


\clef "neomensural-c2"
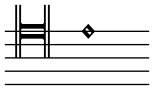


\clef "neomensural-c3"
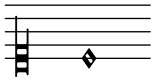


\clef "neomensural-c4"



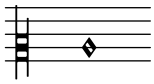
\clef "neomensural-c5"



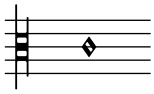
\clef "petrucci-c1"
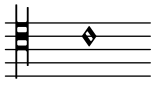


\clef "petrucci-c2"



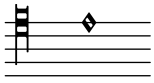
\clef "petrucci-c3"
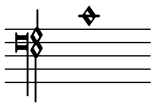


\clef "petrucci-c4"



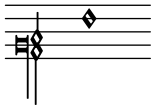
\clef "petrucci-c5"
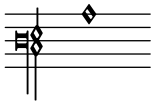


\clef "petrucci-f"



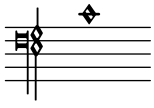
\clef "petrucci-f2"



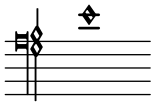
\clef "petrucci-f3"
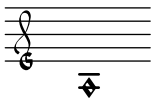


\clef "petrucci-f4"



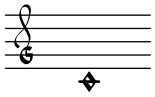
\clef "petrucci-f5"



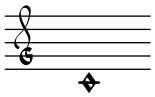
\clef "petrucci-g1"
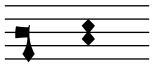


\clef "petrucci-g2"



\clef "petrucci-g"



Kiévien

Exemple	**Résultat**
\clef "kievan-do"	

B.12 Liste des caractères spéciaux

Voici une table des caractères spéciaux disponibles. Pour plus de précisions, voir Section 22.4.3 [Équivalents ASCII], page 665.

Cette liste utilise la syntaxe HTML, à l'instar de la plupart des caractères qui la composent ; les autres sont inspirés du langage LaTeX.

Les caractères sont ici inclus dans une boîte, de façon à mettre leur taille en évidence, et un léger décalage a été appliqué pour les décoller de l'encadrement.
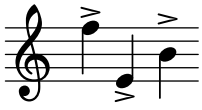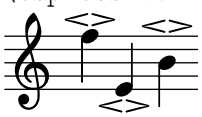
¡ ¡ **¿** ¿ **&solidus;** ⁄ **&flq;** ‹

B.13 Liste des signes d'articulation
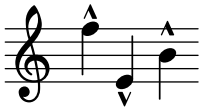
Dans la logique interne de LilyPond, une « articulation » est un objet, hormis les nuances, qui peut s'attacher directement après un événement rythmique, qu'il s'agisse d'une note ou d'un accord, y compris un silence, un saut ou bien une construction en accord vide <> (voir Section "Structuration de la saisie des notes" dans *Manuel d'initiation*). Même les liaisons, doigtés et scripts textuels sont techniquement des articulations, bien qu'ils ne soient pas indiqués ici.
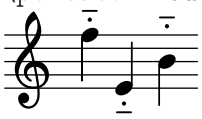
Les listes qui suivent recensent les articulations et autres symboles prédéfinis dans la fonte Emmentaler, que vous pouvez attacher à une note (par ex. `f\accent` ou `f->`). Chaque exemple illustre les trois positionnements : en surplomb (*up*), en dessous (*down*) et à l'appréciation de LilyPond (*neutral*). Consultez aussi [Script glyphs], page 936, pour un recensement plus exhaustif des glyphes accessibles par la commande de *markup* `\musicglyph` en suivant les préceptes énoncés dans Section 8.2.5 [Notation musicale dans du texte formaté], page 345.
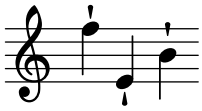
Scripts d'articulation
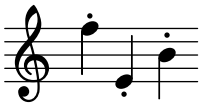
\accent ou -> 
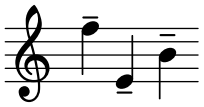
\espressivo 

\marcato ou -^ 
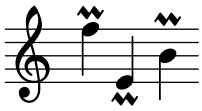
\portato ou -_ 
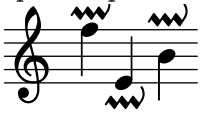
\staccatissimo ou -! 

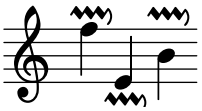
\staccato ou -. 
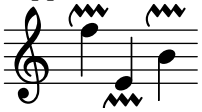
\tenuto ou -- 
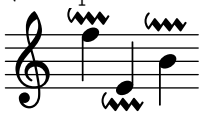
Scripts d'ornement

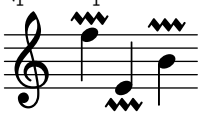
\prall 
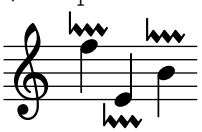
\prallup 
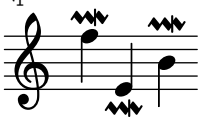
\pralldown 

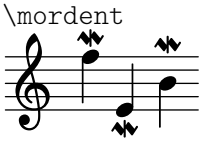
\upprall 

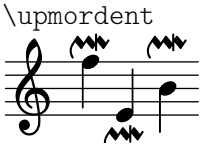
\downprall 
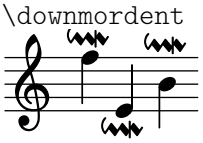
\prallprall 
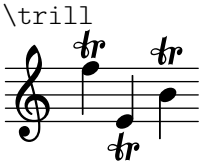
\lineprall 
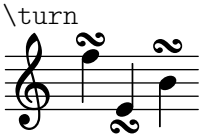
\prallmordent 

\mordent 
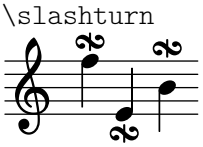
\upmordent 
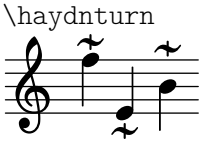
\downmordent 

\trill 
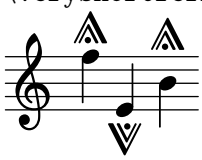
\turn 
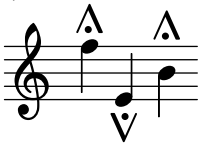
\reverseturn 
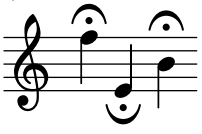
\slashturn 
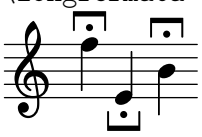
\haydnturn 
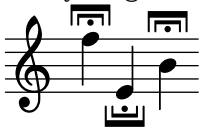
Scripts de point d'orgue et point d'arrêt
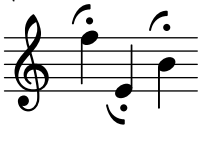
\veryshortfermata 
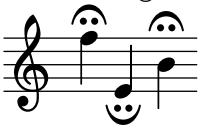
\shortfermata 

\fermata 

\longfermata 

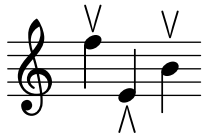
\verylongfermata 

\henzeshortfermata 

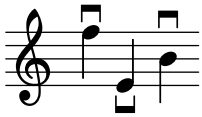
\henzelongfermata 

Scripts spécifiques à certains instruments

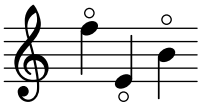
\upbow



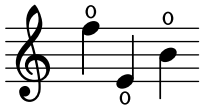
\downbow
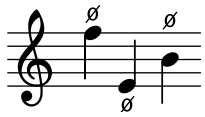


\flageolet



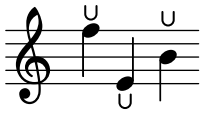
\open



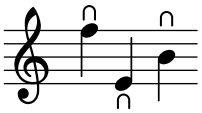
\halfopen



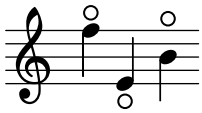
\heel
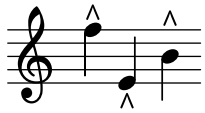


\varheel



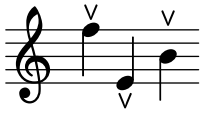
\heelcircle



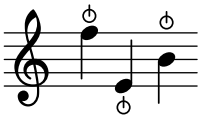
\toe



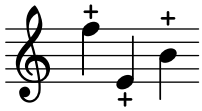
\vartoe



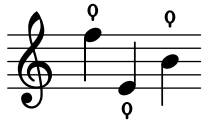
\snappizzicato



\stopped ou -+
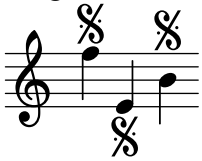


\thumb
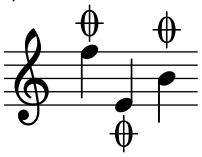


Scripts de reprise et de répétition

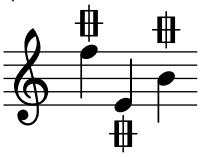
\segno



\coda

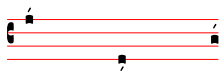


\varcoda



Scripts pour musique ancienne

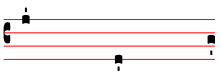
\accentus



\circulus



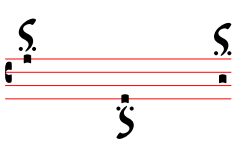
\ictus



\semicirculus
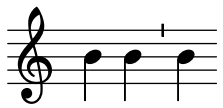


\signumcongruentiae
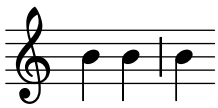


B.14 Liste des marques de respiration
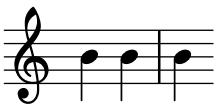
'chantquarterbar
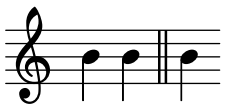


'chanthalfbar
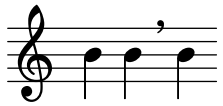


'chantfullbar
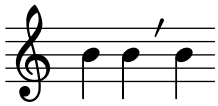


'chantdoublebar
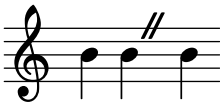


'comma
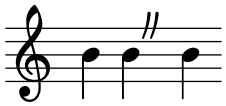


'varcomma
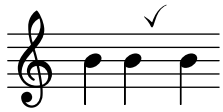


'caesura
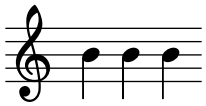


'curvedcaesura



'tickmark
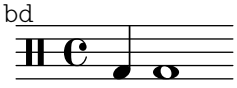


'spacer
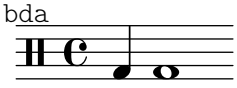


B.15 Notes utilisées en percussion
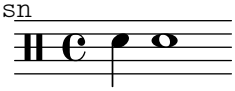
bassdrum
bd
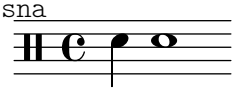


acousticbassdrum
bda
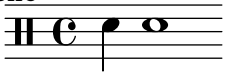


snare
sn
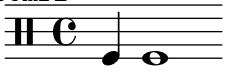


acousticsnare
sna
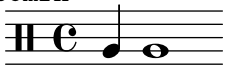


electricsnare
sne
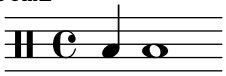


lowfloortom
tomfl
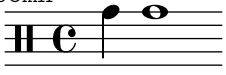


highfloortom
tomfh
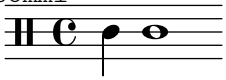


lowtom
toml
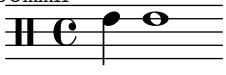


hightom
tomh



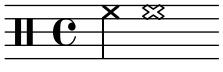
lowmidtom
tomml



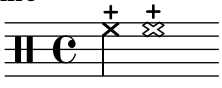
himidtom
tommh



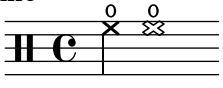
hihat
hh
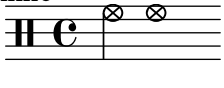


closedhihat
hhc



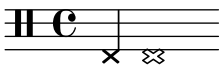
openhihat
hho
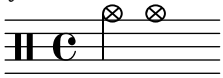


halfopenhihat
hhho
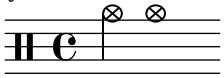


pedalhihat
hhp
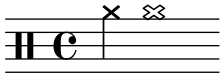


crashcymbal
cymc
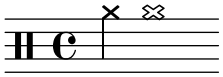


crashcymbala
cymca
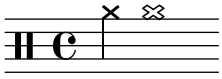


crashcymbalb
cymcb
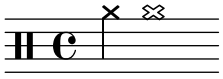


ridecymbal
cymr
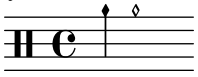


ridecymbala
cymra
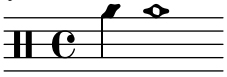


ridecymbalb
cymrb
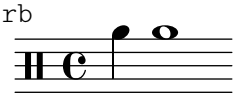


chinesecymbal
cymch
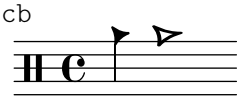


splashcymbal
cyms
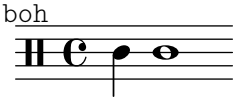


ridebell
rb
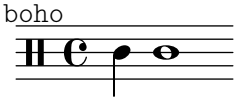


cowbell
cb
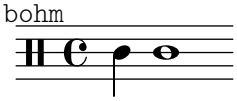


hibongo
boh
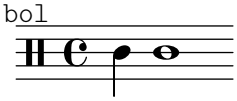


openhibongo
boho
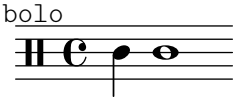


mutehibongo
bohm
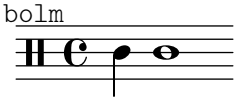


lobongo
bol
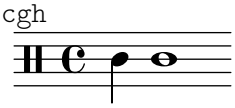


openlobongo
bolo
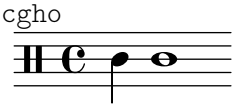


mutelobongo
bolm
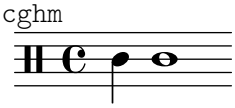


hiconga
cgh
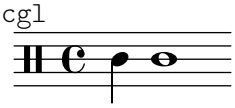


openhiconga
cgho
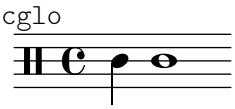


mutehiconga
cghm
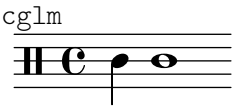


loconga
cgl
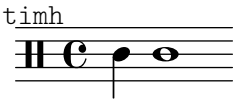


openloconga
cglo
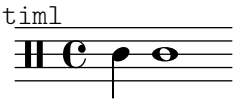


muteloconga
cglm
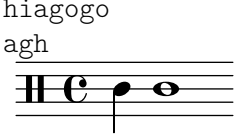


hitimbale
timh



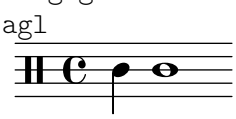
lotimbale
timl
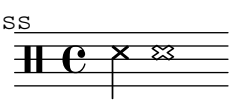


hiagogo
agh
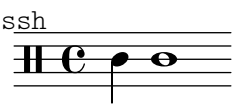


loagogo
agl
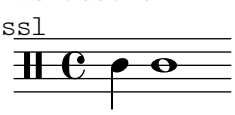


sidestick
ss
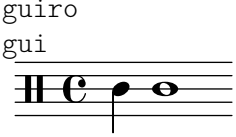


hisidestick
ssh
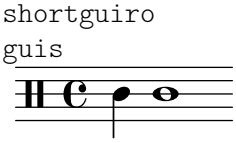


losidestick
ssl
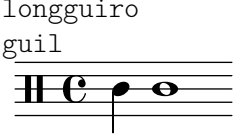


guiro
gui
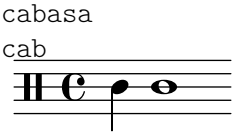


shortguiro
guis
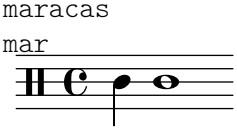


longguiro
guil
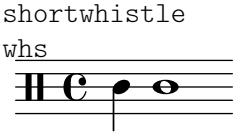


cabasa
cab
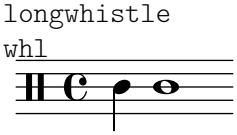


maracas
mar
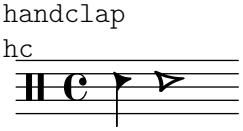


shortwhistle
whs



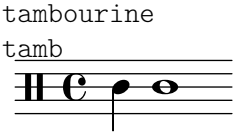
longwhistle
whl
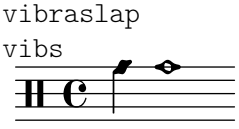


handclap
hc
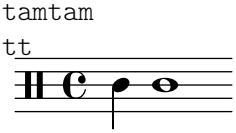


tambourine
tamb
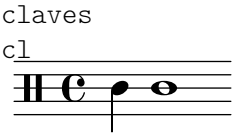


vibraslap
vibs



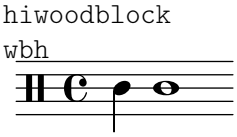
tamtam
tt



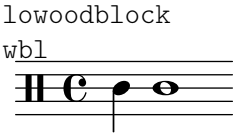
claves
cl
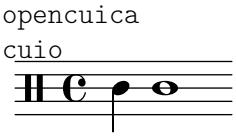


hiwoodblock
wbh
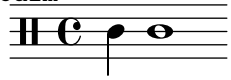


lowoodblock
wbl
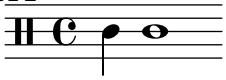


opencuica
cuio
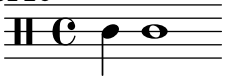


mutecuica
cuim
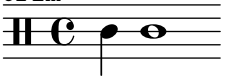


triangle
tri



opentriangle
trio



mutetriangle
trim



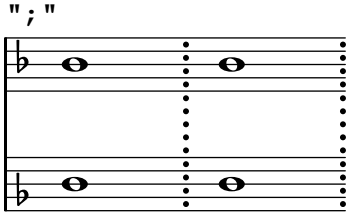
B.16 Liste des types de barre de mesure

Dans chacune des systèmes suivants est indiquée l'apparence d'un type particulier de barre selon qu'elle intervient au début, en cours et en fin de ligne.

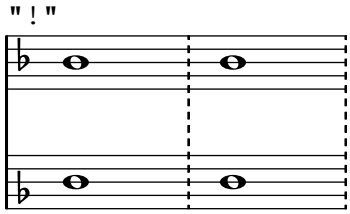
Pour de plus amples informations, voir Section 2.5.1 [Barres de mesure], page 123, and Section 2.5.2 [Barres de mesure automatiques], page 133.

Simple bar lines

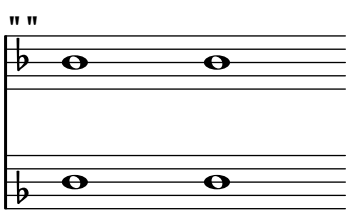
";"



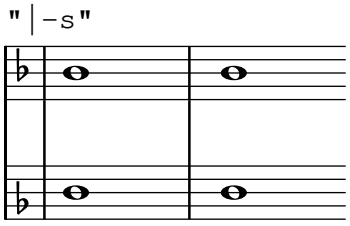
"!"



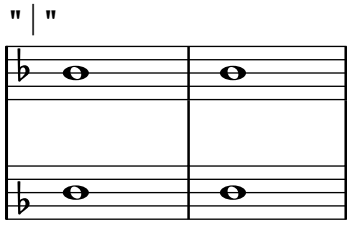
""



"|-s"

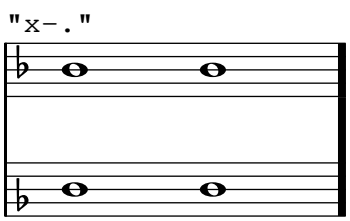


"|"

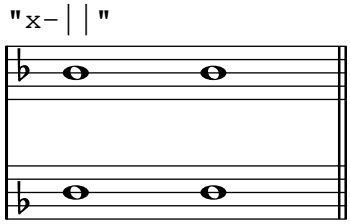


End-of-line bar lines

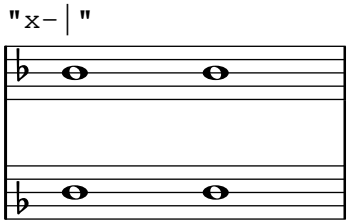
"x-."



"x-||"

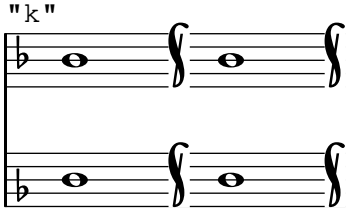


"x-|"

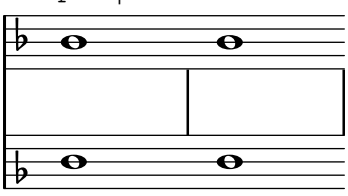


Chant bar lines

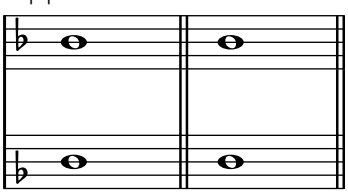
"k"



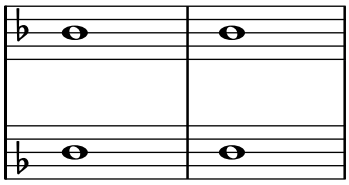
"-span|"



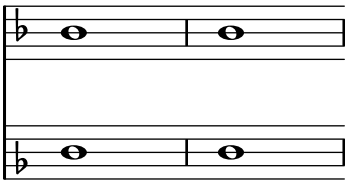
"||"



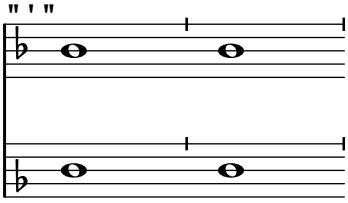
"|"



","

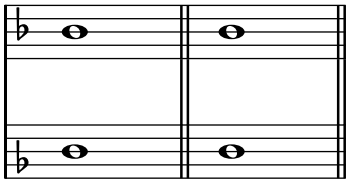


"'"

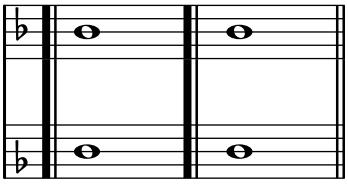


Section bar lines

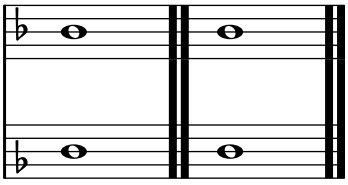
"||"



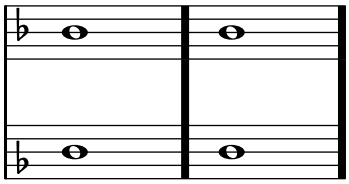
".|-||"



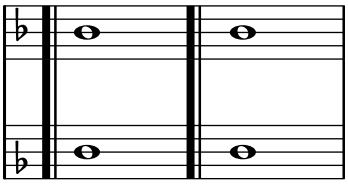
".."



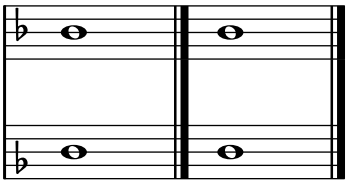
"."



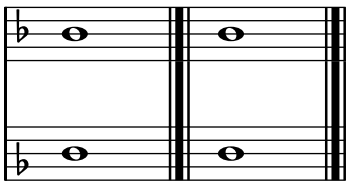
".|-|"



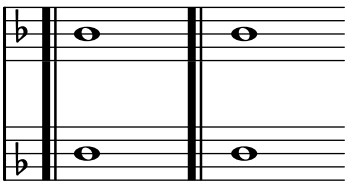
"|."



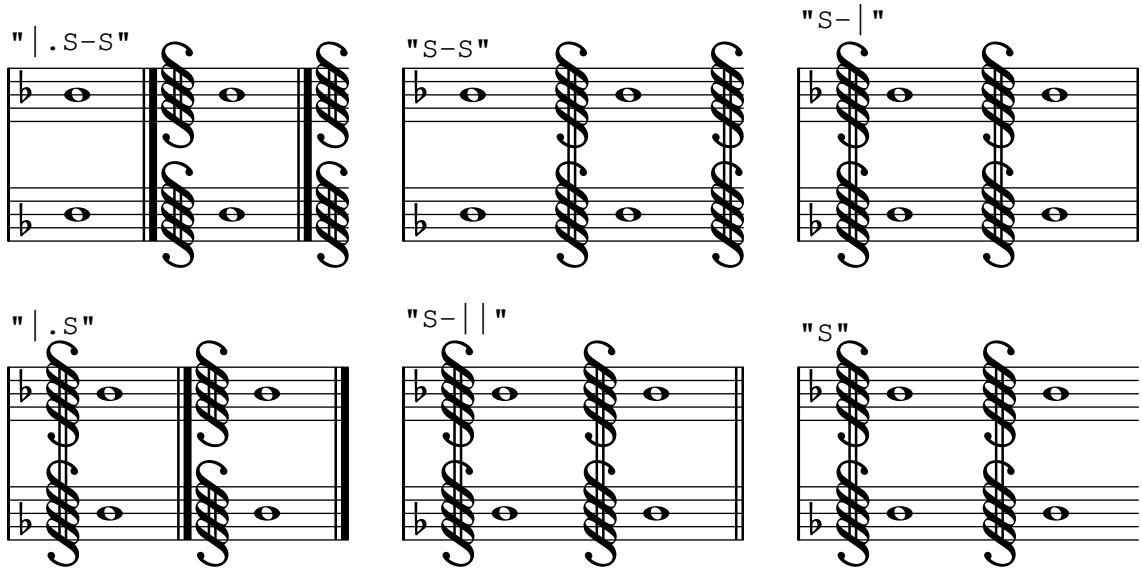
"|.|"



".|"

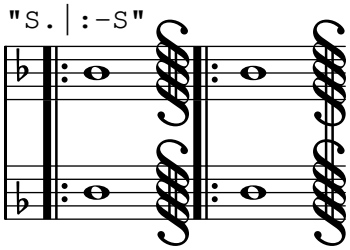


Segno bar lines

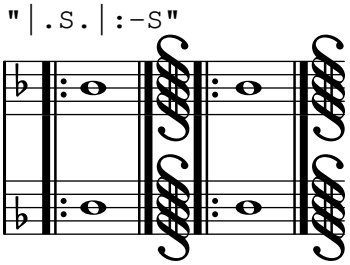


Start-repeat bar lines

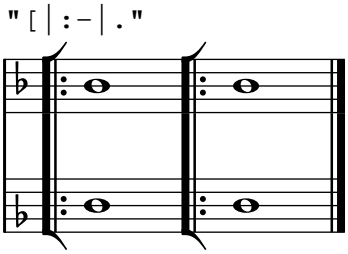
"s.|:-s"



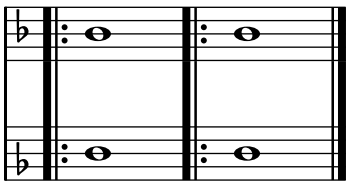
"|.s.|:-s"



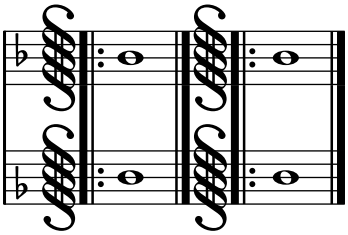
"[|:-|."



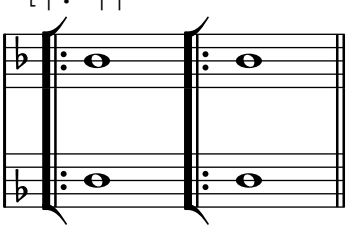
".|:-|."



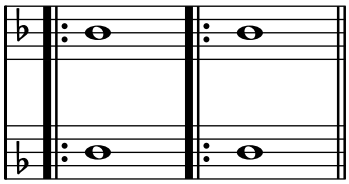
"|.s.|:"



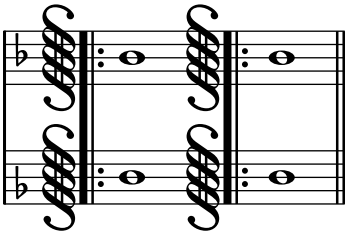
"[|:-||"



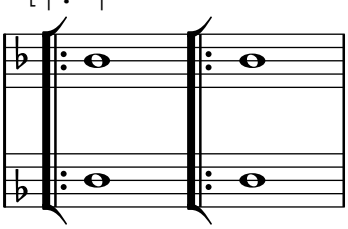
".|:-||"



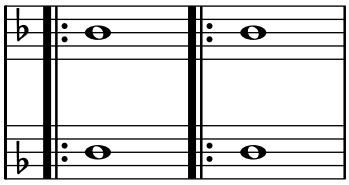
"s.|:-||"



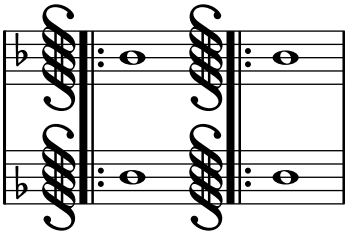
"[|:-|"



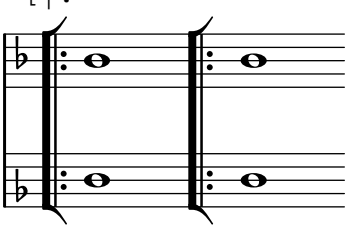
".|:-|"



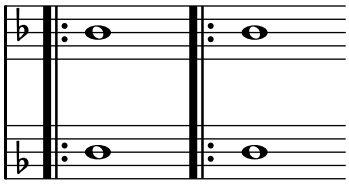
"s.|:-|"



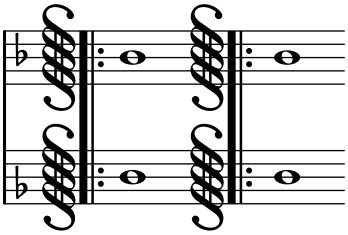
"[|:"



".|:"

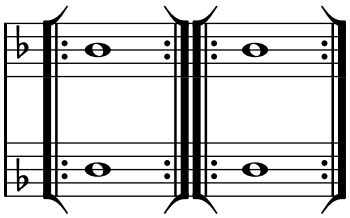


"s.|:"

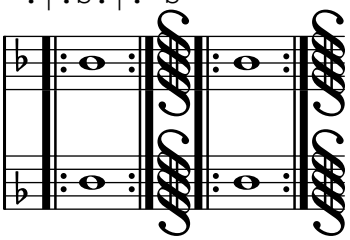


Double-repeat bar lines

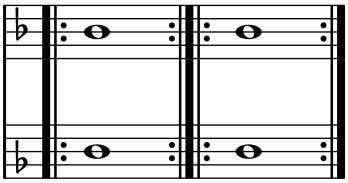
":|][|:"



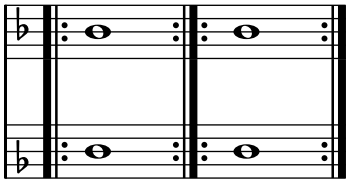
":|.S.|:-S"



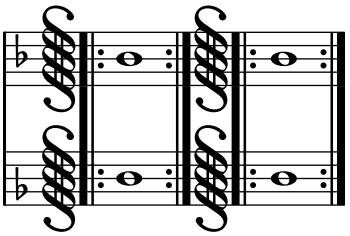
":|.|:"



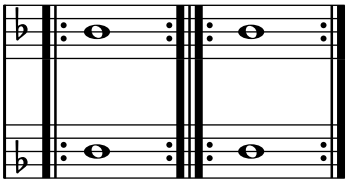
":|.:"



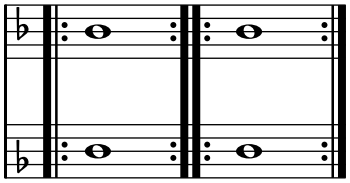
":|.S.|:"



":.|.:"

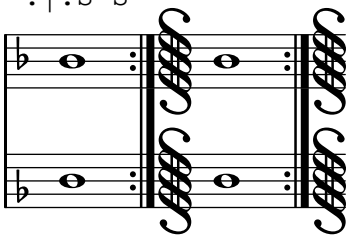


":..:"

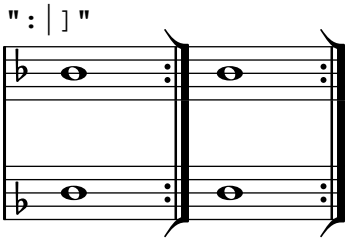


End-repeat bar lines

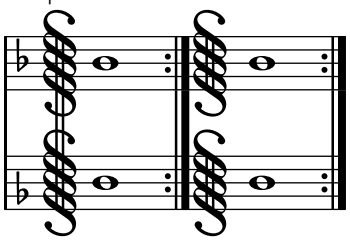
":|.S-S"



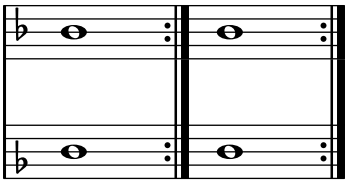
":|]"



":|.S"



":|."



B.17 Valeurs par défaut de `outside-staff-priority`

Le tableau suivant indique la valeur par défaut de la propriété `outside-staff-priority` de tous les objets graphiques extérieurs à la portée. Au plus la valeur est faible, au plus l'objet est proche de la portée.

Grob	Priority
AccidentalSuggestion	0
BassFigureAlignmentPositioning	25
MultiMeasureRestScript	40
TrillSpanner	50
BarNumber	100
LigatureBracket	200
DynamicLineSpanner	250
TextSpanner	350
OttavaBracket	400
MultiMeasureRestText	450
TextScript	450
CombineTextScript	475
InstrumentSwitch	500
VoltaBracketSpanner	600
MeasureCounter	750
MeasureSpanner	750
HorizontalBracket	800
SostenutoPedalLineSpanner	1000
SustainPedalLineSpanner	1000
UnaCordaPedalLineSpanner	1000
CenteredBarNumberLineSpanner	1200
TextMark	1250
MetronomeMark	1300
JumpScript	1350
CodaMark	1400
SegnoMark	1400
SectionLabel	1450
RehearsalMark	1500

B.18 Valeurs par défaut de `script-priority`

Le tableau suivant indique la valeur par défaut de de la propriété `script-priority` de tous les objets (nom avec capitale initiale) liés aux scripts (nom en minuscules). Le objets aux plus petites valeurs sont plus proches de la portée.

Les objets `Fingering`, `StringNumber` et `StrokeFinger`, lorsqu'ils font partie d'un accord, voient leur priorité finale accrue de la position verticale de la tête de note d'attachement. La valeur de `script-priority` des autres scripts de l'accord ne sont quant à elles pas modifiées.

En ce qui concerne les scripts non attachés à un accord, c'est la position de l'événement considéré qui est ajoutée. Par conséquent, lorsque plusieurs scripts ont une même valeur de `script-priority`, le premier aura préséance.

Les objets qui ne sont pas ici mentionnés bien qu'ils soient relatifs à des scripts voient leur `script-priority` fixée à 200 par défaut.

Grob or Script	Priority
AccidentalPlacement	-100
accentus	-100
circulus	-100
ictus	-100
semicirculus	-100
staccato	-100
tenuto	-50

AccidentalSuggestion	0
Arpeggio	0
ChordBracket	0
ChordSlur	0
flageolet	50
Fingering	100
StrokeFinger	125
StringNumber	150
trill	150
fermata	175
henzelongfermata	175
henzeshortfermata	175
longfermata	175
shortfermata	175
verylongfermata	175
veryshortfermata	175
downbow	180
upbow	180
CombineTextScript	200
TextScript	200

B.19 Objets graphiques et leurs symboles d'alignement après rupture

L'ordre et l'espacement des éléments de notation tels que clefs, métriques, barres de mesure, etc. sont déterminés à l'aide de certains symboles Scheme, appelés en anglais des *break-align symbols* et qu'on pourrait traduire par symboles d'alignement après rupture. Plus précisément, leur ordre est déterminé par des listes de symboles *break-align* dans le vecteur `BreakAlignment.break-align-orders` (voir Section "break-alignment-interface" dans *Référence des propriétés internes*) et leur espacement est contrôlé par les propriétés `space-alist` des objets respectifs, dont les éléments sont à leur tour indexés selon les symboles *break-align* Par exemple, l'espacement entre un objet `Clef` et la barre de mesure qui le suit est déterminé par la valeur de `Clef.space-alist .staff-bar`.

Ce tableau recense la correspondance par défaut entre les symboles *break-align* et les objets graphiques.

break-align symbol	**used by grob(s)**
ambitus	Ambitus
breathing-sign	BreathingSign
clef	Clef
cue-clef	CueClef
cue-end-clef	CueEndClef
custos	Custos
key-cancellation	KeyCancellation
key-signature	KeySignature
left-edge	LeftEdge
optional-material-end-bracket	OptionalMaterialBracket
optional-material-start-bracket	OptionalMaterialBracket
signum-repetitionis	SignumRepetitionis
staff-bar	SpanBar, DoublePercentRepeat, Divisio, and BarLine
staff-ellipsis	StaffEllipsis

time-signature	TimeSignature

Pour aller plus loin, il existe trois symboles *break-align* abstraits susceptible de s'utiliser dans des définitions de space-alist, à savoir first-note, next-note et right-edge. De plus amples détails sur la propriété space-alist des objets graphiques sont disponibles en consultant Section "break-aligned-interface" dans *Référence des propriétés internes*.

Le tableau suivant liste les symboles *break-align* utilisés par chacun des objets graphiques.

Grob	**break-align symbol(s)**
Ambitus	ambitus
BarLine	staff-bar
BreathingSign	breathing-sign
Clef	clef
CueClef	cue-clef
CueEndClef	cue-end-clef
Custos	custos
Divisio	staff-bar
DoublePercentRepeat	staff-bar
KeyCancellation	key-cancellation
KeySignature	key-signature
LeftEdge	left-edge
OptionalMaterialBracket	optional-material-start-bracket and optional-material-end-bracket
SignumRepetitionis	signum-repetitionis
SpanBar	staff-bar
StaffEllipsis	staff-ellipsis
TimeSignature	time-signature

B.20 Glossaire technique

Ce glossaire regroupe les termes techniques et concepts utilisés en interne par LilyPond. Ils apparaissent aussi bien dans les manuels, que sur les listes de diffusion et dans le code source.

alist (liste associative)
: Une liste associative – *alist* pour *association list* – est une paire Scheme qui associe une valeur à une clé : (clé . valeur). Le fichier scm/lily.scm contient par exemple une *alist* type-p-name-alist qui associe certains types de prédicat (par ex. ly:music?) à des noms (par ex. « music ») de telle sorte qu'une erreur lors d'un contrôle de typage puisse être rapportée en console avec mention du type de prédicat attendu.

callback (rappel)
: Un **rappel**, *callback* en anglais, est une routine, fonction ou méthode qui est passée en argument à une autre fonction. Cette dernière peut alors faire usage de cette fonction de rappel comme de n'importe quelle autre fonction, alors qu'elle ne la connaît pas par avance. Cette façon de procéder permet à des couches logicielles de bas niveau d'appeler des fonctions définies à des niveaux plus élevés. LilyPond utilise abondamment les *callbacks* afin que le code Scheme saisi par l'utilisateur puisse contrôler les actions de bas niveau.

closure (clôture)
: En Scheme, une **clôture** (en anglais *closure*) est créée lorsqu'une fonction, généralement une expression lambda, est passée en tant que variable. La clôture comporte, en plus du code de la fonction, des références à des variables libres dans l'environnement lexical – variables utilisées dans l'expression mais définies

ailleurs. Lorsque la fonction est par la suite appliquée aux différents arguments, les références aux variables libres, capturées par la clôture, permettent d'obtenir la valeur de ces variables libres qui sera utilisée lors des calculs. L'une des propriétés intéressantes des clôtures est la rétention de la valeur de variables internes tout au long des différentes invocations, leur état étant alors persistant.

glyphe

Un *glyphe* est une représentation graphique particulière d'un caractère typographique ou d'une combinaison de deux caractères formant une ligature. Un jeu de glyphes aux mêmes style et allure constitue une fonte ; un jeu de fontes comprenant plusieurs styles et tailles constitue un type de caractère.

Voir aussi : Section 8.3 [Fontes], page 347, Section 22.4 [Caractères spéciaux], page 664.

grob (objet graphique)

Dans LilyPond, les objets représentants les différents éléments de notation, comme les têtes de note, hampes, liaisons, doigtés, clefs etc. sont appelés « objets de rendu » ou objet graphique – en anglais *GRaphical OBjects* couramment abrégé en *grob*. Ils sont représentés par des instances de la classe `grob`.

Voir aussi : Section "Objets et interfaces" dans *Manuel d'initiation*, Section "Propriétés des objets de rendu" dans *Manuel d'initiation*.
Section B.24 [Conventions de nommage], page 989.
Section "grob-interface" dans *Référence des propriétés internes*, Section "All layout objects" dans *Référence des propriétés internes*.

inaltérable

Un objet est dit *inaltérable* – *immutable* en anglais – dès lors que son état ne peut être modifié après sa création ; les objets altérables, à l'inverse, peuvent être modifiés après création.

Pour LilyPond, une propriété est inaltérable ou commune lorsqu'elle définit les style ou le comportement par défaut d'objets graphiques ; une telle propriété est partagée par un certain nombre de *grobs*. En apparente contradiction avec ce que leur nom laisse accroire, de telles propriétés peuvent être adaptées par `\override` et `\revert`.

Voir plus avant pour les objets altérables.

interface

Les actions et propriétés communes à plusieurs objets graphiques sont regroupées dans un objet appelé `grob-interface`, ou *interface* pour faire court.

Voir aussi : Section 34.2 [Interfaces de rendu], page 783.
Section "Objets et interfaces" dans *Manuel d'initiation*, Section "Propriétés listées par interface" dans *Manuel d'initiation*.
Section "Graphical Object Interfaces" dans *Référence des propriétés internes*.

lexer (analyseur lexical)

Un *lexer* est un programme chargé de convertir une séquence de caractères en une séquence de jetons. Cette opération s'appelle *analyse lexicale*. L'analyseur lexical de LilyPond convertit le flot d'information contenu dans un fichier `.ly` en flot de jetons qui pourront être traités lors de l'étape suivant, l'analyse grammaticale abordée plus avant. L'analyseur lexical de LilyPond repose sur le programme `flex` ; les règles lexicales sont regroupées dans le fichier `lily/lexer.ll`. Ce fichier, partie intégrante des sources, n'est pas distribué avec les programmes binaires de LilyPond.

altérable

Un objet est dit *altérable* – *mutable* en anglais – lorsque son état est sujet à modification après sa création, à l'inverse des objets inaltérables dont l'état est figé dès leur création.

Les propriétés altérables contiennent, pour LilyPond, des valeurs spécifiques à un objet graphique. En particulier, les listes d'autres objets ou résultats de calculs sont enregistrés sous dorme de propriétés altérables.

Voir ci-dessus pour les objets inaltérables.

output-def (définition de sortie)
: Une instance de la classe `Output-def` contient les méthodes et structures des données associées à un bloc de sortie. Ces instances sont créées par les blocs `\midi`, `\layout` et `\paper`.

parser (analyseur syntaxique)
: Un analyseur syntaxique – *parser* en anglais – est un programme qui analyse la séquence de jetons produite par l'analyseur lexical pour en déterminer la structure grammaticale. Les jetons sont, pour ce faire, regroupés progressivement en tronçons plus importants, selon des règles grammaticales. Lorsque la séquence de jetons est valide, le produit final est une arborescence de jetons ayant à sa base le symbole grammatical de début. Dès lors que cette étape n'est pas concluante, le fichier est déclaré invalide ; un message approprié est alors émis. Les différents regroupements syntaxiques ainsi que les règles de construction des regroupements relatifs à la grammaire de LilyPond sont définis dans le fichier `lily/parser.yy`. Ce fichier est utilisé par le générateur de *parser* `bison` lors de la construction du programme. Partie intégrante des sources, il n'est pas distribué avec les programmes binaires de LilyPond.

variable de l'analyseur grammatical
: Il s'agit de variables définies directement en Scheme. Dans la mesure où leur champ sémantique peut porter à confusion, il est fortement déconseillé de les utiliser tels quels.

  La modification des valeurs de l'une de ces variables dans un fichier `.ly` est effective de manière globale. Les valeurs modifiées, sauf à être explicitement remises à leur état d'origine, affectent tous les blocs `\score` rencontrés, y compris s'ils proviennent d'autres fichiers ajoutés par une commande `\include`. Ceci peut avoir des conséquences inattendues et les erreurs qui pourraient en découler difficiles à localiser dans le cadre d'un projet d'envergure.

  LilyPond utilise, entre autres, les variables suivantes :

  `afterGraceFraction`

  `input-language`

  `musicQuotes`

  `output-count`

  `output-suffix`

  `partCombineListener`

  `pitchnames`

  `toplevel-bookparts`

  `toplevel-scores`

  `showLastLength`

  `showFirstLength`

prob (objet de propriété)
: Les objets de propriété – *probs* pour *PRoperty OBjects* – sont des instances de la classe `Prob`, une classe de base simple pour les objets qui disposent de de listes associatives de propriétés altérables et inaltérables ainsi que les méthodes pour les manipuler. Les classes `Music` et `Stream_event` dérivent d'un `prob`. Les instances de la classe `prob` se créent aussi pour garder trace du contenu des systèmes une fois formatés et des blocs de titrage lors de la phase de mise en forme des pages.

smob (objet Scheme)
: Les objets Scheme – *Smobs* pour *ScheMe OBjects* – font partie du mécanisme utilisé par l'interpréteur Guile pour exporter en code Scheme les objets C ou C++. Dans LilyPond, les *smobs* sont créés, grâce à des macros, à partir d'objets C++. On peut distinguer deux types d'objets *smob* : des *smobs* simples destinés aux objets inaltérables comme les nombres par exemples, et des *smobs* complexes utilisés pour des objets possédant une identité. De plus amples informations sont disponibles dans les sources de LilyPond, au sein du fichier `lily/includes/smob.hh`.

spanner (bandeau)
: Les *bandeaux (spanners)* constituent une classe d'objets graphiques qui ne sont pas fixés horizontalement, mais s'étendent entre deux points. On peut citer à titre d'exemple les ligatures, les liaisons quel que soit leur type, les soufflets, et même les lignes de la portée. Contrairement aux objets ponctuels (contraire des bandeaux), qui peuvent apparaître au plus deux fois à l'occasion d'un saut, comme la duplication d'une clef en fin de ligne et au début de la suivante, les bandeaux sont sectionnés en autant de tronçons que de besoin selon leur point de départ et leur terminaison, comme un crescendo qui s'étend sur trois systèmes ou les lignes de portée qui s'étendent tout au long de la partition.

  Techniquement, les bandeaux se définissent comme des objets graphiques disposant de la `spanner-interface` ; du côté C++ de LilyPond, ils constituent des instances de la sous-classe `Spanner` de `grob`. Leurs extrémités gauche et droite peuvent se récupérer ou définir respectivement à l'aide de `ly:spanner-bound` et `ly:spanner-set-bound!`. Leurs extrémités sont toujours des objets ponctuels. Le parent X d'un bandeau n'a que peu de sens en termes de musique, mais il se détermine habituellement par son extrémité gauche.

  Voir aussi : Section 36.4 [Extenseurs et prolongateurs], page 804.
  Section “All layout objects” dans *Référence des propriétés internes*, Section “spanner-interface” dans *Référence des propriétés internes*.

stencil
: Une instance de la classe `Stencil` comporte l'information nécessaire à l'impression d'un objet typographique. Il s'agit d'un *smob* simple qui contient un espace de confinement qui définit l'envergure verticale et horizontale de l'objet ainsi qu'une expression Scheme qui imprimera l'objet après évaluation. Les stencils peuvent se combiner et adopter une forme plus complexe définie par une arborescence d'expressions Scheme des stencils qui la composent.

  La propriété `stencil`, qui permet de connecter un *grob* à son stencil, est définie par l'interface `grob-interface`.

  Voir aussi Section “grob-interface” dans *Référence des propriétés internes*.

B.21 Fonctions musicales prédéfinies

Dans le tableau suivant, le premier argument d'une fonction est présenté en italique avec son type entre parenthèses. Vient ensuite le deuxième argument avec, lui aussi, son type entre parenthèses, et ainsi de suite. Est indiqué, après la flèche, le type de valeur de retour.

Lorsqu'un argument et son type sont placés entre crochets, cet argument est optionnel et peut être omis, indiquant à LilyPond de prendre une valeur par défaut. Toutefois, si le dernier argument d'une fonction est optionnel, il *ne saurait* être omis ; il faudra le remplacer par `\default` en l'absence de valeur explicite. Voir Section "Utilisation de fonctions Scheme" dans *Extension de LilyPond* pour de plus amples détails sur le traitement des arguments optionnels.

`\absolute` *music* (music) ⇒ music
: Make *music* absolute.

  This does not actually change the music itself but rather hides it from surrounding `\relative` (page 972) and `\fixed` (page 964) commands.

`\acciaccatura` *music* (music) ⇒ music
: Create an acciaccatura from *music*.

`\accidentalStyle` *style* (symbol list) ⇒ music
: Set accidental style to *style*.

  style is a (predefined) symbol list like `piano-cautionary`; voir Section 1.3.5 [Altérations accidentelles automatiques], page 32, for the available styles. If it is preceded by a context name, the settings are applied to that context (example: `Staff.piano-cautionary`). Otherwise, the context defaults to `Staff`, except for piano styles, which use `GrandStaff` as a context.

`\addChordShape` *key-symbol* (symbol) *tuning* (pair) *shape-definition* (string or pair) ⇒ void
: Add *shape-definition* as a chord shape.

  It gets added to the `chord-shape-table` hash with the key `(cons` *`key-symbol tuning`*`)`.

`\addInstrumentDefinition` *name* (string) *lst* (list) ⇒ void
: Create instrument *name* with properties *lst*.

  This function is deprecated.

`\addQuote` *name* (string) *music* (music) ⇒ void
: Define *music* as a quotable music expression named *name*.

`\addToTagGroup` *tag-group* (symbol list) *more-tags* (symbol list) ⇒ void
: Add *more-tags* to the tag group identified by *tag-group*.

`\after` *delta* (duration-or-music) *ev* (music) *mus* (music) ⇒ music
: Add music *ev* with a delay of *delta* after the onset of *mus*.

  ev is usually a post-event.

`\afterGrace` [*fraction* (non-negative rational, fraction, or moment)] *main* (music) *grace* (music) ⇒ music
: Create *grace* as grace notes after a *main* music expression.

  The musical position of the grace expression is after a given fraction of the main note's duration has passed. If optional argument *fraction* is omitted, the fraction value is taken from `afterGraceFraction`, defaulting to 3/4.

`\allowPageTurn` ⇒ music
: Allow a page turn.

  May be used at top level (i.e., between scores or markups), or inside a score.

\allowVoltaHook *bar* (string) ⇒ void
: Allow the volta bracket hook being drawn over bar line *bar*.

\alterBroken *property* (key list or symbol) *arg* (list) *target* (key list or music) ⇒ music
: Override *property* for pieces of broken spanner *target* with *arg*.

  target may either be music in the form of a starting spanner event, or a symbol list of the form `Context.Grob` or just `Grob`. If *target* is in the form of a spanner event, *property* may also have the form `Grob.property` for specifying a directed tweak.

  arg is a list of values, one for each broken piece.

\ambitusAfter *target* (symbol) ⇒ music
: Move the ambitus after the break-align symbol *target*.

\appendToTag *tag* (symbol) *more* (music) *music* (music) ⇒ music
: Append *more* to *music* tagged with *tag*.

  A post-event can be added to the articulations of rhythmic events or chords; other expressions may be added to chords, sequential or simultaneous music.

\appendToTagMarkup *tag* (symbol) *more* (markup) *music* (music) ⇒ music
: Append *more* to every markup in *music* tagged with *tag*.

\applyContext *proc* (procedure) ⇒ music
: Modify context properties with Scheme procedure *proc*.

\applyMusic *func* (procedure) *music* (music) ⇒ music
: Apply procedure *func* to *music*.

\applyOutput *target* (symbol list or symbol) *proc* (procedure) ⇒ music
: Apply function `proc` to every layout object matched by *target*.

  target takes the form `Context` or `Context.Grob`.

\appoggiatura *music* (music) ⇒ music
: Create an appoggiatura from *music*.

\approximatePitch *note* (music) ⇒ music
: Indicate that the pitch of *note* approximates a pitch that cannot be known exactly, such as the highest note a singer can sing.

\arpeggioArrowDown ⇒ music
: Let \arpeggio produce a wavy line with a down arrow.

\arpeggioArrowUp ⇒ music
: Let \arpeggio produce a wavy line with an up arrow.

\arpeggioBracket ⇒ music
: Let \arpeggio produce a square bracket.

  Note: For a bracket designating a non-arpeggiated chord, it is better to use \nonArpeggiato than to use \arpeggio and alter the appearance.

\arpeggioNormal ⇒ music
: Let \arpeggio produce a wavy line without an arrow.

\arpeggioParenthesis ⇒ music
: Let \arpeggio produce a vertical slur.

  Note: For a vertical slur designating a quasi-non-arpeggiated chord, it is better to use \chordSlur than to use \arpeggio and alter the appearance.

\arpeggioParenthesisDashed ⇒ music
: Let \arpeggio produce a dashed vertical slur.

Note: For a vertical slur designating a quasi-non-arpeggiated chord, it is better to use `\chordSlur` than to use `\arpeggio` and alter the appearance.

`\assertBeamQuant` *l* (pair) *r* (pair) ⇒ music
Testing function: check whether the beam quants *l* and *r* are correct.

`\assertBeamSlope` *comp* (procedure) ⇒ music
Testing function: check whether the slope of the beam is the same as *comp*.

`\atLeft` *m* (music) ⇒ post-event
Set `side-axis` to `X` and `direction` to `LEFT` for *mus*.

`\atRight` *m* (music) ⇒ post-event
Set `side-axis` to `X` and `direction` to `RIGHT` for *mus*.

`\augmentum` *expr* (music) ⇒ music
Add augmentum dots (*morae*) to Gregorian chant *expr*.

`\autoChange` [*pitch* (pitch)] [*clef-1* (context modification)] [*clef-2* (context modification)] *music* (music) ⇒ music
Make voices for *music* that switch between staves automatically.

The optional argument *pitch* specifies where to switch staves; the default is `c'`. Optional arguments *clef-1* and *clef-2* specify which clefs to use; this only works for implicitly instantiated staves.

Example:

```
\autoChange d' \with { \clef alto } { g4 d' g' }
```

`\balloonGrobText` *grob-name* (symbol) *offset* (pair of numbers) *text* (markup) ⇒ music
Attach *text* to *grob-name* at offset *offset*.

Use like `\once` (page 968).

`\balloonText` *offset* (pair of numbers) *text* (markup) ⇒ post-event
Attach *text* at *offset*.

Use like `\tweak` (page 977).

`\bar` *type* (string) ⇒ music
Insert a bar line of type *type*, overriding any automatic bar lines.

`\barNumberCheck` *n* (integer) ⇒ music
Print a warning if the current bar number is not *n*.

`\beamExceptions` *music* (music) ⇒ any type
Set beam exceptions.

This function extracts a value suitable for setting `Timing.beamExceptions` from the given pattern with explicit beams in *music*. A bar check '`|`' has to be used between bars of patterns in order to reset the timing.

`\bendAfter` *delta* (real number) ⇒ post-event
Create a fall or doit of pitch interval *delta*.

`\bendHold` *mus* (music) ⇒ post-event
Set `BendSpanner.style` to `'hold` for *mus*.

`\bendStartLevel` *idx* (non-negative, exact integer) *mus* (music) ⇒ post-event
Set `Bendspanner.details.successive-level` to *idx* for *mus*.

`\bookOutputName` *newfilename* (string) ⇒ void
Direct output for the current book block to *newfilename*.

This is equivalent to setting `output-filename` in the current book's `\paper` block.

\bookOutputSuffix *newsuffix* (string) ⇒ void
: Set the output file name suffix for the current book block to *newsuffix*.

  This is equivalent to setting `output-suffix` in the current book's \paper block.

\breakAlignInsert [*positions* (vector)] *symbol-to-move* (symbol) *sorting-symbol* (symbol) *anchor-symbol* (symbol) ⇒ music
: Re-order break-align symbols.

  This function is for tasks like putting a clef change after the bar line, which would be accomplished by \breakAlignInsert `clef after staff-bar`.

  sorting-symbol can be `before` or `after`. Depending on its value, the break-align symbol *symbol-to-move* gets moved directly before/after the symbol *anchor-symbol* in the `break-align-orders` vector of `Score.BreakAlignment`.

  The available break-align symbols and their corresponding grobs are listed in Section "Grobs and their break-align symbols" dans *Manuel de notation*.

  The optional argument *positions* determines whether the specified change should become visible at the beginning/the end/the middle of a line. Possible values are: `#all-visible` (default), `#begin-of-line-invisible`, `#center-invisible`, `#end-of-line-invisible`, `#begin-of-line-visible`, `#center-visible`, `#end-of-line-visible`, `#all-invisible`.

\breathe ⇒ music
: Insert a breath mark.

\cadenzaOff ⇒ music
: Resume advancing `Timing.measurePosition`.

  This sets `Timing.timing` to `#t`.

\cadenzaOn ⇒ music
: Stop advancing `Timing.measurePosition`.

  This sets `Timing.timing` to `#f`.

  Compare to \senzaMisura (page 973).

\caesura ⇒ music
: Insert a caesura.

\chordRepeats [*event-types* (list)] *music* (music) ⇒ music
: Extend 'q' to also repeat articulation.

  This function walks through *music*, putting the notes of the previous chord into repeat chords, as well as an optional list of *event-types* such as `#'(string-number-event)`.

\clef *type* (string) ⇒ music
: Set the current clef to *type*.

\codaMark [*num* (non-negative, exact integer)] ⇒ music
: Create a coda mark.

  num may be 1 for the first mark, 2 for the second, etc., or it may be \default to use the next number in sequence automatically.

\compressMMRests *music* (music) ⇒ music
: Convert empty bars to multi-measure rests in *music*.

\contextPropertyCheck *property-path* (symbol list or symbol) [*expected-value* (any type)] ⇒ music
: Check that the context property identified by *property-path* is set to *expected-value* in that very context: being set in an enclosing context is insufficient. If *expected-value* is \default, check that the property is unset in that context.

If *property-path* does not name a context, the check occurs in the current context.

Print a warning if the requested context is not visible looking upward from the current context or if the state of the property in the requested context is unexpected.

Caveat: Like some other context-specific commands, a context given in *property-path* is ignored in certain cases, such as inside `\with`.

`\crossStaff` *notes* (music) ⇒ music
Create cross-staff stems for *notes*.

`\cueClef` *type* (string) ⇒ music
Set the current cue clef to *type*.

`\cueClefUnset` ⇒ music
Unset the current cue clef.

`\cueDuring` *what* (string) *dir* (direction) *main-music* (music) ⇒ music
Create a cue.

This function inserts the contents of quote *what* corresponding to *main-music*, in a `CueVoice` context called `cue` oriented by *dir*.

`\cueDuringWithClef` *what* (string) *dir* (direction) *clef* (string) *main-music* (music) ⇒ music
Create a cue with clef.

This function inserts the contents of quote *what* corresponding to *main-music*, in a `CueVoice` context called `cue` oriented by *dir* and using clef *clef*.

`\deadNote` *note* (music) ⇒ music
Print *note* with a cross-shaped note head.

`\defineBarLine` *bar* (string) *glyph-list* (list) ⇒ void
Define bar line settings for bar line *bar*.

The list *glyph-list* must have three entries, defining substitute glyphs for the end of a line, the beginning of a line, and a span bar, respectively. The substitute glyphs may be either strings or Booleans: `#t` calls for the same value as *bar* and `#f` calls for no glyph.

`\displayLilyMusic` [*port* (output port)] *music* (music) ⇒ music
Write LilyPond's input representation of *music*.

If *port* is omitted, the output defaults to the console (stdout).

`\displayMusic` [*port* (output port)] *music* (music) ⇒ music
Write the internal representation of *music*.

If *port* is omitted, the output defaults to the console (stdout).

`\displayScheme` [*port* (output port)] *expr* (any type) ⇒ any type
Write the internal Scheme representation of *expr*.

If *port* is omitted, the output defaults to the console (stdout).

`\dropNote` *num* (integer) *music* (music) ⇒ music
'Drop' the *num*-th note in each chord of *music*.

This function moves the affected notes down (usually by an octave) to be lower than the other notes of the chord. The position in a chord is counted downwards from the top.

The opposite function is `\raiseNote` (page 972).

`\enablePerStaffTiming` ⇒ void
Enable polymeter with unaligned measures.

This function moves the timing management from `Score` to `Staff`-like contexts. This is done by removing the `Timing_translator` from `Score`, and adding it to all contexts having the `Staff` alias.

Use this within an output definition.

`\endSpanners` *music* (music) ⇒ music
Terminate spanners.

This function prematurely ends all spanners in *music* by inserting an end-spanner event at the end of the argument, without the need of specific end-spanner commands.

`\eventChords` *music* (music) ⇒ music
Compatibility function: Handle isolated rhythmic events in *music*.

Use this to wrap `EventChord` around isolated rhythmic events occuring since version 2.15.28, after expanding repeat chords '`q`'.

Not needed for new code.

`\featherDurations` *scale* (non-negative rational, fraction, or moment) *music* (music) ⇒ music
Adjust feathered beam durations in *music* by *scale*.

`\finger` *finger* (index or markup) ⇒ post-event
Apply *finger* as a fingering indication.

`\fixed` *pitch* (pitch) *music* (music) ⇒ music
Use the octave of *pitch* as the default octave for *music*.

`\footnote` [*mark* (markup)] *offset* (pair of numbers) *footnote* (markup) *item* (symbol list or music) ⇒ music
Make the markup *footnote* a footnote on *item*.

The footnote is marked with a markup *mark* moved by *offset* with respect to the marked music.

If *mark* is not given or specified as `\default`, it is replaced by an automatically generated sequence number. If *item* is a symbol list of form *`Grob`* or *`Context.Grob`*, then grobs of that type are marked at the current time step in the given context (default `Bottom`).

If *item* is music, the music gets a footnote attached to a grob immediately attached to the event, like `\tweak` does. For attaching a footnote to an *indirectly* caused grob, write `\single\footnote`, use *item* to specify the grob, and follow it with the music to annotate.

Like with `\tweak`, if you use a footnote on a following post-event, the `\footnote` command itself needs to be attached to the preceding note or rest as a post-event with '`-`'.

`\grace` *music* (music) ⇒ music
Insert *music* as grace notes.

`\grobdescriptions` *descriptions* (list) ⇒ any type
Create a context modification from *descriptions*.

The argument is a list in the format of `all-grob-descriptions`.

`\harmonicByFret` *fret* (number) *music* (music) ⇒ music
Convert *music* into mixed harmonics.

The resulting notes resemble harmonics played on a fretted instrument by touching the strings at *fret*.

\harmonicByRatio *ratio* (number) *music* (music) ⇒ music
: Convert *music* into mixed harmonics.

  The resulting notes resemble harmonics played on a fretted instrument by touching the strings at the point given through *ratio*.

\harmonicNote *note* (music) ⇒ music
: Print *note* with a diamond-shaped note head.

\harmonicsOn ⇒ music
: Set the default note head style to a diamond-shaped style.

\hide *item* (symbol list or music) ⇒ music
: Make *item* invisible while still retaining its dimensions.

  If *item* is a symbol list of form *GrobName* or *Context.GrobName*, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

  This function sets *item*'s `transparent` property to `#t`.

\incipit [*offset* (number)] *incipit-music* (music) ⇒ music
: Output *incipit-music* as an incipit.

  incipit-music is typeset within a `MensuralStaff` context; the result is positioned before the main staff (as part of an `InstrumentName` grob) to indicate the music's original notation. The optional argument *offset* specifies how much the incipit should be moved to the left (default value is 0).

  In the special case that *incipit-music* has the form `\new` *xxx* ..., where *xxx* is a context type not accepted by `MensuralStaff`, it is taken directly.

\inherit-acceptability *to* (symbol) *from* (symbol) ⇒ void
: Make two contexts 'compatible'.

  When used in an output definition, modify all context definitions such that context *to* is accepted as a child by all contexts that also accept *from*.

\initialContextFrom *music* (music) ⇒ music
: Enter the initial context of *music* and ignore the rest of it.

  This is useful for prepending music while preserving the influence of the original music on the context.

  Example:

```
{
  \initialContextFrom \originalMusic
  \prependedMusic
  \originalMusic
  \appendedMusic
}
```

\inStaffSegno ⇒ music
: Put the segno variant `varsegno` at this position into the staff.

  This is compatible with the repeat command.

\instrumentSwitch *name* (string) ⇒ music
: Switch instrument to *name*.

  name must have been predefined with function `\addInstrumentDefinition` (page 959).

  This function is deprecated.

`\inversion` *around* (pitch) *to* (pitch) *music* (music) ⇒ music
Invert *music* about *around* and transpose from *around* to *to*.

`\invertChords` *num* (integer) *music* (music) ⇒ music
Invert any chords in *music* into their *num*-th position.

Chord inversions may be directed downwards using negative integers.

`\jump` *text* (markup) ⇒ music
Use *text* to mark a point of departure, e.g., 'Gavotte I D.C.'.

`\keepWithTag` *tags* (symbol list or symbol) *music* (music) ⇒ music
Keep tagged music.

This function only includes elements of *music* that are tagged with one of the tags in *tags*. *tags* may be either a single symbol or a list of symbols.

Each tag may be declared as a member of at most one tag group (defined with `\tagGroup`). If none of a *music* element's tags share a tag group with one of the specified *tags*, the element is retained.

`\key` [*tonic* (pitch)] [*pitch-alist* (list of number pairs)] ⇒ music
Set key to *tonic* and scale *pitch-alist*.

If both arguments are omitted (i.e., replaced by `\default`), just generate a `KeyChangeEvent`, which prints the current key signature again.

`\killCues` *music* (music) ⇒ music
Remove cue notes from *music*.

`\label` *label* (symbol) ⇒ music
Create *label* as a referrable label.

The value stored in *label* is the page number, which can be extracted with the `\page-ref` markup command later on.

`\language` *language* (string) ⇒ void
Set note names for language *language*.

This function updates the Scheme variables `pitchnames` and `input-language`. The former is an alist that holds the pitches of all note names for the selected language; the latter holds the language name itself (as a symbol).

`\languageRestore` ⇒ void
Restore the previously-saved `pitchnames` alist.

`\languageSaveAndChange` *language* (string) ⇒ void
Save current `pitchnames` alist and change note names to *language*.

`\ligature` *music* (music) ⇒ music
Make a ligature from Gregorian Chant *music*.

This is equivalent to enclosing *music* with `\[` and `\]`.

`\magnifyMusic` *mag* (positive number) *music* (music) ⇒ music
Magnify the size of *music* by factor *mag*.

This doesn't change the staff size. Stems, beams, slurs, ties, and horizontal spacing are adjusted automatically.

`\magnifyStaff` *mag* (positive number) ⇒ music
Change the staff size by factor *mag*.

This adjusts notation size and horizontal spacing automatically.

`\makeClusters` *arg* (music) ⇒ music
Display chords in *arg* as clusters.

\makeDefaultStringTuning *symbol* (symbol) *pitches* (list) ⇒ void
: Define string tuning *symbol* by a list of *pitches*.

  symbol also gets registered in `defaultStringTunings` for documentation purposes.

\mark [*label* (index or markup)] ⇒ music
: Create a rehearsal mark.

  If *label* is an integer, create the rehearsal mark for the given sequence number. If *label* is `\default`, create the next sequential rehearsal mark. If *label* is markup, use it for the mark.

\markupMap *path* (symbol list or symbol) *markupfun* (markup-function) *music* (music) ⇒ music
: Apply *markupfun* to property *path* in *music*.

  Argument *path* is either of the form *`property`* or *`MusicExpression.property`*. If *`MusicExpression`* is not given, *markupfun* gets applied to all properties called *`property`*, otherwise it is restricted to *`MusicExpression`* events. If *`property`* is not a markup, it is ignored.

  In the following example, both the tempo indication and the bowing instruction are printed in red. If you replace `text` with `TempoChangeEvent.text`, only the tempo indication changes the color.

```
\markupMap
  text
  \markup \with-color #red \etc
  { \tempo "Largo" g'2_"arco" c'' }
```

\measure *music* (music) ⇒ music
: Create a complete measure from *music*.

  This is equivalent to `\measureRemainder` *`music`*, but with a bar check at the beginning as well as the end.

\measureRemainder *music* (duration-or-music) ⇒ music
: End the current measure after *music*.

  The effect is equivalent to this:

```
{
  \initialContextFrom %{ copy of music %}
  \setMeasureLengthFromHere %{ duration of music %}
  music |
  \setDefaultMeasureLength
}
```

  Passing a duration as *music* (e.g., `\measureRemainder 2`) is reserved for future use.

  See `\setMeasureLengthFromHere` (page 974) for more information.

  Compare to `\premeasure` (page 971).

\modalInversion *around* (pitch) *to* (pitch) *scale* (music) *music* (music) ⇒ music
: Invert *music* about *around* using *scale* and transpose from *around* to *to*.

\modalTranspose *from* (pitch) *to* (pitch) *scale* (music) *music* (music) ⇒ music
: Transpose *music* from pitch *from* to pitch *to* using *scale*.

\musicLength *music* (music) ⇒ any type
: Return the duration of *music* as a `ly:moment`.

  Examples:

```
\musicLength 8 ⇒ #(ly:make-moment 1/8)
\musicLength {8. 8 8} ⇒ #(ly:make-moment 7/16)
```

`\musicMap` *proc* (procedure) *mus* (music) ⇒ music
: Apply *proc* to *mus* and all of the music it contains.

`\noPageBreak` ⇒ music
: Forbid a page break.

  May be used at top level (i.e., between scores or markups), or inside a score.

`\noPageTurn` ⇒ music
: Forbid a page turn.

  May be used at top level (i.e., between scores or markups), or inside a score.

`\noStanza` ⇒ music
: Forget the current stanza setting, thereby stopping any stanza reminders derived from it.

`\octaveCheck` *pitch* (pitch) ⇒ music
: Do an octave check.

  This prints a warning if the interval between the previous note and *pitch* is not within a fourth.

`\offset` *property* (symbol list or symbol) *offsets* (any type) *item* (key list or music) ⇒ music
: Offset the default value of *property* of *item* by *offsets*.

  If *item* is a string, the result is an override for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

`\omit` *item* (symbol list or music) ⇒ music
: Omit *item* without taking up space.

  If *item* is a symbol list of form *`GrobName`* or *`Context.GrobName`*, the result is an override for the grob name specified by it. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied to it.

  This function sets *item*'s `stencil` property to `#f`.

`\once` *music* (music) ⇒ music
: Set property `once` to `#t` on all layout instruction events in *music*.

`\ottava` *octave* (integer) ⇒ music
: Set the octavation to *octave*.

  A positive value *n* indicates *n* octaves higher; a negative value *n* octaves lower, and value 0 means no octavation.

`\overrideProperty` *grob-property-path* (list of indices or symbols) *value* (any type) ⇒ music
: Set the grob property specified by *grob-property-path* to *value*.

  grob-property-path is a symbol list of the form *`Context.GrobName.property`* or *`GrobName.property`*, possibly with subproperties given as well.

  As opposed to `\override`, which overrides the context-dependent defaults with which a grob is created, this command uses `Apply_output_engraver` at the grob acknowledge stage. This may be necessary for overriding values set after the initial grob creation.

`\overrideTimeSignatureSettings` *time-signature* (boolean-or-fraction) *beat-base* (positive exact rational, fraction (as pair), moment, or +inf.0) *structure* (a number list or a list of them) *beam-exceptions* (list) ⇒ music
: Override time signature settings.

  This function sets the `timeSignatureSettings` entry for *time-signature* to use *beat-base*, *structure*, and *beam-exceptions*.

beat-base and *beam-exceptions* are used as-is for `beatBase` and `beamExceptions`.

structure is used to derive values for `beatStructure` and `submeasureStructure`, as described for `\time` (page 976).

`\pageBreak` ⇒ music
Force a page break.

May be used at top-level (i.e., between scores or markups), or inside a score.

`\pageTurn` ⇒ music
Force a page turn.

May be used at top-level (i.e., between scores or markups), or inside a score.

`\palmMute` *note* (music) ⇒ music
Print *note* with a triangle-shaped note head.

`\palmMuteOn` ⇒ music
Set the default note head style to a triangle-shaped style.

`\parallelMusic` *voice-ids* (list) *music* (music) ⇒ void
Define parallel music sequences.

Within *music*, parallel music sequences are separated by '`|`' characters. The sequences are assigned to the LilyPond music identifiers provided in *voice-ids*.

For example, this code

```
\parallelMusic A,B,C {
  c c | d d | e e |
  d d | e e | f f |
}
```

is equivalent to

```
A = { c c | d d }
B = { d d | e e }
C = { e e | f f }
```

The last bar checks in a sequence are not copied to the result in order to facilitate ending the last entry at non-bar boundaries.

`\parenthesize` *arg* (symbol list or music) ⇒ music
Tag *arg* to be parenthesized.

arg may be either a music event or a grob path.

`\partCombine` [*chord-range* (pair of numbers)] *part1* (music) *part2* (music) ⇒ music
Combine two parts into a single staff.

This takes the music in *part1* and *part2* and returns a music expression containing simultaneous `Voice` contexts (called `one` for the upper and `two` for the lower voice). Where appropriate, *part1* and *part2* are combined into a single voice (called `shared` or `solo`, depending on context).

Optional argument *chord-range* is a pair `(start . stop)` that defines the range in which the two voices are printed as chords (or unison); the default value is `(0 . 8)`, which means that intervals up to and including a ninth are unified.

`\partCombineDown` [*chord-range* (pair of numbers)] *part1* (music) *part2* (music) ⇒ music
Combine two parts into a single staff with all stems downwards.

For details, see `\partCombine` (page 969).

`\partCombineForce` [*type* (symbol)] ⇒ music
Override the part-combiner mode with *type*.

The following table gives the possible values for *type*, together with the corresponding shorthand functions.

apart	\partCombineApart
chords	\partCombineChords
unisono	\partCombineUnisono
solo1	\partCombineSoloI
solo2	\partCombineSoloII
\default	\partCombineAutomatic

\partCombineUp [*chord-range* (pair of numbers)] *part1* (music) *part2* (music) ⇒ music
Combine two parts into a single staff with all stems upwards.

For details, see \partCombine (page 969).

\partial *dur* (duration) ⇒ music
End the current measure after *dur*.

Adjust Timing.measurePosition to end the current measure at *dur* past the point of use. As a special case, when used at the start, create an anacrusis before the first measure.

Notational elements that communicate metric structure (e.g., beam breaks) will be engraved appropriately for the final *dur* part of a measure. Compare to \setMeasureLengthFromHere (page 974).

The calculation requires Timing.measureLength to be finite, so \partial cannot be used in a senza-misura passage.

See also \premeasure (page 971).

\phrasingSlurDashPattern *dash-fraction* (number) *dash-period* (number) ⇒ music
Set up a custom dash pattern style for phrasing slurs.

dash-fraction gives the size of one dash relative to *dash-period*; *dash-period* is the length of one dash plus one space. LilyPond adjusts *dash-period* to produce symmetrical output.

More complex patterns can be achieved by directly manipulating the PhrasingSlur.dash-definition property.

\pitchedTrill *main-note* (music) *secondary-note* (music) ⇒ music
Print a pitched trill.

main-note is the main note of the trill; *secondary-note* gets printed as a stemless note head in parentheses.

\pointAndClickOff ⇒ void
Suppress links to LilyPond source code in music output.

\pointAndClickOn ⇒ void
Generate links to LilyPond source code in music output.

This enables the creation of code in a PDF or SVG output file to reference the originating LilyPond source code (i.e., file name, line number, and column). This is helpful when developing a score; however, the output file becomes much larger.

\pointAndClickTypes *types* (symbol list or symbol) ⇒ void
Generate point-and-click info for music of type *types* only.

types is a single music expression (such as #'note-event) or a list of music expressions.

\polymetric [*time-signature-command* (music)] ⇒ music
Apply *time-signature-command* to the local context rather than to Timing.

Example:

```
\context Staff \polymetric \time 6/8
```

The local measure must align with the reference measure defined in `Timing`. So that explicit changes to `Timing.measureLength` (e.g., for ad-hoc irregular measures) remain visible, this command does not set the `measureLength` property in the local context.

A nominally incompatible local measure can be fitted to the controlling `Timing` measure by applying `\scaleDurations` (page 973) to this command. That adjusts the value of the internal property `meterScalingFactor` in the local context.

To unset the local properties and resume using the values from the `Timing` context, use `\polymetric \default`.

`\popContextProperty` *path* (list of indices or symbols) ⇒ music
Pop value of context property *path* from stack and set it.

This is the opposite to function `\pushContextProperty` (page 972).

`\preBend` *mus* (music) ⇒ post-event
Set `BendSpanner.style` to `'pre-bend` for *mus*.

`\preBendHold` *mus* (music) ⇒ post-event
Set `BendSpanner.style` to `'pre-bend-hold` for *mus*.

`\premeasure` *music* (duration-or-music) ⇒ music
Create a lead-in partial measure from *music*.

The effect is equivalent to this:

```
{
  \initialContextFrom %{ copy of music %}
  \partial %{ duration of music %}
  music |
}
```

Passing a duration as *music* (e.g., `\premeasure 2`) is reserved for future use.

Compare to `\measureRemainder` (page 967).

`\propertyOverride` *grob-property-path* (list of indices or symbols) *value* (any type) ⇒ music
Set the grob property specified by *grob-property-path* to *value*.

grob-property-path is a symbol list of the form *`Context.GrobName.property`* or *`GrobName.property`*, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\override` command.

`\propertyRevert` *grob-property-path* (list of indices or symbols) ⇒ music
Revert the grob property specified by *grob-property-path* to its previous value.

grob-property-path is a symbol list of the form *`Context.GrobName.property`* or *`GrobName.property`*, possibly with subproperties given as well. This music function is mostly intended for use from Scheme as a substitute for the built-in `\revert` command.

`\propertySet` *property-path* (symbol list or symbol) *value* (any type) ⇒ music
Set the context property specified by *property-path* to *value*.

This music function is mostly intended for use from Scheme as a substitute for the built-in `\set` command.

`\propertyTweak` *prop* (key list or symbol) *value* (any type) *item* (key list or music) ⇒ music
Add a tweak to *item*, usually music.

This function sets the value of property *prop* to *value*; it generally behaves like `\tweak` but will turn into an `\override` when *item* is a symbol list. In that case, *item* specifies the grob path to override. This is mainly useful when using `\propertyTweak` as as a component for building other functions like `\omit`. It is not the default behavior for `\tweak` since many input strings in `\lyricmode` can serve equally as music or as symbols, which causes surprising behavior when tweaking lyrics using the less specific semantics of `\propertyTweak`.

prop can contain additional elements in which case a nested property (inside of an alist) is tweaked.

`\propertyUnset` *property-path* (symbol list or symbol) ⇒ music
Unset the context property specified by *property-path*.

This music function is mostly intended for use from Scheme as a substitute for the built-in `\unset` command.

`\pushContextProperty` *path* (list of indices or symbols) ⇒ music
Push the current value of context property *path* to a stack.

The property can later be restored to the saved value with function `\popContextProperty` (page 971).

`\pushToTag` *tag* (symbol) *more* (music) *music* (music) ⇒ music
Add *more* to the front of *music* tagged with *tag*.

A post-event can be added to the articulations of rhythmic events or chords; other expressions may be added to chords, sequential or simultaneous music.

`\pushToTagMarkup` *tag* (symbol) *more* (markup) *music* (music) ⇒ music
Prepend *more* to every markup in *music* tagged with *tag*.

`\quoteDuring` *what* (string) *main-music* (music) ⇒ music
Indicate a section of music to be quoted.

what indicates the name of the quoted voice specified with `\addQuote` (page 959). *main-music* is used to indicate the length of music to be quoted; it usually contains spacers or multi-measure rests.

`\raiseNote` *num* (integer) *music* (music) ⇒ music
'Raise' the *num*-th note in each chord of *music*.

This function moves the affected notes up (usually by an octave) to be higher than the other notes of the chord. The position in a chord is counted upwards from the bottom.

The opposite function is `\dropNote` (page 963).

`\reduceChords` *music* (music) ⇒ music
Reduce chords contained in *music* to single notes.

This is intended mainly for reusing music in a `RhythmicStaff` context. It does not reduce simultaneous music.

`\relative` [*pitch* (pitch)] *music* (music) ⇒ music
Make *music* relative to *pitch*.

If *pitch* is omitted, the first note in *music* is given in absolute pitch.

`\removeFromTagGroup` *tag-group* (symbol list) *tags* (symbol list) ⇒ void
Remove *tags* from the tag group identified by *tag-group*.

`\removeWithTag` *tags* (symbol list or symbol) *music* (music) ⇒ music
Remove elements of *music* that are tagged with one of the tags in *tags*.

tags may be either a single symbol or a list of symbols.

\replaceWithTag *tag* (symbol) *replacement* (music) *music* (music) ⇒ music
Replace tagged elements in *music*.

Everything tagged with *tag* (including the tagging itself) gets replaced with *replacement*.

\replaceWithTagMarkup *tag* (symbol) *replacement* (markup) *music* (music) ⇒ music
Replace tagged markup in *music*.

All markup tagged with *tag* (including the tagging itself) gets replaced with *replacement*.

\resetRelativeOctave *pitch* (pitch) ⇒ music
Set the octave inside a \relative section to *pitch*.

\resetTagGroup *tag-group* (symbol list) ⇒ void
Reset the tag group equal to *tag-group*.

\resetTagGroups ⇒ void
Reset all tag groups previously created with \tagGroup.

\responsum *music* (music) ⇒ music
Prepend character U+211F (RESPONSE) to the lyrics represented by *music*.

\retrograde *music* (music) ⇒ music
Return *music* in reverse order.

\revertTimeSignatureSettings *time-signature* (pair) ⇒ music
Revert timeSignatureSettings for time signatures equal to *time-signature*.

\rightHandFinger *finger* (index or markup) ⇒ post-event
Apply *finger* as a right-hand fingering indication.

\scaleDurations *fraction* (non-negative rational, fraction, or moment) *music* (music) ⇒ music
Multiply the duration of events in *music* by *fraction*.

\sectionLabel *text* (markup) ⇒ music
Mark the beginning of a named passage with *text*, e.g., « Coda ».

This is well suited for use at a section division created with \section, but it does not imply \section and may be used alone.

\segnoMark [*num* (non-negative, exact integer)] ⇒ music
Create a segno mark (or bar line).

num may be 1 for the first segno, 2 for the second, etc., or it may be \default to use the next number in sequence automatically.

If the segnoStyle context property is 'bar-line, a segno bar line is created instead of a segno mark.

\senzaMisura ⇒ music
Cancel the time signature; begin free time.

This sets Timing.timeSignature to #f.

This sets Timing.measureLength to +inf.0, allowing Timing.measurePosition to advance indefinitely without starting a new measure.

Beyond this point, ending the current measure requires setting Timing.measureLength back to a finite value *before* (not *at*) the transition to the next measure. See \measure (page 967) and related functions for assistance.

Compare to \cadenzaOn (page 962).

\setDefaultMeasureLength ⇒ music
Set Timing.measureLength from Timing.timeSignature.

`\setMeasureLengthFromHere` *dur* (duration) ⇒ music
End the current measure after *dur*.

Adjust `Timing.measureLength` to end at *dur* past the point of use. The resulting measure length remains in effect until it is changed by another command.

Notational elements that communicate metric structure (e.g., beam breaks) will be engraved appropriately to continue the measure. Compare to `\partial` (page 970).

See also `\measure` (page 967) and `\measureRemainder` (page 967).

`\shape` *offsets* (list) *item* (key list or music) ⇒ music
Offset control points of *item* by *offsets*.

offsets is a list of number pairs `(x . y)` or a list of such lists. Each pair represents an offset to a control point. The 'y' value of each pair is scaled by staff space.

If *item* is a string, the result is `\once\override` for the specified grob type. If *item* is a music expression, the result is the same music expression with an appropriate tweak applied.

`\shiftDurations` *dur* (integer) *dots* (integer) *arg* (music) ⇒ music
Change duration of *arg*.

This function walks over all durations and dot counts in *arg*, adding *dur* to the durations and *dots* to the dot counts.

`\single` *overrides* (music) *music* (music) ⇒ music
Convert *overrides* to tweaks and apply them to *music*.

This does not convert `\revert`, `\set` or `\unset`.

`\skip` *arg* (duration-or-music) ⇒ music
Skip over *arg*, which may be music or a duration.

`\slashedGrace` *music* (music) ⇒ music
Create slashed graces from *music*.

This produces slashes through stems, but no slur.

`\slurDashPattern` *dash-fraction* (number) *dash-period* (number) ⇒ music
Set up a custom dash pattern style for slurs.

dash-fraction gives the size of one dash relative to *dash-period*; *dash-period* is the length of one dash plus one space. LilyPond adjusts *dash-period* to produce symmetrical output.

More complex patterns can be achieved by directly manipulating the `Slur .dash-definition` property.

`\staffHighlight` *color* (color) ⇒ music
Start a highlight with color *color*.

`\stanza` *text* (markup) ⇒ music
Set and display the stanza number *text* in lyrics.

`\startGradualTempoChange` [*text* (markup)] ⇒ music
Start an accelerando or ritardando.

Begin a gradual departure from the tempo most recently set with `\tempo`. The gradual change extends to the next `\stopGradualTempoChange` or `\tempo` command. When `\stopGradualTempoChange` is used, it defines the target tempo; otherwise, `\tempo` defines it.

The *text* argument is a placeholder. Only `\default` is accepted. It is intended eventually to serve the same purpose as the text argument to `\tempo`.

\stopGradualTempoChange [*text* (markup)] *tempo-unit* (duration) *metronome-count* (number) ⇒ music

End an accelerando or ritardando.

End a gradual tempo change at the tempo defined by *tempo-unit* and *metronome-count*. Unless there is a simultaneous \tempo command setting a new tempo, resume the tempo most recently set with \tempo.

The *text* argument is a placeholder. Only \default is accepted. It is intended eventually to serve the same purpose as the text argument to \tempo.

\storePredefinedDiagram *fretboard-table* (hash table) *chord* (music) *tuning* (pair) *diagram-definition* (string or pair) ⇒ void

Add a predefined fret diagram to *fretboard-table*.

It is defined by *diagram-definition* for the chord pitches *chord* and the string tuning *tuning*.

\stringTuning *chord* (music) ⇒ any type

Convert *chord* to a string tuning.

chord must be in absolute pitches and should have the highest string number (generally the lowest pitch) first.

\styledNoteHeads *style* (symbol) *heads* (symbol list or symbol) *music* (music) ⇒ music

Set *heads* in *music* to *style*.

\tabChordRepeats [*event-types* (list)] *music* (music) ⇒ music

Extend 'q' to also repeat string and fingering information.

This function walks through *music* putting the notes, fingerings and string numbers of the previous chord into repeat chords, as well as an optional list of *event-types* such as #'(articulation-event).

\tabChordRepetition ⇒ void

Include the string and fingering information in a chord repetition.

This function is deprecated; use \tabChordRepeats instead.

\tag *tags* (symbol list or symbol) *music* (music) ⇒ music

Tag *music* with *tags*.

This function adds the single symbol or symbol list *tags* to the tags property of *music* and returns the result.

\tagGroup *tags* (symbol list) ⇒ void

Define a tag group comprising the symbols in the symbol list *tags*.

Tag groups must not overlap.

\tagGroupRef *tags* (symbol list) ⇒ any type

Define a tag group comprising the symbols in the symbol list *tags*.

It returns the created tag group for further reference.

\temporary *music* (music) ⇒ music

Make \override reversible with \revert.

This function makes any \override in *music* replace an existing grob property value only temporarily, restoring the old value when a corresponding \revert is executed. This is achieved by clearing the pop-first property normally set on \overrides.

An \override/\revert sequence created by using \temporary and \undo on the same music containing overrides will cancel out perfectly or cause a warning.

Non-property-related music is ignored, warnings are generated for any property-changing music that isn't an \override.

`\textEndMark` *text* (markup) ⇒ music
: Create a right-aligned text mark using *text*.

`\textMark` *text* (markup) ⇒ music
: Create a (left-aligned) text mark using *text*.

`\tieDashPattern` *dash-fraction* (number) *dash-period* (number) ⇒ music
: Set up a custom dash pattern style for ties.

  dash-fraction gives the size of one dash relative to *dash-period*; *dash-period* is the length of one dash plus one space. LilyPond adjusts *dash-period* to produce symmetrical output.

  More complex patterns can be achieved by directly manipulating the `Tie .dash-definition` property.

`\time` [*structure* (a number list or a list of them)] *time-sig* (time signature) ⇒ music
: Set the time signature to *time-sig*.

  When *structure* is a plain list of numbers, it is used as-is for `beatStructure`, and `submeasureStructure` is left at the default.

  When *structure* is a list of lists, `beatStructure` is derived by discarding the grouping, and `submeasureStructure` is derived by summing each group. For example, a *structure* of `'((1 2) (4 8))` yields a `beatStructure` of `'(1 2 4 8)` and a `submeasureStructure` of `'(3 12)`.

  time-sig may be a fraction, e.g., `3/4`.

  time-sig may also describe a complex time signature as a Scheme expression. Fractions are represented as pairs, (*`numerator . denominator`*), where the denominator is always a number. The numerator is one number or a list of two or more numbers. A list represents concatenation.

  For example, a time signature of (3+1)/8 + 2/4 can be created with `\time #'(((3 1) . 8) (2 . 4))`

`\timeAbbrev` *time-sig* (pair) ⇒ music
: Set the time signature to *time-sig*.

  This is like `\time` *time-sig*, except that it allows abbreviating fractions as lists. For example, a time signature of (3+1)/8 + 2/4 can be created with `\timeAbbrev #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 with either `\timeAbbrev #'((3 2 8))` or the shorter version `\timeAbbrev 3,2,8`.

  This is for backward compatibility. Using `\time` (page 976) instead is recommended.

`\times` *fraction* (fraction, as pair) *music* (music) ⇒ music
: Scale *music* in time by *fraction*.

`\tocItem` [*label* (symbol list or symbol)] *text* (markup) ⇒ music
: Add *text* as an entry to the table of contents.

  This uses the `tocItemMarkup` paper variable markup for formatting and assigns it to *label* if one is provided. If a hierarchy of labels is given, make the current item a child of the corresponding objects.

`\transpose` *from* (pitch) *to* (pitch) *music* (music) ⇒ music
: Transpose *music* from pitch *from* to pitch *to*.

`\transposedCueDuring` *what* (string) *dir* (direction) *pitch* (pitch) *main-music* (music) ⇒ music
: Create a transposed cue.

This function inserts notes from the part *what* into a `CueVoice` context called `cue`, using the transposition defined by *pitch*. This happens simultaneously with *main-music*, which is usually a rest. The argument *dir* determines whether the cue notes should be notated as a first or second voice.

`\transposition` *pitch* (pitch) ⇒ music
: Set instrument transposition to *pitch*.

`\tuplet` *ratio* (fraction, as pair) [*tuplet-span* (duration)] *music* (music) ⇒ music
: Scale the given *music* to tuplets.

  ratio is a fraction that specifies how many notes are played in place of the nominal value: it will be 3/2 for triplets, namely three notes being played in place of two.

  If the optional duration *tuplet-span* is specified, it is used instead of `tupletSpannerDuration` for grouping the tuplets. For example,

  ```
  \tuplet 3/2 4 { c8 c c c c c }
  ```

  results in two groups of three tuplets, each group lasting for a quarter note.

`\tupletSpan` [*tuplet-span* (duration)] ⇒ music
: Set `tupletSpannerDuration` to the duration *tuplet-span*.

  This context property is the length into which `\tuplet` without an explicit tuplet span argument of its own will group its tuplets. To revert to the default of not subdividing the contents of a `\tuplet` command without an explicit tuplet span argument, use

  ```
  \tupletSpan \default
  ```

`\tweak` *prop* (key list or symbol) *value* (any type) *music* (music) ⇒ music
: Add a tweak to *music*.

  Layout objects created by *music* get their property *prop* set to *value*. If *prop* has the form *`Grob.property`*, like with

  ```
  \tweak Accidental.color #red cis'
  ```

  an indirectly created grob (`Accidental` is caused by `NoteHead`) can be tweaked; otherwise only directly created grobs are affected.

  prop can contain additional elements in which case a nested property (inside of an alist) is tweaked.

  If *music* is an `event-chord`, every contained `rhythmic-event` is tweaked instead.

`\undo` *music* (music) ⇒ music
: Convert `\override` and `\set` in *music* to `\revert` and `\unset`, respectively.

  Any reverts and unsets already in *music* cause a warning. Non-property-related music is ignored.

`\unfolded` *music* (music) ⇒ music
: Mask *music* until the innermost enclosing repeat is unfolded.

`\unfoldRepeats` [*types* (symbol list or symbol)] *music* (music) ⇒ music
: Unfold `\repeat`.

  This forces `\repeat volta`, `\repeat tremolo` or `\repeat percent` commands in *music* to be interpreted as `\repeat unfold`, if specified in the optional symbol-list *types*. The default for *types* is an empty list, which forces any of those commands in *music* to be interpreted as `\repeat unfold`. Possible entries are `volta`, `tremolo` or `percent`. Multiple entries are possible.

`\verboseBarNumbers` ⇒ music
: Print bar numbers at every bar line and line break.

This function is a concise way to add detail to documentation examples. The output is subject to change without notice.

See Section “Bar_number_engraver” dans *Référence des propriétés internes*.

`\versus` *music* (music) ⇒ music
Prepend character U+2123 (VERSICLE) to the lyrics represented by *music*.

`\voices` *ids* (list of indices or symbols) *music* (music) ⇒ music
Specify voice order in simultaneous music.

This takes the key list *ids* of numbers (indicating the use of ‘`\voiceOne`’...) or symbols (indicating voice names, typically converted from strings by argument list processing) and assign the following \\-separated music in *music* to contexts according to that list. Named rather than numbered contexts can be used for continuing one voice (for the sake of spanners and lyrics), usually requiring a `\voiceOne`-style override at the beginning of the passage and a `\oneVoice` override at its end.

The default

```
<< ... \\ ... \\ ... >>
```

construct would correspond to

```
\voices 1,2,3 << ... \\ ... \\ ... >>
```

`\void` *arg* (any type) ⇒ void
Accept a Scheme argument *arg* and return a void expression.

Use this if you want to have a Scheme expression evaluated because of its side effects but its return value being ignored.

`\volta` *volta-numbers* (number list) *music* (music) ⇒ music
Mark *music* as being limited to the volte given in *volta-numbers*.

This gets used when the innermost enclosing repeat is unfolded. Volta number begins at 1 and increases by 1 with each repetition.

`\vshape` *offsets* (list) *item* (key list or music) ⇒ music
Like `\shape`, but additionally show control points for ease of tweaking.

`\withMusicProperty` *sym* (symbol) *val* (any type) *music* (music) ⇒ music
Set music property *sym* to *val* in *music*.

`\withRelativeDir` *file-name* (string) ⇒ any type
Prepend directory of current input file to string *file-name*.

Use this for markup commands that include files, and where such files should be found relative to the input file. Example:

```
\markup { \image #X #3 \withRelativeDir "test.png" }
```
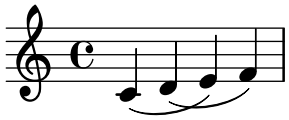
`\xNote` *note* (music) ⇒ music
Print *note* with a cross-shaped note head.

`\=` *id* (index or symbol) *event* (post-event) ⇒ post-event
Assign an ID to a spanner or an item.

This sets the `spanner-id` or `id` property of *event* to the given *id*, which is a non-negative integer or a symbol.

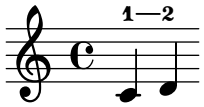
For spanners this can be used to tell LilyPond how to connect overlapping or parallel slurs or phrasing slurs within a single `Voice` context.

```
\fixed c' { c\=1( d\=2( e\=1) f\=2) }
```



For items this can be used, for example, to tell LilyPond how to connect a `FingerGlideSpanner` with non-matching fingers.

```
\fixed c' { c\glide \= #'foo -1 d\= #'foo -2 }
```



\% *count* (number) *music* (music) ⇒ music
: This is the same as `\repeat percent` *count music*.

  See `\repeat`.

* *count* (number) *music* (music) ⇒ music
: This is the same as `\repeat unfold` *count music*.

  See `\repeat`.

\~ *music* (music) ⇒ music
: Insert pes or flexa between previous and next note.

B.22 Identificateurs de modification de contexte

Les commandes suivantes permettent de modifier des contextes au sein d'un bloc `\layout` ou `\with`.

`\EnableGregorianDivisiones`
: Configure division commands such as `\section` to create `Divisio` grobs rather than `BarLine` grobs. This does not affect measure bar lines or the properties of the grobs themselves.

  - Sets translator property `caesuraTypeTransform` to `caesura-to-divisio`.
  - Sets translator property `doubleRepeatBarType` to `'()`.
  - Sets translator property `endRepeatBarType` to `'()`.
  - Sets translator property `fineBarType` to `""`.
  - Sets translator property `sectionBarType` to `""`.
  - Sets translator property `startRepeatBarType` to `'()`.
  - Sets translator property `underlyingRepeatBarType` to `""`.
  - Sets translator property `doubleRepeatSegnoBarType` to `"S-||"`.
  - Sets translator property `endRepeatSegnoBarType` to `"S-||"`.
  - Sets translator property `fineSegnoBarType` to `"S-||"`.
  - Sets translator property `fineStartRepeatSegnoBarType` to `"S-||"`.
  - Sets translator property `segnoBarType` to `"S-||"`.
  - Sets translator property `startRepeatSegnoBarType` to `"S-||"`.

`\RemoveAllEmptyStaves`
: Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`, including those in the first system.

  - Sets grob property `remove-empty` in Section "VerticalAxisGroup" dans *Référence des propriétés internes* to `#t`.

- Sets grob property `remove-first` in Section "VerticalAxisGroup" dans *Référence des propriétés internes* to `#t`.

`\RemoveEmptyStaves`
: Remove staves which are considered to be empty according to the list of interfaces set by `keepAliveInterfaces`.

  - Sets grob property `remove-empty` in Section "VerticalAxisGroup" dans *Référence des propriétés internes* to `#t`.

B.23 Variables pour format imprimable

Les variables suivantes concernent les sortie imprimables et sont prédéfinies par LilyPond. Les valeurs susceptibles d'être mises à l'échelle selon le format de papier sont présentées pou la taille par défaut, soit A4. Les variables du style `xxx-default` ici mentionnées sont aussi des variables de papier.

Certaines valeurs comme `mm` or `pt` sont des constantes et normalement en lecture seule ; les modifier pourrait grandement perturber LilyPond.

La plupart de ces variables peuvent se définir dans un bloc `\paper` ou `\layout`, voir Section 27.1 [Le bloc `\layout`], page 705.

`annotate-spacing` (boolean)
: If this value is set to `#t`, dimensions of vertical layout variables that may be altered for page formatting are graphically displayed. Default: `#f`.

`auto-first-page-number` (boolean)
: The page-breaking algorithm is affected by the first page number being odd or even. If set to `#t`, the algorithm decides whether to start with an odd or even number. This results in the first page number remaining as is or being increased by one. Default: `#f`.

`binding-offset` (number)
: This amount gets added to `inner-margin` so that nothing is hidden by the binding. Has only an effect if `two-sided` is set to `#t`. Default: `binding-offset-default` (5 mm, scaled to paper size).

`blank-after-score-page-penalty` (number)
: The penalty for having a blank page after the end of one score and before the next. By default, this is smaller than `blank-page-penalty` so that LilyPond prefers blank pages after scores to blank pages within a score. Default: 2.

`blank-last-page-penalty` (number)
: The penalty for ending the score on an odd-numbered page. Default: 0.

`blank-page-penalty` (number)
: The penalty for having a blank page in the middle of a score. Note that this is not used by `ly:optimal-breaking`, which never considers blank pages in the middle of a score. Default: 5.

`blot-diameter` (non-negative number)
: This value globally defines the smallest 'round' feature LilyPond uses while constructing almost all non-glyph elements like beams or stems. Essentially, it sets up how round the corners and line ends are. It only makes sense to change the value if you use a different music glyph font with crisper corners, say. Default: 0.4 pt.

`book-title` (procedure)
: Internal. This is what LilyPond actually uses for handling `bookTitleMarkup`.

`book-title-properties` (association list (list of pairs))
: Internal.

`bookpart-level-page-numbering` (boolean)
: If set to `#t`, restart page numbering for each `\bookpart` block. Otherwise all pages in the document are enumerated continuously. Default: `#f`.

`bookTitleMarkup` (markup)
: The titling markup within a `\book`, using standard fields from the `\header` block. The default value is defined in file `titling-init.ly`.

`bottom-margin` (non-negative number)
: The margin between the bottom of the printable area and the bottom of the page. Default: `bottom-margin-default` (10 mm, scaled to paper size).

`bp` (positive number)
: The big point unit, also called *desktop publishing point* (DTP point). It is 1/72 of an inch, approx. 0.353 mm (0.0138 in).

`check-consistency` (boolean)
: If set to `#t`, print a warning if the left margin, line width, and right margin do not exactly add up to `paper-width`, and replace each of these (except `paper-width`) with its default value (scaled to the paper size if necessary). If set to `#f`, ignore any inconsistencies and allow systems to run off the edge of the page. Default: `#t`.

`clip-regions` (association list (list of pairs))
: A list of rhythm location pairs to output fragments of a score.

```
clip-regions
= #(list (cons (make-rhythmic-location 2 0 1)
               (make-rhythmic-location 4 0 1))
         (cons (make-rhythmic-location 5 1 2)
               (make-rhythmic-location 7 3 4)))
```

  The above example defines two regions: the first is from the beginning of bar 2 to the beginning of bar 4, the second starts after a half note duration in bar 5 and ends after the third quarter in bar 7.

  Has only an effect if LilyPond's command-line option `-dclip-systems` is active. Default: unset.

`cm` (positive number)
: The centimeter unit, approx. 0.39 in.

`debug-beam-scoring` (boolean)
: If set to `#t`, print demerits together with their cause, followed by the number of configurations that have been scored before concluding. Default: `#f`.

  Example:

  > `L 18.95 C 655.12 c19/625` → demerits for stem lengths (‘`L`’) and collisions (‘`C`’), scored 19 out of 625 initially considered configurations.

  Possible demerit causes: collision (‘`C`’), inappropriate stem length (‘`L`’), beam direction different from damping direction (‘`Sd`’), difference between beam slope and musical slope (‘`Sm`’), deviation from ideal slope (‘`Si`’), horizontal inter-quants (‘`H`’), forbidden quants (‘`Fl`’/‘`Fs`’).

  Demerits are configurable, see Section “beam-interface” dans *Référence des propriétés internes* for a list of tunable parameters.

`debug-slur-scoring` (boolean)
: If set to `#t`, print demerits together with their cause, followed by the sum of all demerits and the index of the slur configuration finally chosen. Default: `#f`.

  Example:

  ```
  slope=2.00, R edge=10.51, variance=0.03 TOTAL=12.54 idx=4        →
  ```

  demerits for slope, distance of the right edge to the attachment point, variance of distance between note heads and slur. Total demerits: 12.54, index of the chosen configuration: 4.

  Possible demerit causes: distance of the left/right slur edge to the attachment points (‘`L edge`’/‘`R edge`’), inappropriate slope (‘`slope`’), distance variations between note heads and slur (‘`variance`’), distances for heads that are between the slur and an imaginary line between the attachment points (‘`encompass`’), too small distance between slur and tie extrema (‘`extra`’).

  Demerits are configurable, see Section “slur-interface” dans *Référence des propriétés internes* for a list of tunable parameters.

`debug-tie-scoring` (boolean)
: If set to `#t`, print the basic configuration of ties, followed by demerits and their corresponding causes and the total sum of demerits. Default: unset.

  Example:

  ```
  0 (0.23) u: vdist=1.08 lhdist=1.79 tie/stem dir=8.00 TOTAL=10.87 →
  ```

  offset from the center of the staff according tie specification: 0 staff-spaces, vertical distance of the tie’s center in y-direction to the bottom (or top) of the tie: 0.23, direction: up. Demerits for vertical and horizontal distance to note head, same direction of stem and tie. Total demerits: 10.87.

  Possible demerit causes: wrong tie direction (‘`wrong dir`’), vertical distance to note heads (‘`vdist`’), horizontal distance to left or right note head (‘`lhdist`’/‘`rhdist`’), same direction of stem and tie (‘`tie/stem dir`’), position and direction of tie not matching, e.g., tie is in the upper half of the staff but has direction `DOWN` (‘`tie/pos dir`’), tie is too short (‘`minlength`’), tip of tie collides with staff line (‘`tipline`’), collision with dot (‘`dot collision`’), center of tie is too close to a staff line (‘`line center`’), y-position (edge or center) of currently considered tie is less than the y-position of the previous tie (‘`monotone edge`’/ ‘`monotone cent`’), edge or center of tie is too close to the one considered previously (‘`tietie center`’/‘`tietie edge`’), unsymmetrical horizontal positioning with respect to the note heads (‘`length symm`’), unsymmetrical vertical positioning with respect to the note heads (‘`pos symmetry`’).

  Demerits are configurable, see Section “tie-interface” dans *Référence des propriétés internes* for a list of tunable parameters.

`dimension-variables` (list)
: Internal.

`evenFooterMarkup` (markup)
: The footer markup used for even-numbered pages. If not set, `oddFooterMarkup` is used instead. The default value is defined in file `titling-init.ly`.

`evenHeaderMarkup` (markup)
: The header markup used for even-numbered pages. If not set, `oddHeaderMarkup` is used instead. The default value is defined in file `titling-init.ly`.

`first-page-number` (integer)
: The value of the page number on the first page. Default: 1.

`footnote-footer-padding` (number)
: The padding between the footer and the bottom-most footnote. Default: 0.5 mm.

`footnote-number-raise` (number)
: This controls how high the annotation numbers of both footnotes and in-notes are raised relative to the footnote or in-note text. Default: 0.5 mm.

`footnote-numbering-function` (procedure)
: This variable holds the name of the function that formats both footnote and in-note numbers (without positioning it). LilyPond provides two predefined functions (in file `output-lib.scm`): `numbered-footnotes` ('1', '2', etc.) and `symbol-footnotes` ('*', '†', etc.). Default: `numbered-footnotes`.

  The function takes an integer as an argument (starting with value 0) and returns the appropriate markup. You can create your own function, for example,

```
footnote-numbering-function =
#(lambda (x)
  #{ \markup \concat { "[" #(number->string (1+ x)) "]" } #})
```

`footnote-padding` (number)
: The padding between two footnotes. Default: 0.5 mm.

`footnote-separator-markup` (markup)
: Markup to separate the music from the footnotes, usually a horizontal line. The default value is defined in file `paper-defaults-init.ly`.

`horizontal-shift` (number)
: If set, the main content block of all pages (but not headers and footers) is shifted horizontally. Positive values shift to the right (on both even and odd pages). Default: 0 mm.

`in` (positive number)
: The inch unit, equal to 2.54 cm.

`in-note-padding` (number)
: The padding between two in-notes. Default: 0.5 mm.

`in-note-system-padding` (number)
: The padding between an in-note and the associated music system. Default: 0.5 mm.

`incipit-width` (positive number)
: The width of an incipit as created by the `\incipit` command. Must be smaller than or equal to `indent`. Default: `(indent / 2)`.

`indent` (non-negative number)
: The indentation of the first system in a score. The space within the `line-width` available for the first system is reduced by this amount. Default: `indent-default` (15 mm, scaled to paper size).

`inner-margin` (non-negative number)
: The margin all pages have at the inner side if they are part of a book. Has only an effect if `two-sided` is set to `#t`. If unset, `inner-margin-default` (15 mm, scaled to paper size) is used in computations. Default: unset.

`is-last-bookpart` (boolean)
: To be documented.

`label-alist-table` (list)
: Internal. LilyPond uses this to construct a table of contents.

`label-page-table` (association list (list of pairs))
: Internal. LilyPond uses this to implement the `\with-link` functionality and to construct a table of contents.

`landscape` (boolean)
: Internal. Use `set-default-paper-size` or `set-paper-size` to set the paper orientation.

`last-bottom-spacing` (association list (list of pairs))
: The distance from the last system or top-level markup on a page to the bottom of the printable area (i.e., the top of the bottom margin). The default value is defined in file `paper-defaults-init.ly`.

`left-margin` (non-negative number)
: The margin between the left edge of the page and the start of the staff lines in unindented systems. If `left-margin` is not set, and both `line-width` and `right-margin` are set, then `left-margin` is set to `(paper-width - line-width - right-margin)`. If only `line-width` is set, then both margins are set to `((paper-width - line-width) / 2)`, and the systems are consequently centered on the page. If unset, `left-margin-default` (15 mm, scaled to paper size) is used in computations. Default: unset.

  If `two-sided` is set to `#t`, this value is ignored. Also see `check-consistency`.

`line-thickness` (positive number)
: This value globally defines the default line thickness LilyPond uses while constructing many non-glyph elements like lines or boxes. The actual line thickness is usually specified as a factor of this base value, either as a default value, or overridden by the user. The default depends on the staff space and is computed as `(0.328571 + 0.0342857 * staff-space)` (see function `calc-line-thickness` in file `paper.scm`), which is synchronized with the Emmentaler font.

`line-width` (positive number)
: The horizontal extent of the staff lines in unindented, non-ragged systems, equal to `(paper-width - left-margin - right-margin)` if not set. If `line-width` is set, and both `left-margin` and `right-margin` are not set, then the margins are updated to center the systems on the page automatically. Also see `check-consistency`. Default: unset.

  As a feature of LilyPond, if the command-line option `-dcrop` is set, the dimensions of the cropped output file are always rounded up to integer (PostScript) big points; this might cause a little bit of whitespace at the right margin. To avoid that, set `line-width` to an integer big point value like `450\bp`.

`make-footer` (procedure)
: Internal. This is what LilyPond actually uses for for handling `evenFooterMarkup` and `oddFooterMarkup`.

`make-header` (procedure)
: Internal. This is what LilyPond actually uses for for handling `evenHeaderMarkup` and `oddHeaderMarkup`.

`markup-markup-spacing` (association list (list of pairs))
: The distance between two (title or top-level) markups. The default value is defined in file `paper-defaults-init.ly`.

`markup-system-spacing` (association list (list of pairs))
: The distance between a (title or top-level) markup and the system that follows it. The default value is defined in file `paper-defaults-init.ly`.

`max-systems-per-page` (non-negative, exact integer)
: The maximum number of systems that are placed on a page. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`min-systems-per-page` (non-negative, exact integer)
: The minimum number of systems that are placed on a page. This may cause pages to be overfilled if it is made too large. This is currently supported only by the `ly:optimal-breaking` algorithm. Default: unset.

`mm` (positive number)
: The millimeter unit, approx. 0.039 in.

`number-footnote-table` (list)
: Internal. LilyPond uses this to manage footnotes and footnote numbers.

`oddFooterMarkup` (markup)
: The footer markup used for odd-numbered pages. The default value is defined in file `titling-init.ly`.

`oddHeaderMarkup` (markup)
: The header markup used for odd-numbered pages. The default value is defined in file `titling-init.ly`.

`orphan-penalty` (number)
: To be documented. See `lily/page-breaking.cc`. Default is 100000.

`outer-margin` (non-negative number)
: The margin all pages have at the outer side if they are part of a book. Has only an effect if `two-sided` is set to `#t`. If unset, `outer-margin-default` (15 mm, scaled to paper size) is used in computations. Default: unset.

`output-filename` (string)
: If set, use this value as the output file name for LilyPond output. See also `output-suffix`. The default is the input file name (without the file extension).

`output-scale` (positive number)
: Internal. This value globally defines the output scale LilyPond uses while creating output. Use `set-global-staff-size` or `layout-set-staff-size` to change the scaling. Default: 1.7573, which corresponds to a 20 pt staff size.

`output-suffix` (string)
: If set, use this value as the output file name suffix for LilyPond output, appended to the output name with a dash in between. See also `output-filename`. The default is no output suffix if there is only a single book block (either implicit or explicit), and a running integer starting with value 1 otherwise.

`page-breaking` (procedure)
: The page-breaking algorithm to use. Choices are `ly:minimal-breaking`, `ly:page-turn-breaking`, `ly:one-page-breaking`, `ly:one-line-breaking`, `ly:one-line-auto-height-breaking`, and `ly:optimal-breaking`. Default: `ly:optimal-breaking`.

`page-breaking-system-system-spacing` (association list (list of pairs))
: This tricks the page breaker into thinking that `system-system-spacing` is set to something different than it really is. For example, if `page-breaking-system-system-spacing.padding` is set to something substantially larger than `system-system-spacing.padding`, then the page breaker puts fewer systems on each page. Default: unset.

page-count (non-negative, exact integer)
: The number of pages to be used for a score. Default: unset.

page-number-type (symbol)
: The type of numerals used for page numbers. Choices include arabic, roman-ij-lower, roman-ij-upper, roman-lower, and roman-upper. Default: arabic.

page-post-process (procedure)
: If this function is defined (within the \paper block), LilyPond uses it to post-process pages, for example, to extract the table of contents and writing the information to an auxiliary file. The call happens after page breaking has been performed.

  The syntax is

  ```
  #(define (page-post-process layout pages) ... )
  ```

page-spacing-weight (number)
: When using the ly:optimal-breaking algorithm for page breaking, LilyPond has to make trade-offs between horizontal and vertical stretching so that the overall spacing is more acceptable. This parameter controls the relative importance of (vertical) page spacing and (horizontal) line spacing. High values makes page spacing more important. Default: 10.

paper-height (positive number)
: The height of the page. Note that the automatic scaling of some vertical dimensions is not affected if you set this value directly. Default: 297 mm (A4 paper height).

paper-width (positive number)
: The width of the page. While setting paper-width directly has no effect on the automatic scaling of some horizontal dimensions, it does influence the line-width variable. If both paper-width and line-width are set, then left-margin and right-margin also get updated. Also see check-consistency. Default: 210 mm (A4 paper width).

papersizename (string)
: Internal. Use set-default-paper-size or set-paper-size to set the paper size.

print-all-headers (boolean)
: If set to #t, print all headers for each \score in the output. If set to #f, only the piece and opus header variables are printed. Default: #f.

print-first-page-number (boolean)
: If set to #t, a page number is printed on the first page. Default: #f.

print-page-number (boolean)
: If set to #f, page numbers are not printed. Default: #t.

property-defaults (association list (list of pairs))
: This variable holds some settings that are used for top-level markups and as fallback values if nothing else is specified, for example, the standard fonts or the baseline skip. The default value is defined in file paper-defaults-init.ly.

pt (positive number)
: The point unit, equal to approx. 0.351 mm (0.0139 in).

ragged-bottom (boolean)
: If set to #t, systems are set at at their natural spacing, neither compressed nor stretched vertically to fit the page. Default: #f.

ragged-last (boolean)
: If set to #t, the last system in the score does not fill the line width. Instead, the last system ends at its natural horizontal length. Default: #f.

ragged-last-bottom (boolean)
: If set to #f, then the last page, and the last page in each section created with a \bookpart block, is vertically justified in the same way as the earlier pages. Default: #t.

ragged-right (boolean)
: If set to #t, systems don't fill the line width. Instead, systems end at their natural horizontal length. Default: #t for scores with only one system, and #f for scores with two or more systems.

reset-footnotes-on-new-page (boolean)
: If set to #t, footnote and in-note numbers are reset on each page break. For footnotes and in-notes numbered consecutively across page breaks, set to #f. Default: #t.

right-margin (non-negative number)
: The margin between the right edge of the page and the end of the staff lines in non-ragged systems. If right-margin is not set, and both line-width and left-margin are set, then right-margin is set to (paper-width - line-width - left-margin). If only line-width is set, then both margins are set to ((paper-width - line-width) / 2), and the systems are consequently centered on the page. If unset, right-margin-default (15 mm, scaled to paper size) is used in computations. Default: unset.

  If two-sided is set to #t, this value is ignored. Also see check-consistency.

score-markup-spacing (association list (list of pairs))
: The distance between the last system of a score and the (title or top-level) markup that follows it. The default value is defined in file paper-defaults-init.ly.

score-system-spacing (association list (list of pairs))
: The distance between the last system of a score and the first system of the score that follows it, if no (title or top-level) markup exists between them. The default value is defined in file paper-defaults-init.ly.

score-title (procedure)
: Internal. This is what LilyPond actually uses for handling scoreTitleMarkup.

score-title-properties (association list (list of pairs))
: Internal.

scoreTitleMarkup (markup)
: The titling markup of a score within a \book, using standard fields from the \header block. The default value is defined in file titling-init.ly.

short-indent (non-negative number)
: The level of indentation for all systems in a score besides the first system. The space within the line-width available for systems other than the first one is reduced by this amount. Default: short-indent-default (0 mm).

staff-height (positive number)
: Internal. Use set-global-staff-size or layout-set-staff-size to set the staff height. Default: 20 pt.

staff-space (positive number)
: Internal. One staff height contains four staff spaces. Use set-global-staff-size or layout-set-staff-size to set this value. Default: 5 pt.

system-count (non-negative, exact integer)
: The number of systems to be used for a score. Default: unset.

system-separator-markup (markup)
: A markup object that is inserted between systems, often used for orchestral scores. The \slashSeparator markup, defined in file titling-init.ly, is provided as a sensible standard value. Default: unset.

system-system-spacing (association list (list of pairs))
: The distance between two systems in the same score. The default value is defined in file paper-defaults-init.ly.

systems-per-page (non-negative, exact integer)
: The number of systems that should be placed on a page. This is currently supported only by the ly:optimal-breaking algorithm. Default: unset.

tagline (markup)
: This markup gets placed at the bottom of the last page (via oddFooterMarkup). The default value is defined in file titling-init.ly.

text-font-size (positive number)
: This value globally defines the standard text size for markups. The default depends on the staff height and is computed as (staff-height / 20 * 11).

tocFormatMarkup (procedure)
: How the top-level entries of the table of contents are formatted (if there are several hierarchical levels). This variable holds a function like make-bold-markup. The default value is defined in file toc-init.ly.

tocIndentMarkup (markup)
: How the outline's hierarchy in the table of contents is made apparent. This markup is printed zero, one, or several times depending on the level of each entry. The default value is defined in file toc-init.ly.

tocItemMarkup (markup)
: How an item in the table of contents is formatted. The default value of this markup is defined in file toc-init.ly.

tocTitleMarkup (markup)
: How the title of the table of contents is formatted. The default value of this markup is defined in file toc-init.ly.

top-margin (non-negative number)
: The margin between the top of the page and the top of the printable area. Default: top-margin-default (10 mm, scaled to paper size).

top-markup-spacing (association list (list of pairs))
: The distance from the top of the printable area (i.e., the bottom of the top margin) to the first (title or top-level) markup on a page, when there is no system between the two. The default value is defined in file paper-defaults-init.ly.

top-system-spacing (association list (list of pairs))
: The distance from the top of the printable area (i.e., the bottom of the top margin) to the first system on a page, when there is no (title or top-level) markup between the two. The default value is defined in file paper-defaults-init.ly.

two-sided (boolean)
: If set to #t, use inner-margin, outer-margin and binding-offset to determine margins depending on whether the page number is odd or even. left-margin and right-margin are then ignored. Default: #f.

B.24 Conventions de nommage

Afin de s'y retrouver plus aisément et d'éviter les erreurs de frappe, voici quelques conventions en matière de nommage.

Objet ou type de propriété	**Convention de nommage**	**Exemples**
Mot-clé	`aaaa`	`\score`, `\addlyrics`
Fonctions musicales et raccourcis	`aaaa` or `aaaaAaaaAaaa`	`\time`, `\stemUp`
Fonctions pour *markups*	`aaaa` or `aaaa-aaaa-aaaa`	`\bold`, `\with-dimensions`
Variables du bloc \paper	`aaaa` or `aaaa-aaaa-aaaa`	`indent`, `page-breaking`
Expressions musicales	`Aaaa` or `AaaaAaaaAaaa`	`NoteEvent`, `SequentialMusic`
Classes musicales	`aaaa` or `aaaa-aaaa-aaaa`	`note-event`, `rhythmic-event`
Propriétés musicales	`aaaa` or `aaaa-aaaa-aaaa`	`force-accidental`, `iterator-ctor`
Contextes	`Aaaa` or `AaaaAaaaAaaa`	`Staff`, `GrandStaff`
Propriétés de contexte	`aaaa` or `aaaaAaaaAaaa`	`stanza`, `instrumentName`
Graveurs	`Aaaa_engraver` or `Aaaa_aaaa_aaaa_engraver`	`Clef_engraver`, `Note_heads_engraver`
Objets de rendu	`Aaaa` or `AaaaAaaaAaaa`	`Slur`, `NoteHead`
Propriétés d'objet de rendu	`aaa` or `aaa-aaa-aaa`	`direction`, `beam-thickness`
Interfaces	`aaa-aaa-interface`	`grob-interface`, `break-aligned-interface`
Fonctions Scheme définies par Guile	`aaaa` or `aaaa-aaaa-aaaa`	`map`, `for-each`
Fonctions Scheme définies dans LilyPond	`ly:aaaa` or `ly:aaaa-aaaa-aaaa` ou `aaaa` ou `aaaa-aaaa-aaaa`[1]	`ly:round-filled-box`, `music-map`

B.25 Types de prédicats prédéfinis

Predicates return `#t` (true) if their argument is of the named type and `#f` (false) if it isn't.

B.25.1 R5RS primary predicates

Primary predicates can be applied to any expression. They can be used on their own as predicates for LilyPond functions. The predicates here are part of the Scheme standard R5RS.

Type predicate	**Description**
`boolean?`	boolean
`char?`	character
`complex?`	complex number
`eof-object?`	end-of-file object
`input-port?`	input port
`integer?`	integer
`list?`	list (use `cheap-list?` for faster processing)
`null?`	null
`number?`	number
`output-port?`	output port

[1] Pour des raisons historiques, toutes les fonctions définies dans LilyPond ne sont pas forcément préfixées d'un `ly:`.

pair?	pair
port?	port
procedure?	procedure
rational?	rational number
real?	real number
string?	string
symbol?	symbol
vector?	vector

B.25.2 R5RS secondary predicates

Secondary predicates are only applicable to specific expressions (for example, to numbers). They will throw a type error when applied to expressions they are not intended for. The predicates here are part of the Scheme standard R5RS.

Type predicate	**Description**
char-alphabetic?	alphabetic character
char-lower-case?	lower-case character
char-numeric?	numeric character
char-upper-case?	upper-case character
char-whitespace?	whitespace character
even?	even number
exact?	exact number
inexact?	inexact number
negative?	negative number
odd?	odd number
positive?	positive number
zero?	zero

B.25.3 Guile predicates

These predicates are defined by Guile but are not part of a Scheme standard.

Type predicate	**Description**
hash-table?	hash table

B.25.4 LilyPond scheme predicates

These predicates are only available within LilyPond and defined in Scheme.

Type predicate	**Description**
alist?	association list (list of pairs)
boolean-or-symbol?	boolean or symbol
cheap-list?	list (use this instead of list? for faster processing)
color?	color
exact-rational?	an exact rational number
fraction?	fraction, as pair
grob-list?	list of grobs
grouped-number-list?	list of non-empty lists of numbers
index?	non-negative, exact integer
index-or-markup?	index or markup
key?	index or symbol
key-list?	list of indices or symbols
key-list-or-music?	key list or music
key-list-or-symbol?	key list or symbol
ly:skyline-pair?	pair of skylines

markup?	markup
markup-command-list?	markup command list
markup-list?	markup list
moment-pair?	pair of moment objects
musical-length?	non-negative exact rational, fraction (as pair), moment, or +inf.0
musical-length-as-moment?	non-negative moment with no grace part
musical-length-as-number?	non-negative exact rational or +inf.0
non-negative-number?	non-negative number
number-list?	number list
number-or-grob?	number or grob
number-or-number-pair?	number or pair of numbers
number-or-pair?	number or pair
number-or-string?	number or string
number-pair?	pair of numbers
number-pair-list?	list of number pairs
optionally-grouped-beat-structure?	a number list or a list of them
positive-exact-rational?	a positive, exact rational number
positive-fraction?	positive, finite fraction, as pair
positive-musical-length?	positive exact rational, fraction (as pair), moment, or +inf.0
positive-musical-length-as-moment?	positive moment with no grace part
positive-musical-length-as-number?	positive exact rational or +inf.0
positive-number?	positive number
rational-or-procedure?	an exact rational or procedure
rhythmic-location?	rhythmic location
sane-simple-time-signature?	simple time signature
sane-time-signature?	time signature
scale?	non-negative rational, fraction, or moment
scheme?	any type
string-or-music?	string or music
string-or-pair?	string or pair
string-or-symbol?	string or symbol
symbol-key-alist?	alist, with symbols as keys
symbol-list?	symbol list
symbol-list-or-music?	symbol list or music
symbol-list-or-symbol?	symbol list or symbol
time-signature?	time signature
void?	void
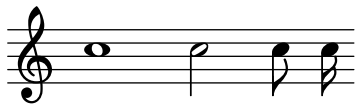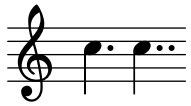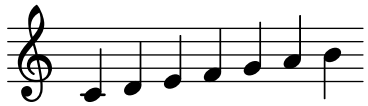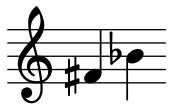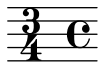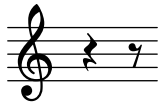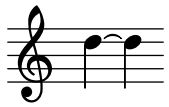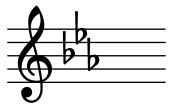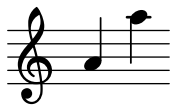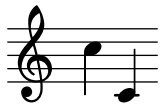
B.25.5 LilyPond exported predicates
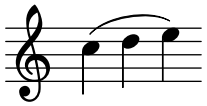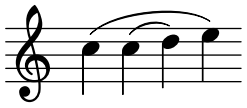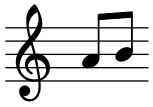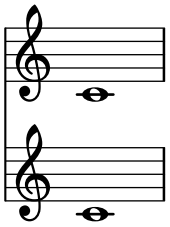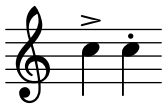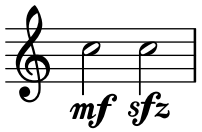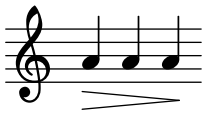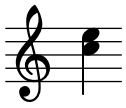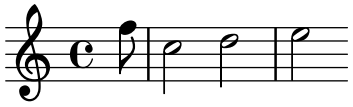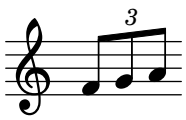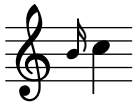
These predicates are only available within LilyPond and usually defined in C++.

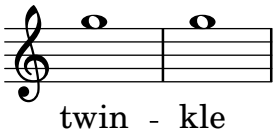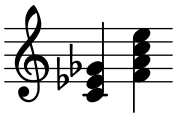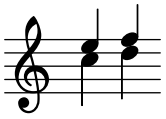
Type predicate	**Description**
ly:book?	book
ly:book-or-bookpart?	book or bookpart
ly:bookpart?	bookpart
ly:context?	context
ly:context-def?	context definition
ly:context-mod?	context modification
ly:dimension?	dimension, in staff space
ly:dir?	direction

`ly:dispatcher?`	dispatcher
`ly:duration?`	duration
`ly:event?`	post-event
`ly:font-metric?`	font metric
`ly:grob?`	graphical (layout) object
`ly:grob-array?`	array of grobs
`ly:grob-properties?`	grob properties
`ly:input-location?`	input location
`ly:item?`	item
`ly:iterator?`	iterator
`ly:lily-lexer?`	lily-lexer
`ly:lily-parser?`	lily-parser
`ly:listener?`	listener
`ly:moment?`	moment
`ly:music?`	music
`ly:music-function?`	music function
`ly:music-list?`	list of music objects
`ly:music-output?`	music output
`ly:note-scale?`	note scale
`ly:otf-font?`	OpenType font
`ly:output-def?`	output definition
`ly:page-marker?`	page marker
`ly:pango-font?`	Pango font
`ly:paper-book?`	paper book
`ly:paper-system?`	paper-system Prob
`ly:pitch?`	pitch
`ly:prob?`	property object
`ly:score?`	score
`ly:skyline?`	skyline
`ly:source-file?`	source file
`ly:spanner?`	spanner
`ly:spring?`	spring
`ly:stencil?`	stencil
`ly:stream-event?`	stream event
`ly:transform?`	coordinate transform
`ly:translator?`	translator
`ly:translator-group?`	translator group
`ly:unpure-pure-container?`	unpure/pure container

Annexe C Aide-mémoire

Syntaxe	Description	Exemple
`1 2 8 16`	valeurs rythmiques	
`c4. c4..`	notes pointées	
`c d e f g a b`	gamme	
`fis bes`	altérations	
`\clef treble \clef bass`	clés	
`\time 3/4 \time 4/4`	chiffre de mesure, métrique	
`r4 r8`	silences	
`d ~ d`	liaison de tenue	
`\key es \major`	armure	
note`'`	monter d'une octave	
note`,`	baisser d'une octave	

`c( d e)`	liaisons	
`c\( c( d) e\)`	liaisons de phrasé	
`a8[ b]`	ligatures	
`<< \new Staff ... >>`	ajouter des portées	
`c-> c-.`	indications d'articulation	
`c2\mf c\sfz`	nuances	
`a\< a a\!`	crescendo	
`a\> a a\!`	decrescendo	
`< >`	accords	
`\partial 8`	levées, anacrouses	
`\tuplet 3/2 {f g a}`	triolets	
`\grace`	appoggiatures	

`\lyricmode { twinkle }`	ajouter des paroles	twinkle
`\new Lyrics`	imprimer les paroles	twinkle
`twin -- kle`	diviser un mot en plusieurs syllabes	
`\chordmode { c:dim f:maj7 }`	accords chiffrés	
`\new ChordNames`	imprimer les chiffrages d'accords	C° F△
`<<{e f} \\ {c d}>>`	polyphonie	
`s4 s8 s16`	silences invisibles	

Annexe D GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
https://fsf.org/

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Annexe E Index des commandes et concepts

En plus des commandes et mots réservés de LilyPond, cet index recense les termes musicaux qui s'y rapportent.

« Amazing Grace » : exemple pour cornemuse 513
« accent », articulation 158, 159
« bachschleifer », articulation 158
« coda », signe 158
« espressivo », articulation 158
« flageolet », harmoniques 158
« grupetto barré », ornement 158
« grupetto Haydn », ornement 158
« grupetto renversé », ornement 158
« grupetto », ornement 158
« hyphénation », trait d'union 367
« marcato », articulation 158, 159
« mordant ascendant », ornement 158
« mordant descendant », ornement 158
« mordant », ornement 158
« No Chord », personnalisation du symbole 530
« No Chord », symbole 528
« ouvert », articulation 158
« pointe », pédalier 158, 436
« portato », articulation 158, 159
« pouce », doigté 158
« poussé », archet 158
« poussé », indication d'archet 441
« segno », signe 158
« semi-ouvert », articulation 158
« signum congruentiae », ornement 158
« snappizzicato », signe 158
« staccatissimo », articulation 158, 159
« staccato », articulation 158, 159
« stop », articulation 158
« talon », pédalier 158, 436
« tenuto », articulation 158, 159
« tiré », archet 158
« tiré », indication d'archet 441
« trille », ornement 158
« variante coda », signe 158

!

! 9
\! 163

"

"|" 144

#

#f (false) 989
#t (true) 989

%

% 609, 614
\% 213, 979
%{ ... %} 609, 614

'

' 3

(

(..... 174
\(..... 177

)

) 174
\) 177

*

* 62
* 193, 979

,

, ... 3

-

- 158, 801
\- 549
-! 944
-+ 945
-- 367, 944
-. 944
-> 944
-^ 944
-_ 944

.

. 54

/

/ 62, 526
/+ 526

:

: 217, 523

<

< 221
\< 163
<...> 221
<< 225
<< ... \\ ... >> 230
<< ... >> 225
<> 164, 222, 428

=

= 13
\= 174, 978

>

> 221
\> 163
>> 225

?

? 9

[

[..... 117
\[..... 554

]

] 117
\] 554

^

^ 526, 801
\^ 453

_

_ 363, 801
__ 367

\

\\ 230

|

| 144

~

~ 64, 363
\~ 573, 579, 979

1

15ma 27
15mb 27

8

8va 27
8vb 27

A

a due 238
à l'italienne, papier 693
à la française, partition 265
\abs-fontsize 334, 837
\abs-hspace 850
\abs-vspace 850
absolue, hauteur 3
absolue, octave 3
\absolute 959
\accent 158, 944
\accentus 572, 945
accentus, grégorien, articulation 572
\accepts 776, 777, 779
\acciaccatura 151, 959
acciaccature 151
acciaccature, multinotes 155
\accidental 879
\accidentalStyle 32, 959
AccidentalSuggestion 161
accidentel, quart de ton 10
accidentelle automatique 32
accidentelle, altération 32
accolade verticale 250
accord 221
accord arpégé 186
accord chiffré, exception 533
accord en bloc 187
accord enroulé 187
accord et altération 39
accord et couleur 300
accord et doigtés 291
accord et liaisons de tenue 64
accord et mode relatif 222
accord et octave relative 7
accord et reprise 535
accord jazz 531
accord nommé et diagramme de fret 480
accord plaqué 187
accord répété, suppression 487
accord vide 153, 164, 222, 428
accord, additions 525
accord, altération d'un degré 525
accord, carrure pour cordes frettées 483
accord, chiffrage 522, 527
accord, crochet 187
accord, crochet interportée 434
accord, diagramme 469, 479
accord, diagramme automatique 490
accord, inversion 526
accord, liaison 187
accord, mode 522
accord, modificateur 523
accord, modification de l'une des notes 791
accord, No Chord 528
accord, nom 522
accord, nom alternatif 531
accord, personnalisation du chiffrage 531
accord, répétition 223, 448
accord, répartition sur plusieurs portées avec \autoChange 431
accord, renversement 527
accord, séparateur 535
accord, spécifier la basse 526

accord, suppressiion des répétitions ... 487
accord, suppression d'un degré ... 526
accord, suppression des répétitions ... 529
accord, vocification ... 527
accordéon, symbole de registre ... 437
accordéon, tirettes ... 437
accordage de banjo ... 498
accordage non-occidental ... 591
accordage personnalisé ... 466
accordages prédéfinis et cordes frettées ... 465
accords ... 527
accords, grille ... 536
acoustic bass ... 946
acoustic snare ... 946
add-grace-property ... 153
add-stem-support ... 292
add-toc-item! ... 642
\addChordShape ... 483, 959
adding white background, to text ... 912
\addInstrumentDefinition ... 959
additionalPitchPrefix ... 532
additions à un accord ... 525
\addlyrics ... 357, 359, 360
\addQuote ... 274, 959
\addToTagGroup ... 653, 959
\aeolian ... 25
affinage (*tweak*) ... 791
\after ... 164, 214, 959
\afterGrace ... 151, 959
afterGraceFraction ... 957
agogo ... 946
Aiken, note profilée, variante fine ... 50
Aiken, tête de note ... 48
\aikenHeads ... 48
\aikenHeadsMinor ... 49
\aikenThinHeads ... 48
\aikenThinHeadsMinor ... 49
ajout d'incipit ... 583
ajout de texte ... 320
ajustement (*tweak*) ... 791
ajustement des extrémités, liaison ... 175
al coda ... 201
al fine ... 201
al niente, soufflet ... 168
\alias ... 776
\align-on-other ... 850
aligné, *ragged* ... 691
alignAboveContext ... 264, 370, 781
alignBelowContext ... 264, 370, 380, 781
alignement des *markups* ... 337
alignement des objets ... 820
alignement du numéro de mesure ... 141
alignement du texte ... 337
alignement du texte, commandes ... 342
alignement et cadence ... 156
alignement horizontal du texte ... 337
alignement horizontal, mélisme ... 375
alignement horizontal, paroles ... 375
alignement sur un objet ... 826
alignement vertical du texte ... 338
alignement vertical, nuance ... 169
alignement vertical, rupture pour les nuances ... 170
alignement, point d' ... 820
alignment vertical, script textuel ... 169
alist ... 955
alla breve ... 82, 83
alla coda ... 201
\allowBreak ... 123, 711
\allowPageTurn ... 718, 959
\allowVoltaHook ... 960
alpha, transparence ... 298
altérable, objet et propriété ... 956
altération ... 8, 561, 581
altération de précaution ... 9
altération de précaution, style *modern* ... 35
altération de précaution, style *modern voice* ... 35
altération entre parenthèses ... 9
altération et accord ... 39
altération et cadence ... 94
altération et liaison de tenue ... 9
altération et musica ficta ... 561
altération et notes simultanées ... 39
altération masquée sur note tenue au début du système suivant ... 9
altération, basse chiffrée, position ... 544
altération, glyphes alternatifs ... 40
altération, grégorien ... 571
altération, style ... 32
altération, style *default* ... 34
altération, style *modern* ... 34
altération, style *modern cautionary* ... 35
altération, style par défaut ... 32
altération, trille avec hauteur explicite ... 191
altérations multivoix ... 35, 37
altérations perses (sori, koron) ... 599
altérations, jeux de glyphes ... 938
altérations, style *choral* ... 36
altérations, style *choral cautionary* ... 36
altérations, style *dodecaphonic* ... 37
altérations, style *dodecaphonic* néo-moderne ... 38
altérations, style *forget* ... 38
altérations, style *modern* ... 34
altérations, style *modern-cautionary* ... 34
altérations, style *neo-modern* ... 36
altérations, style *neo-modern-cautionary* ... 37
altérations, style *neo-modern-voice* ... 37
altérations, style *no reset* ... 38
altérations, style *piano cautionary* ... 36
altérations, style *teaching* ... 38
altérations, style *voice* ... 34
altérationsn style *piano* ... 35
alteration, figured bass, glyphs ... 544
\alterBroken ... 804, 960
alternatative, table de diagrammes d'accord ... 488
alternatif, glyphe d'altération ... 40
\alternative ... 195, 197
alternative et liaison ... 65
alternative et paroles ... 381
alternative et texte ... 211
alternative, bascule sur une mélodie ... 395
alternative, reprise ... 195
alto, clef ... 939
alto, varC, clef ... 939
ambitus ... 41
ambitus avec ligne ... 43
ambitus, placement ... 44
ambitus, un par voix ... 42
ambitus, voix multiples ... 42
ambitusAfter ... 44, 960
AmbitusLine ... 43

amplitude ... 41
anacrouse ... 93
anacrouse et reprise ... 195
anacrouse, cantique ... 422
analyse lexicale ... 956
analyse musicologique ... 313
analyse, crochet, étiquette ... 314
analyseur syntaxique ... 957
ancien silence ... 560
ancien, crochet ... 559
ancien, script ... 945
ancienne altération ... 561
ancienne ligature ... 563
ancienne tête de note ... 558
ancienne, clef ... 20, 557
ancienne, métrique ... 558
annotate moving by spacing, in text ... 902
\annotate-moving ... 902
annotate-spacing ... 752, 980
annotation ... 329
annulation d'un *override* ... 789
annulation de dérogation ... 789
\append-to-tag ... 655, 903
\appendToTag ... 654, 960
\appendToTagMarkup ... 656, 960
\applyContext ... 766, 960
\applyMusic ... 960
\applyOutput ... 960
\applySwing ... 684
\applySwingWithOffset ... 684
\appoggiatura ... 151, 960
appoggiature ... 151
\approximatePitch ... 10, 960
approximative, hauteur ... 10
arabe, armure ... 593
arabe, exemple de musique ... 596
arabe, improvisation ... 596
arabe, métrique ... 595
arabe, maqam ... 591
arabe, nom de note ... 592
\arabicStringNumbers ... 445
archet, indication ... 158, 441
armure ... 8, 25, 561
armure arabe ... 593
armure non traditionnelle ... 26
armure, changement avant saut de ligne ... 742
armure, grégorien ... 571
armure, visibilité après changement explicite ... 816
arpège ... 186
arpège, style parenthèse ... 189
arpège, symbole spécial ... 187
arpège, voix, enjambement ... 188
\arpeggio ... 186
arpeggio et tenue ... 67
arpeggio-direction ... 186
\arpeggioArrowDown ... 186, 960
\arpeggioArrowUp ... 186, 960
\arpeggioBracket ... 187, 960
\arpeggioNormal ... 186, 960
\arpeggioParenthesis ... 187, 960
\arpeggioParenthesisDashed ... 187, 960
arranger ... 618
Arrastre ... 296
\arrow-head ... 343, 868
\articulate ... 683
articulate.ly script ... 683
articulation ... 943
articulation grégorienne, *accentus* ... 572
articulation grégorienne, *circulus* ... 572
articulation grégorienne, *episemFinis* ... 572
articulation grégorienne, *episemInitium* ... 572
articulation grégorienne, *ictus* ... 572
articulation grégorienne, *semicirculus* ... 572
articulation, « accent » ... 158, 159
articulation, « bachschleifer » ... 158
articulation, « espressivo » ... 158
articulation, « marcato » ... 158, 159
articulation, « ouvert » ... 158
articulation, « portato » ... 158, 159
articulation, « semi-ouvert » ... 158
articulation, « staccatissimo » ... 158, 159
articulation, « staccato » ... 158, 159
articulation, « stop » ... 158
articulation, « tenuto » ... 158
articulation, liaison ... 174
articulation, script ... 944
articulation, valeur par défaut, modification ... 160
articulation-event ... 276
artificiels, harmoniques ... 442
\ascendens ... 573, 579
aspect d'un symbole de demi-bémol ... 593
\assertBeamQuant ... 961
\assertBeamSlope ... 961
associatedVoice ... 357, 359, 395
\atLeft ... 159, 961
\atRight ... 159, 961
\auctum ... 573, 579
aug ... 523
\augmentum ... 579, 961
auto-first-page-number ... 701, 980
\auto-footnote ... 903
autoBeaming ... 107, 760
\autoBeamOff ... 104, 433
\autoBeamOn ... 104
\autoBreaksOff ... 713
\autoBreaksOn ... 713
\autoChange ... 430, 961
\autoChange et mode relatif ... 431
\autoLineBreaksOff ... 713
\autoLineBreaksOn ... 713
automatique, altération accidentelle ... 32
automatique, barre de mesure ... 133
automatique, changement de portée ... 430
automatique, diagramme d'accord ... 490
automatique, diagramme de fret ... 490
\autoPageBreaksOff ... 715
\autoPageBreaksOn ... 715

B

bécarre ... 8
bécarre, supression du signe supplémentaire ... 10, 26
bémol ... 8
Bézier, points de contrôle d'une courbe ... 830
bachschleifer ... 159
\bachschleifer ... 158
backslashed digit ... 904
\backslashed-digit ... 904
bagpipe ... 512

balance MIDI ... 682
balise ... 647
balise et *markup* ... 655
balise et raccordement ... 654
balises, exclure ... 647
balises, garder ... 647
balises, regroupement ... 651
Balloon_engraver ... 309
\balloonGrobText ... 309, 961
\balloonLengthOff ... 309
\balloonLengthOn ... 309
\balloonText ... 309, 961
bandeau *(spanner)* ... 804, 958
bandeau avec rupture, modification ... 804
bandeau, modification ... 804
banjo, accordage ... 498
banjo, tablature ... 465, 498
banjo, tablature pour ... 444
banjo-c-tuning ... 498
banjo-double-c-tuning ... 498
banjo-double-d-tuning ... 498
banjo-modal-tuning ... 498
banjo-open-d-tuning ... 498
banjo-open-dm-tuning ... 498
banjo-open-g-tuning ... 498
\bar ... 123, 132, 961
bar line, in markup ... 879
\bar-line ... 879
bariton, varC, clef ... 939
bariton, varF clef ... 939
baritone, clef ... 939
BarNumber ... 136
\barNumberCheck ... 144, 961
barNumberVisibility ... 136
barré, indication de ... 469
barré, indication de la position ... 494
barrée, hampe ... 153
barrée, ligature ... 122
barre de mesure ... 123
barre de mesure automatique ... 133
barre de mesure double ... 150
barre de mesure et cadence ... 94
barre de mesure et ChordNames ... 534
barre de mesure et paroles ... 376
barre de mesure et point d'orgue ... 79
barre de mesure et soufflet ... 167
barre de mesure invisible ... 123
barre de mesure kiévienne ... 581
barre de mesure manuelle ... 124
barre de mesure par défaut, modification du style ... 132
barre de mesure pointillée ... 134
barre de mesure, césure ... 134
barre de mesure, entre portées ... 253
barre de mesure, grégorien ... 571
barre de mesure, hymnes ... 421
barre de mesure, impression à intervalle régulier ... 137
barre de mesure, impression du premier numéro ... 136
barre de mesure, liste ... 948
barre de mesure, personnalisation ... 128
barre de mesure, phrase ... 134
barre de mesure, reprise ... 134
barre de mesure, reprise dès le début ... 194
barre de mesure, reprise sous-jascente ... 134
barre de mesure, reprises successives ... 135
barre de mesure, section ... 150
barre de mesure, *segno* ... 134, 204
barre de mesure, suppression ... 94
barre de mesure, symbole au dessus ... 323
barre de reprise ... 123
barre double ... 134
barre et *segno* ... 125
barre finale ... 123, 134, 150
barre par défaut ... 134
barre simple ... 134
barre, mesure ... 134
barre, section ... 134
barre, sous-mesure ... 134
Bartók pizzicato ... 443
bartype ... 132
bas de page, note de ... 629
bas, direction (_) ... 801
base-shortest-duration ... 736
bass ... 946
bass, clef ... 939
basse chiffrée ... 540
basse chiffrée, alignement ... 547
basse chiffrée, altération, position ... 544
basse chiffrée, espacement vertical des éléments ... 547
basse chiffrée, lignes d'extension ... 544
basse continue ... 540
basse d'un accord ... 526
\bassFigureExtendersOff ... 543
\bassFigureExtendersOn ... 543
\bassFigureStaffAlignmentDown ... 546
\bassFigureStaffAlignmentNeutral ... 546
\bassFigureStaffAlignmentUp ... 546
battements par minute ... 88
batterie ... 500, 502
batterie, portée ... 248
\beam ... 868
\beamBreak ... 117
beamExceptions ... 84, 107, 961
beatBase ... 84, 107, 112
beatStructure ... 84, 107, 112
\bendAfter ... 180, 961
\bendHold ... 453, 961
\bendStartLevel ... 453, 961
binding-offset ... 699, 980
bisbigliando ... 438
bison, programme ... 957
bitmap, format de fonte ... 348
\blackTriangleMarkup ... 532
blanc ... 614
blanchiment (*whiteout*) ... 66
blank-after-score-page-penalty ... 701, 980
blank-last-page-penalty ... 701, 980
blank-page-penalty ... 701, 980
bloc d'accord ... 187
bloc de commentaire ... 609, 614
bloc de texte ... 320
blot-diameter ... 980
BNF ... 957
boîte englobante *(bounding box)* ... 803
bois, clés, liste ... 520
bois, diagramme, modification ... 518
\bold ... 333, 837
bongo ... 946
\book ... 609, 612
\book et \layout ... 611
\book et variables ... 610

book-title 980
book-title-properties 981
\bookOutputName 611, 961
\bookOutputSuffix 611, 962
\bookpart 610, 613, 715
\bookpart et \layout 611
\bookpart et variables 610
bookpart-level-page-numbering 702, 981
bookTitleMarkup 624, 981
Bottom 763
bottom-margin 694, 981
bouché 945
bounding box (boîte englobante) 803
bounding box, of glyph 913
\box 342, 837
\bp 803, 981
brève, style alternatif 55
brace, in markup 906
\bracket 171, 342, 868
\break 711
break-align-orders 742
break-align-symbol 826, 954
break-align-symbols 826
break-alignable-interface 826
break-aligned-interface 826
break-visibility 815
breakable 105
\breakAlignInsert 962
breakbefore 621
\breakDynamicSpan 170
\breathe 179, 962
\breve 53, 69
broderie 150
bulle 309

C

cæsura 77
césure 77, 571
césure kiévienne 581
césure, barre de mesure 134
césure, barre de mesure, hymnes 421
C, clef 939
cabasa 946
cadence 94, 156
cadence et alignement 156
cadence et altération 94
cadence et barres de mesure 94
cadence et ligature 94
cadence et numéro de mesure 94
cadence et saut de ligne 95
cadence et saut de page 95
cadenza 156
\cadenzaOff 94, 962
\cadenzaOn 94, 962
\caesura 77, 79, 134, 421, 572, 581, 962
callback 955
calque 230
calque *(layer)* 813
cantique 413, 417
cantique, mesure incomplète 422
capo 474
\caps 838
caractère réservé, impression 330
caractère, taille (*markup*) 334
caractères spéciaux 664
caractères spéciaux en mode *markup* 330
case 447
\cavum 573, 579
\center-align 337, 851
\center-column 340, 851
CENTER 823, 824
centering column of text 851
centrage des nuances entre deux portées de piano 427
centrage du texte sur la page 341
CFF2, format de fonte 347
chœur, citation d'instrument 407
chœur, partition de 401
chœur, système 250
\change 428
changement de fonte 333
changement de nom d'instrument 272
changement de portée 432
changement de portée automatique 430
changement de portée et collision 429
changement de portée forcé 428
changement de portée manuel 428
changement de tempo sans indication sur la partition 92
changing direction of text column 852
\char 904
check-consistency 698, 981
chevrons 221
chiffrage d'accord 527
chiffrage d'accord, exception 533
chiffrage d'accord, personnalisation 531
chiffrage de mesure 81
chiffre de n-olet inhabituel 58
chiffre de n-olet, modification 58
chiffre indicateur de mesure 81
chinese cymbal 946
ChoirStaff 761
choral 36
choral cautionary, style d'altérations 36
choral, style d'altérations 36
choral-cautionary 36
chorale et altérations 36
chorale, citation instrumentale 409
chorale, clef de ténor 21
chorale, partition 404
chord, power 497
chordChanges 487, 529
ChordGrid 536, 762
ChordGridScore 761
chordmode 522
\chordmode 6, 17, 480, 605
chordNameExceptions 533
chordNameLowercaseMinor 531
ChordNames 480, 763
ChordNames et barre de mesure 534
chordNameSeparator 532, 535
chordNoteNamer 532
chordPrefixSpacer 533
\chordRepeats 448, 962
chordRootNamer 531
\chords 528, 605
\chordSlur 187
chorus MIDI 683
Christian Harmony, tête de note 48
chute 180

\circle ... 342, 869
circling text ... 869
\circulus ... 572, 945
circulus, grégorien, articulation ... 572
citation ... 274
citation et clef ... 21
citation, fin ... 281
citation, partition chorale ... 409
clé ... 557
clés, bois, liste ... 520
clés, portées pour instrument à ... 427
clôture ... 955
claves ... 946
clavier, portées pour instrument à ... 427
claviers, centrage des nuances ... 427
clef ... 8, 557, 570, 580
\clef ... 20, 962
clef alto ... 939
clef ancienne ... 20
clef bass ... 939
clef C ... 939
clef d'alto varC ... 939
clef d'ut ... 20
clef de bariton varC ... 939
clef de bariton varF ... 939
clef de citation ... 21
clef de fa ... 20
clef de sol ... 20
clef de tablature ... 467
clef de tenor varC ... 939
clef et transposition ... 21
clef F ... 939
clef french ... 939
clef G ... 939
clef G2 ... 939
clef GG ... 939
clef kiévienne ... 941
clef mensurale ... 941
clef mensurale noire ... 941
clef moderntab ... 467
clef Petrucci ... 941
clef soprano ... 939
clef subbass ... 939
clef tenor ... 939
clef tenor G ... 939
clef transposée, visibilité ... 819
clef treble ... 939
clef varbaritone ... 939
clef varC ... 939
clef violin ... 939
clef, mezzo-soprano ... 939
clef, musique ancienne ... 941
clef, percussion ... 500
clef, percussions ... 940
clef, personnalisation des propriétés ... 23
clef, tablature ... 940
clef, visibilité après changement explicite ... 816
clef, visibilité de la transposition ... 819
clip-regions ... 667, 981
closure ... 955
cluster ... 226
\cm ... 803, 981
coche ... 179
coda ... 201
\coda ... 158, 881, 945
Coda et reprise ... 200
coda, signe ... 945
coda, signe, manuel ... 212
coda, signe, variante ... 206
\codaMark ... 134, 204, 212, 962
collision ... 233
collision de notes ... 233
collision et changement de portée ... 429
collision et numéro de mesure ... 143
collision verticale ... 733
collision, ignorer ... 226, 237
colonne ... 740
colonnes de texte ... 340
colonnes, texte ... 328
color ... 298
coloration d'objet ... 298
coloration de note ... 298
colorier des voix ... 232
coloring text ... 913
colorisée, fonte ... 348
colorisation d'objet ... 298
colorisation de note ... 298
colorisation de portées ... 300
\column ... 340, 851
\column-lines ... 914
combinaison de parties ... 238
combinateur de parties ... 238
\combine ... 343, 852
comma ... 597
commandes d'alignement du texte ... 342
commentaire ... 609
commentaire textuel ... 329
commentaire, bloc ... 614
commentaire, ligne ... 614
common-shortest-duration ... 736
Completion_heads_engraver ... 100, 587
Completion_rest_engraver ... 100, 587
complexe, métrique ... 82
composer ... 618
composite, métrique ... 82, 96
\compound-meter ... 881
\compressEmptyMeasures ... 282
\compressMMRests ... 73, 75, 283, 962
compteur, pourcent, visibilité ... 215
compteur, reprise en pourcent ... 215
\concat ... 852
concatenating text ... 852
condenser les silences ... 77
condition et *markup* ... 627
conducteur ... 404
conga ... 946
\consists ... 770, 776
contemporain, glissando ... 182
\context ... 763, 771
\context dans un bloc \layout ... 771
context-spec-music ... 236
contexte de niveau Score ... 779
contexte implicite ... 780
contexte, ajout d'un alias ... 776
contexte, création ... 763, 776
contexte, définition en MIDI ... 677
contexte, définition pour le MIDI ... 778
contexte, durée de vie ... 767
contexte, maintien actif ... 767
contexte, modification des propriétés par défaut ... 771

contexte, propriété de ... 786
contexte, référencement ... 763
contextes, ordonnancement ... 779
contextes, recouvrement ... 251
\contextPropertyCheck ... 962
contrôle de barre de mesure et reprise ... 195
contrôle des hauteurs ... 13
controlling general text alignment ... 854
controlpitch (hauteur_référence) ... 13
copyright ... 665
copyright ... 618
corde à vide, indication ... 441
corde numérotée ... 445
corde, numéro ... 441
corde, numéro, positionnement ... 292
corde, saut, tablature ... 453
cordes d'orchestre ... 440
cordes frettées et accordages prédéfinis ... 465
cordes frettées, carrure d'accord ... 483
cordes frettées, doigtés main droite ... 493
cordes frettées, harmonique ... 495
cordes frettées, indication de la position et du barré ... 494
cordes frettées, note étouffée ... 495
cordes, écriture pour ... 440
cornemuse ... 512
cornemuse : exemple ... 513
couche ... 230
couleur ... 298
couleur d'objet ... 813
couleur RGB ... 299
couleur RVB ... 299
couleur, code CSS ... 298
couleur, notes d'un accord ... 300
couleurs et dichromatisme ... 935
couleurs, liste ... 930
countPercentRepeats ... 215
coup de gratte, indication ... 101
couplet, numéro ... 388
couplet, rappel ... 389
courbes ... 173
cowbell ... 946
\cr ... 163
crash cymbal ... 946
creating a table ... 915
creating empty text object ... 907
creating horizontal space, in text ... 850, 856
creating text fraction ... 905
creating vertical space, in text ... 850, 865, 911
\cresc ... 165
crescendo ... 163
crescendo-event ... 276
crescendoSpanner ... 170
crescendoText ... 170
\crescHairpin ... 165
\crescTextCresc ... 165
crochet ... 313, 559
crochet d'accord ... 187
crochet d'accord interportée ... 434
crochet de n-olet, affichage ... 60
crochet de n-olet, positionnement ... 56
crochet de phrasé ... 313
crochet de regroupement de notes ... 313
crochet de regroupement, imbrication ... 254
crochet de regroupement, portée unique ... 252
crochet de reprise raccourci ... 206
crochet de reprise, plusieurs portées ... 207
crochet interportée ... 434
crochet plat ... 120
crochet rectiligne ... 119, 120
crochet vertical ... 250
crochet vertical, division de voix ... 403
crochets empilés ... 120
crochets, note entre ... 304
croix, tête de note ... 45
cross ... 45
\crossStaff ... 433, 963
CSS, code couleur ... 298
\cueClef ... 277, 963
\cueClefUnset ... 277, 963
\cueDuring ... 278, 963
\cueDuring et balise ... 649
\cueDuringWithClef ... 278, 963
CueVoice ... 277, 763
cuica ... 946
currentBarNumber ... 80, 135
custodes ... 555
\customTabClef ... 883
custos ... 555
cymbal, various ... 946

D

débogage, espacement vertical ... 752
décalé, grupetto ... 161
décalage ... 793
décalage de note ... 233
décalage de voix ... 233
décalage horizontal dans un *markup* ... 337
décalage vertical dans un *markup* ... 339
décallage, note ... 237
décoration du texte ... 342
défaut, barre de mesure ... 134
défaut, durée ... 54
défaut, hauteur ... 54
défaut, vertical, positionnement (-) ... 801
définition d'une barre de mesure ... 128
définition de sortie ... 760, 957
délimitation, systèmes imbriqués ... 254
dérogation, annulation des effets ... 789
désinence et tablature ... 453
D.C. ... 202
D.C., manuel ... 213
D.S. ... 202
D.S., forçage au début ... 204
D.S., manuel ... 213
da capo ... 202
da capo, manuel ... 213
dal segno ... 202
dal segno, forçage au début ... 204
dal segno, manuel ... 213
daltonien ... 935
daltonisme ... 299
\dashBang ... 160
\dashDash ... 160
\dashDot ... 160
\dashHat ... 160
\dashLarger ... 160
\dashPlus ... 160

\dashUnderscore ... 160
\deadNote ... 45, 963
\deadNotesOff ... 45
\deadNotesOn ... 45
debug-beam-scoring ... 703, 981
debug-slur-scoring ... 703, 982
debug-tie-scoring ... 703, 982
\decr ... 163
\decresc ... 165
decrescendo ... 163
decrescendoSpanner ... 170
decrescendoText ... 170
dedication ... 618
default ... 34, 630
\default ... 145, 630
default, style d'altération ... 34
default-staff-staff-spacing ... 719
\defaultchild ... 762, 780
\defaultTimeSignature ... 82
\defineBarLine ... 128, 963
demi bémol, symbole perse (koron) ... 599
demi dièse, symbole perse (sori) ... 599
demi-bémol ... 9, 12, 593
demi-dièse ... 9, 12
\deminutum ... 573, 579
\denies ... 776, 777, 779
\descendens ... 573, 579
dessus, direction (^) ... 801
Devnull ... 762, 763
dièse ... 8
diagramme d'accord pour instrument fretté ... 469
diagramme d'accord, table alternative ... 488
diagramme de fret ... 469
diagramme de fret et *markup* ... 470
diagramme de fret et transposition ... 481
diagramme de fret personnalisé ... 476
diagramme de fret personnalisé, ajout ... 482
diagramme de fret pour gaucher ... 472
diagramme de fret prédéfini, définition ... 485
diagramme de fret, personnalisation ... 484
diagramme de fret, positionnement ... 476
diagramme personnalisé de fret ... 469, 476
dichromatisme et couleurs ... 935
\dim ... 165, 523
dimension ... 803
dimension, of bounding box ... 913
dimension-variables ... 982
\dimHairpin ... 165
diminuendo ... 163
\dimTextDecr ... 165
\dimTextDecresc ... 165
\dimTextDim ... 165
\dir-column ... 852
direction ... 822
direction, bas (_) ... 801
direction, défaut (-) ... 801
direction, haut (^) ... 801
\discant ... 898
dischromatisme ... 299
\displayLilyMusic ... 686, 963
\displayMusic ... 963
\displayScheme ... 963
distance absolue ... 803
distance entre deux portées de piano ... 433
distance entre les portées ... 719
distance extensible ... 803
distance relative ... 803
divisi, portées ... 268
divisio ... 571
\divisioMaior ... 572
\divisioMaxima ... 572
\divisioMinima ... 572
division de note ... 100
division de portée ... 268
division de silence ... 100
division de voix ... 403
division et paroles ... 385
divisiones ... 571
dodecaphonic ... 37
dodecaphonic, style d'altération ... 38
dodecaphonic, style d'altérations ... 37
dodecaphonic, style néo-moderne ... 38
dodecaphonic-first ... 38
dodecaphonic-no-repeat ... 38
doigté ... 290
doigté et hampe ... 292
doigté et silence multimesure ... 77
doigté main droite, positionnement ... 493
doigté ou numéro de corde ... 445
doigté, « pouce » ... 158
doigté, ajout au diagramme de fret ... 491
doigté, dans la portée ... 292
doigté, positionnement ... 292
doigtés main droite et cordes frettées ... 493
doigtés, symboles pour vents ... 510
doigtés, table ... 510
doigtés,diagrammes pour bois ... 517
doigte, glissé ... 293
\dorian ... 25
dorien ... 25
\dotsDown ... 54
\dotsNeutral ... 54
\dotsUp ... 54
doublé, legato d'accords ... 176
double bémol ... 8
double barre ... 123, 134, 150
double dièse ... 8
double pause ... 69
double point ... 54
\doubleflat ... 883
doublement pointée, note ... 54
\doublesharp ... 883
doubleSlurs ... 176
DOWN ... 822, 824
\downbow ... 158, 441, 945
\downmordent ... 158, 944
\downprall ... 158, 944
\draw-circle ... 343, 869
\draw-dashed-line ... 870
\draw-dotted-line ... 870
\draw-hline ... 870
\draw-line ... 343, 871
\draw-squiggle-line ... 871
drawing beam, within text ... 868
drawing box, with rounded corners ... 872
drawing box, with rounded corners, around text ... 877
drawing circle, within text ... 869
drawing dashed line, within text ... 870
drawing dotted line, within text ... 870
drawing ellipse, around text ... 872

drawing line, across a page 870
drawing line, within text 871
drawing oval, around text 873
drawing path 874
drawing polygon 875
drawing solid box, within text 872
drawing squiggled line, within text 871
drawing triangle, within text 878
\dropNote 527, 963
\drummode 248, 500, 605
drumPitchNames 505
drumPitchTable 506
\drums 500, 605
DrumStaff 248, 762
drumStyleTable 505
DrumVoice 763
durée 53
durée d'un silence 69
durée isolée 54
durée par défaut 54
durée, ligne de 549
durées, échelonnement 62
\dwn 593
\dynamic 171, 838
dynamic-event 276
\dynamicDown 166
DynamicLineSpanner 166, 169
\dynamicNeutral 166
Dynamics 763
\dynamicUp 166

E

easy play, notation, numéro 46
easy play, tête de note 46
\easyHeadsOff 46
\easyHeadsOn 46
échelonnement de musique 63
échelonnement des durées 62
écrire la musique en parallèle 244
effets MIDI 682
égalisation en MIDI, adaptation 676
élément musical 740
élément non-musical 740
electric snare 946
\ellipse 872
Emmentaler, fonte 935
Emoji, fontes 348
empilés, crochets 120
empilement 740
en-tête, impression 702
\EnableGregorianDivisiones 979
\enablePerStaffTiming 963
encadrement du texte 342
encapsulated postscript 668
enclosing text, in box 837
enclosing text, in box with rounded corners 877
\endcr 163
\enddecr 163
\endSpanners 812, 964
\englishChords 531
Engraver_group 777
enroulé, accord 187
ensemble, musique d' 404
entête 615
entête de page 622
éolien 25
\episemFinis 572
episemFinis, grégorien, articulation 572
\episemInitium 572
episemInitium, grégorien, articulation 572
EPS image 873
EPS, format de sortie 668
\epsfile 344, 872
espace 614
espace dans les paroles 363
espace, dans les paroles 356
espacement 736
espacement au sein d'un système 719
espacement autour du texte 343
espacement des paroles 374
espacement entre les portées 719
espacement horizontal 736
espacement horizontal, éléments non musicaux ... 741
espacement horizontal, entre empilements 743
espacement horizontal, modification 738
espacement vertical 719
espacement vertical des éléments d'une basse chiffrée 547
espacement, affichage des valeurs 752
espacement, modification en cours de partition ... 737
espacement, ornement 154
espressivo 165
\espressivo 158, 165, 944
\etc 333, 660
étendue des objets 820
étiquette 329
étiquette de texte 320
étiquette et silence multimesure 74
étiquette textuelle 329
étiquette, crochet d'analyse 314
événementielle, note de bas de page 630
evenFooterMarkup 982
evenHeaderMarkup 982
\eventChords 964
exact positionnement des portées 726
exception, chiffrage d'accord 533
exemple de musique arabe 596
\expandEmptyMeasures 282
expansion de reprise 193
explicitClefVisibility 816
explicitKeySignatureVisibility 816
exposant 335
expression *markup* 329
expression MIDI 682
expression musicale, copie 661
expression musicale, partage 661
extenseur 321, 367
extenseur, mise en forme 321
extenseur, modification 804
extenseur, nuance, personnalisation 321
extension avec rupture, modification 804
extent, of actual inking 913
extent, of bounding box 913
extrême hauteur 10
extra-offset 719
extra-spacing-height 743
extra-spacing-width 743
extraction, fragment 667

\eyeglasses 904
Ez_numbers_engraver 46

F

\f 162
F, clef 939
fa, clef de 20
false (#f) 989
fantôme, note 304
fantôme, note, percussion 507
fausse note, percussion 507
\featherDurations 121, 964
feature, OpenType font 843
fermé 510
\fermata 79, 158, 883, 944
Ferneyhough, soufflet 168
Feta, glyphes 935
feuille de chant 529
\ff 162
\fff 162
\ffff 162
\fffff 162
fichiers, intégration au PDF 669
figured bass, alteration, glyphs 544
\figured-bass 838
FiguredBass 763
figuredBassAlterationDirection 544
figuredBassPlusDirection 544
figuredBassPlusStrokedAlist 544
\figuremode 541, 605
\figures 541, 605
\fill-line 341, 853
\fill-with-pattern 643, 854
\filled-box 343, 872
fin de citation 281
fin de note et ornement 151
fin de réplique 281
finale, barre 123, 150
finale, barre de mesure 134
finalis 571
\finalis 572
fine 201
\fine 134, 150, 199, 201, 581
\finger 291, 839, 964
fingeringOrientations 292
first-page-number 702, 982
\first-visible 904
\fixed 4, 964
flageolet 945
\flageolet 158, 945
\flagStyleDefault 120
\flagStyleStacked 120
\flat 883
flex, programme 956
\flexa 573, 579
flexe et ligatures mensurales 564
floor tom tom 946
fois, première 194
followVoice 432
fonctionnalités OpenType 350
fondamentale 523
font feature, OpenType 843
font-interface 288
\font-select 839
font-size 287, 290
\fontCaps 839
FontConfig 347
fonte 956
fonte Emmentaler 935
fonte, changement 333
fonte, définition de la taille 707
fonte, famille 336
fonte, fichiers externes 348
fonte, format OTC 348
fonte, généralités 348
fonte, localisation 348
fontes en couleur 348
fontes musicales, modification 669
fontes pour Emojis 348
fontes, choix par défaut 351
fonts 351
\fontsize 334, 839
fontSize 288
\footnote 629, 904, 964
footnote, note de bas de page 629
footnote-footer-padding 983
footnote-number-raise 983
footnote-numbering-function 983
footnote-padding 983
footnote-separator-markup 703, 983
Forbid_line_break_engraver 61
forget 38
forget, style d'altérations 38
format de fonte bitmap 348
format de fonte OTC 348
format de fonte, CFF2 347
format de fonte, Font Variations (VF) 348
format de repère 145
formatage du numéro de mesure 139
forme semai 595
four-string-banjo 498
\fp 162
\fraction 905
fragment, extraction 667
\freeBass 898
french, clef 939
\frenchChords 531
Frenched scores 265
fret 447
fret et transposition 481
fret, ajout de diagramme personnalisé 482
fret, ajout de doigté au diagramme 491
fret, diagramme 469, 479
fret, diagramme automatique 490
fret, diagramme avec nom d'accord 480
fret, diagramme et *markup* 470
fret, diagramme personnalisé 469, 476
fret, diagramme pour gaucher 472
fret, mandoline 479
fret, positionnement de diagramme 476
fret, ukulele 479
\fret-diagram 470, 892
fret-diagram-interface 476
\fret-diagram-terse 472, 893
fret-diagram-terse, *markup* 472
\fret-diagram-verbose 474, 894
fret-diagram-verbose, *markup* 474
FretBoards 479, 762

\fromproperty 905
Funk, tête de note 48
\funkHeads 48
\funkHeadsMinor 49
fusion de notes 233
fusion de tête profilée 50

G

galbe des liaisons 830
gaucher, diagramme de fret 472
\general-align 338, 854
\germanChords 531
\glide 293
glissé, doigté 293
glissando 181
\glissando 181
glissando contemporain 182
glissando d'accords et tablature 461
glissando et reprise 184, 209
glissando et tablature 462
glissando tronqué 183
glissando, marque temporelle 182
\glissandoMap 181
Global 760
globale, variable 957
glyph, bounding box 913
glyph, for figured bass alteration 544
glyphe 956
glyphe, altération 938
glyphes Feta 935
glyphes Parmesan 935
gouttière 699
grégorien, altération 571
grégorien, armure 571
grégorien, articulation, *accentus* 572
grégorien, articulation, *circulus* 572
grégorien, articulation, *episemFinis* 572
grégorien, articulation, *episemInitium* 572
grégorien, articulation, *ictus* 572
grégorien, articulation, *semicirculus* 572
grégorien, articulations 572
grégorien, barre de mesure 571
grégorien, clef 570
grégorien, ligature de neumes carrés 573
grégorien, transcription 584
grégorien, transcription moderne 418
\grace 150, 964
grammaire de LilyPond 957
GrandStaff 761
graphique, intégration 344
graphique, objet 956
graphique, tracé d'objet 342
graphisme dans la notation 343
graphisme, tracé 342
gras 333
graveur, affectation à un contexte 776
gravure manuelle, liaison de tenue 67
GregorianTranscriptionLyrics 763
GregorianTranscriptionStaff 249, 762
GregorianTranscriptionVoice 763
Grid_line_span_engraver 311
Grid_point_engraver 311
gridInterval 311
grille d'accords 536
grob 783, 956
grob, alignement 820
grob, blanchiment 66
grob, positionnement vertical 801
grob, propriétés 788
grob, surimpression 66
grob, visibilité 812
grob-interface 956
\grobdescriptions 964
grobs, superposition 813
groupement de mesures 113
groupement de note manuel 117
groupement de pulsations 113
groupetto 944
grow-direction 121
grupetto décalé 161
guidon 555
guillemets dans les paroles 363
guillemets en mode *markup* 330
guillemets, dans les paroles 356
guiro 946
guitare basse, tablature 465
guitare, coup de gratte 101
guitare, grille d'accords 101
guitare, tête de note 45
guitare, tablature pour 444

H

Hal Leonard 46
half-open high hat 158, 946
\halfopen 158, 505, 945
\halign 337, 855
hammer on 463
hampe 306
hampe barrée 153
hampe descendante 306
hampe et doigté 292
hampe et tablature 457
hampe horizontale 457
hampe invisible 306
hampe montante 306
hampe neutre 306
hampe, enjambement de portées 433
hampe, orientation 306
hampe,ligne médiane, direction 306
handclap 946
Harmonia Sacra, tête de note 48
\harmonic 45, 442, 451
\harmonicByFret 451, 964
\harmonicByRatio 451, 965
\harmonicNote 965
\harmonicsOff 441
\harmonicsOn 441, 965
harmonique et cordes frettées 495
harmonique et tablature 458
harmonique naturel 441
harmonique, indication en tablature 451
harmonique, tête de note 45
harmoniques artificiels 442
harmoniques, « flageolet » 158
harmoniques, grilles 536
\harp-pedal 439, 895

harpe ... 438
harpe sacrée, tête de note ... 48
harpe, diagramme de pédales ... 438
harpe, pédale ... 438
hauteur ... 3
hauteur approximative ... 10
hauteur explicite, trille ... 190
hauteur extrême ... 10
hauteur isolée ... 54
hauteur non spécifiée ... 10
hauteur par défaut ... 54
hauteur, nom par défaut ... 8
\haydnturn ... 158, 944
\hbracket ... 342, 873
\hcenter-in ... 856
\header ... 612
\heel ... 158, 437, 945
\heelcircle ... 158, 437, 945
Henze, point d'orgue ... 158
\henzelongfermata ... 158, 944
\henzeshortfermata ... 158, 944
\hide ... 813, 965
\hideKeySignature ... 512
\hideNotes ... 297
\hideSplitTiedTabNotes ... 450
\hideStaffSwitch ... 432
high bongo ... 946
high conga ... 946
high hat ... 946
high hat, half open ... 158
high hat, open ... 158
high hat, pédale ... 158
high timbale ... 946
high tom tom ... 946
horizontal, alignement des paroles ... 375
horizontal, alignement, mélisme ... 375
horizontal, espacement ... 736
horizontal, ornement ... 159
horizontal-shift ... 699, 983
Horizontal_bracket_engraver ... 313
HorizontalBracketText ... 314
horizontale, étendue des objets ... 820
horizontale, hampe ... 457
horizontally centering text ... 851
\hspace ... 337, 856
Hufnagel ... 552, 553
\huge ... 289, 336, 840
hymne ... 413, 417
hyperlink, as QR code ... 909

I

\ictus ... 572, 945
ictus, grégorien, articulation ... 572
identifiant ... 613
identificateur ... 613
\if ... 627, 892
\iij ... 574
\IIJ ... 574
\ij ... 574
\IJ ... 574
image ... 873
\image ... 873
image, intégration ... 344
imbrication de reprise ... 209
imbrication de systèmes ... 254
immutable, inaltérable ... 956
implicite, contexte ... 780
importing stencil, into text ... 911
impression de caractère réservé ... 330
impression de caractères spéciaux ... 330
impression, ordre ... 813
improvisation ... 52
improvisation arabe ... 596
\improvisationOff ... 52, 101
\improvisationOn ... 52, 101
\in ... 803, 983
in-note ... 638
in-note-direction ... 638
in-note-padding ... 638, 983
in-note-system-padding ... 638, 983
inaltérable, objet ... 956
inaltérable, propriété ... 956
incipit ... 583
\incipit ... 583, 965
incipit-width ... 983
\inclinatum ... 573, 579
\include ... 614, 644
inclusion de fichier ... 644
incomplète, mesure ... 93
indépendant, texte ... 328
indent ... 272, 699, 745, 983
indentation ... 699
indicateur de système ... 257
indication d'archet ... 441
indication d'octave relative ... 4
indication de corde à vide ... 441
indication de nuance, personnalisation ... 171
indication de pause ... 77
indication du barré ... 469
indication textuelle ... 323
indication textuelle, alignement ... 326
indication textuelle, empilement vertical ... 325
indication textuelle, sous la portée ... 325
indication textuelle, sur toutes les portées ... 327
indication, repère ... 144
indication, trémolo ... 217
indice ... 335
individuelle, partition ... 270
info-bulle ... 309
\inherit-acceptability ... 777, 965
\initialContextFrom ... 965
inner-margin ... 699, 983
input-language ... 957, 966
inserting music, into text ... 888
inserting PostScript directly, into text ... 876
inserting URL link, into text ... 879
\inStaffSegno ... 134, 212, 965
instrument ... 618
instrument à vent ... 509
instrument MIDI, égalisation ... 676
instrument transpositeur ... 14
instrument, centrage du nom ... 271
instrument, changement de nom ... 272
instrument, citation ... 407
instrument, nom ... 270
instrument, nom abrégé ... 270
instrument, nom complexe ... 271
instrument, nom d' ... 678

instrument, nom, autres contextes 272
instrument, script spécifique 945
\instrumentSwitch 965
intégration d'objet graphique 342
intégration de graphique 344
inter-portée, ligature 428
inter-portée, notes 428
interface 956
interfaces de rendu 783
interportée, hampe 433
interportée, trémolo 218
interportées, barre de mesure 128
intervalle de comma 597
intervalle medium 591
inversion 17
\inversion 17, 966
\invertChords 527, 966
invisible, barre de mesure 123
invisible, note 297
\ionian 25
ionien 25
is-last-bookpart 983
isolée, durée 54
isolée, hauteur 54
\italianChords 531
\italic 333, 840
italique 333

J

jazz, accord 531
\jump 213, 966
justifié, texte 341
\justified-lines 332, 914
\justify 341, 857
\justify-field 857
\justify-line 858
\justify-string 859
justifying lines of text 914
justifying text 857

K

\keep-with-tag 655, 906
keepAliveInterfaces 266
\keepWithTag 647, 966
\key 25, 49, 966
kiévien 579
kiévienne, altération 581
kiévienne, clef 557, 580, 941
kiévienne, ligature 581
kiéviennne, tête de note 580
Kiev 579
Kievan 579
Kievan_ligature_engraver 581
\kievanOff 580
\kievanOn 580
KievanStaff 579, 762
KievanVoice 579, 763
\killCues 281, 966
Kirchenpause 284
koron 599

L

\label 640, 966
label-alist-table 983
label-page-table 984
laissez vibrer 65
\laissezVibrer 65
landscape 984
landscape, papier 693
\language 966
\languageRestore 966
\languageSaveAndChange 966
langue, nom de note 11
\large 289, 336, 840
\larger 334, 336, 840
last-bottom-spacing 697, 731, 984
layer (calque) 813
\layout 612, 705, 760, 771
\layout et \book 611
\layout et \bookpart 611
layout, débogage 752
layout-set-staff-size 707
lead sheet 529
\left-align 337, 859
left-aligning text 859
\left-brace 906
\left-column 860
left-margin 698, 984
LEFT 822, 823
legato 174
levée 93
levée dans une reprise 195
lexer, pour LilyPond 956
\lheel 436
\lheeltoe 436
liaison d'articulation 174
liaison de n-olet 57
liaison de phrasé 177
liaison de prolongation 64
liaison de prolongation et reprise avec alternative . 65
liaison de prolongation, apparence 65
liaison de prolongation, pointillés 65
liaison de prolongation, positionnement 65
liaison de prolongation, tirets 65
liaison de tenue 64
liaison de tenue et accord 64
liaison de tenue et altération 9
liaison de tenue et répétition 65
liaison doublée, pour legato d'accords 176
liaison et reprise 209
liaison et texte 176
liaison verticale 187
liaison, ajustement des extrémités 175
liaison, au-dessous des notes 174
liaison, au-dessus des notes 174
liaison, définition du pointillé 176
liaison, dans les paroles 363
liaison, densité des tirets 175, 178
liaison, inclusion de texte 176
liaison, laissez vibrer 65
liaison, modification 830
liaison, orientation manuelle 174
liaison, style de trait 174, 178
liaison, tirets 174, 178
liaison, tirets et trait continu 174, 178

liaison, trait plein 174, 178
liaison, trait pointillé 174, 178
liaisons d'articulation multiples 174
liaisons d'articulation simultanées 174
liaisons de phrasé multiples 178
liaisons de phrasé simultanées 178
lien de croche 104
ligature 104, 554, 581
\ligature 966
ligature ancienne 563
ligature barrée 122
ligature blanche 563
ligature coudée, personnalisation 105
ligature en fin de partition 114
ligature en fin de voix polyphonique 114
ligature en soufflet 121
ligature et cadence 94
ligature et mélisme 104
ligature et métrique 84
ligature et musique polymétrique 96
ligature et neumes 573
ligature et paroles 107
ligature et saut de ligne 105
ligature inter-portée 428
ligature manuelle 104, 117
ligature manuelle et note d'ornement 117
ligature manuelle, orientation 117
ligature mensurale, flexe 564
ligature, définition de règles 104
ligature, in text 852
ligature, indication des subdivisions 113
ligature, n-olet, saut de ligne 61
ligature, \partCombine et \autoBeamOff 106
ligature, personnalisation 104
ligature, subdivision 112
Ligature_bracket_engraver 563, 581
ligne de commentaire 609, 614
ligne de prolongation, nuance, masquage 170
ligne médiane, hampe, direction 306
ligne supplémentaire 258
ligne, indication de durée 549
ligne, longueur 745
ligne, passer à la suivante 711
lignes 181
lignes de portée, épaisseur 258
lignes de portée, nombre 258
\line 331, 332, 860
line-break-system-details 726
line-thickness 984
line-width 341, 697, 745, 984
\linea 573, 579
\lineprall 158, 944
liste associative 955
liste associative, modification 798
liste de *markups* 330
liste des couleurs 930
liturgie et musique 413
\locrian 25
locrien 25
\longa 53, 69
\longfermata 158, 944
longueur d'un silence multimesure 75
longueur de note 53
longueur minimale, soufflet 167
\lookup 906
losange, tête de note 45, 441
low bongo 946
low conga 946
low timbale 946
low tom tom 946
\lower 338, 860
lowering text 860
\ltoe 436
\ltoeheel 436
luth, personnalisation 499
luth, tablature 499
ly:minimal-breaking 716
ly:one-line-auto-height-breaking 717
ly:one-line-breaking 717
ly:one-page-breaking 717
ly:optimal-breaking 716
ly:page-turn-breaking 717
\lydian 25
lydien 25
\lyricmode 356, 357, 606
Lyrics 763
\lyrics 606
\lyricsto 357, 359

M

m (dans un accord) 523
mélisme 364, 367
mélisme et ligature 104
mélisme, alignement horizontal 375
mélodie alternative 395
mélodie d'une portée à une autre 432
mélodie, affichage du rythme seul 101
métadonnées MIDI 628
métadonnées PDF 628
métrique 81
métrique arabe 595
métrique complexe 82
métrique composite 82, 96
métrique double 96
métrique en cours de mesure 93
métrique et ligature 84
métrique et reprise 87
métrique par défaut 84
métrique polymétrique 96
métrique, mise en évidence 134
métrique, numérateur seulement 86
métrique, retour au propriétés par défaut 85
métrique, style 82, 558
métrique, visibilité 81
métronome, indication 88
magnification->font-size 288, 707
\magnify 334, 841
magnifying text 841
\magnifyMusic 286, 966
\magnifyStaff 707, 966
magstep 288, 707, 803
main droite, doigté, positionnement 493
main droite, doigtés pour cordes frettées 493
maj 523
majeur 25
\major 25
majorSevenSymbol 532
makam 591, 598

makam, exemple ... 598
makamlar ... 591, 597, 598
make-dynamic-script ... 172
make-dynamic-script ... 172
make-footer ... 984
make-header ... 984
make-relative ... 662
\makeClusters ... 226, 966
\makeDefaultStringTuning ... 967
mandoline, tablature ... 465
manuel, saut de page ... 715
manuelle, barre de mesure ... 124
maqam ... 591, 597
maracas ... 946
\marcato ... 158, 944
marge, texte qui dépasse ... 320
\mark ... 144, 323, 967
Mark_engraver ... 146, 327
\markalphabet ... 906
\markLengthOff ... 90, 325
\markLengthOn ... 90, 325
\markletter ... 907
markup ... 329
\markup ... 317, 323, 328, 329, 606
markup conditionnel ... 627
markup et balise ... 655
markup et décalage horizontal ... 337
markup et décalage vertical ... 339
markup et diagramme de fret ... 470
markup et paroles ... 357
markup et silence multimesure ... 76
markup multiligne ... 340
markup multipage ... 332
markup, centrage sur la page ... 341
markup, commandes d'alignement du texte ... 342
markup, encadrement du texte ... 342
markup, expression ... 329
markup, image ... 873
markup, inclusion de musique ... 345
markup, inclusion de partition ... 346
markup, largeur de ligne ... 341
markup, liste ... 330
markup, objet ... 317
markup, ornementation du texte ... 342
markup, rembourrage du texte ... 343
markup, rhythm ... 887
markup, rotation ... 819
markup, syntaxe ... 329
markup, tempo ... 92
markup, texte au kilomètre ... 341
markup, texte justifié ... 341
markup, texte multipage ... 332
markup-markup-spacing ... 697, 984
markup-system-spacing ... 696, 984
\markuplist ... 328, 332
\markupMap ... 967
markups, alignement ... 337
marque temporelle, glissando ... 182
masquée, note ... 297
masquage de hampe ... 306
masquage de portée ... 265
masquage, nuance, ligne de prolongation ... 170
matériel d'orchestre ... 270
matter, optional ... 304
max-systems-per-page ... 700, 985
\maxima ... 53, 69
maximum, hauteur ... 10
\measure ... 967
Measure_grouping_engraver ... 113
measureBarType ... 132
measureLength ... 80, 107
measurePosition ... 80
\measureRemainder ... 967
Medicaea, Editio ... 552, 553
\melisma ... 364
\melismaEnd ... 364
mensural ... 552, 553
mensural, transcription ... 587
Mensural_ligature_engraver ... 563
mensurale noire, clef ... 557, 941
mensurale, clef ... 557, 941
MensuralStaff ... 249, 556, 561, 762
MensuralVoice ... 556, 561, 762
mensuration, signe ... 558
Mensurstriche ... 583
\mergeDifferentlyDottedOff ... 233
\mergeDifferentlyDottedOn ... 233
\mergeDifferentlyHeadedOff ... 233
\mergeDifferentlyHeadedOn ... 233
merging text ... 852, 861
mesure ... 80
mesure à compter ... 73
mesure de silence ... 69
mesure entière de silence ... 73
mesure incomplète ... 93
mesure tronquée et numéro de mesure ... 138
mesure, compteur ... 147
mesure, numéro de ... 80
mesure, numérotation ... 135
mesure, numérotation et reprise ... 207
mesure, position dans la ... 93
mesure, répétition de ... 213
mesure, subdivision ... 113
mesure, vérification des limites ... 144
meter ... 618
mezzo-soprano, clef ... 939
\mf ... 162
micro-intervalle, tablature ... 467
microtonalité ... 12
mid tom tom ... 946
\midi ... 612, 760
MIDI ... 672
MIDI et reprise ... 679
MIDI et transposition ... 30
MIDI, égalisation ... 675
MIDI, égalisation par défaut, adaptation ... 676
MIDI, éléments non pris en compte ... 673
MIDI, éléments pris en compte ... 672
MIDI, canaux ... 680
MIDI, définition de contexte ... 677, 778
MIDI, effets ... 682
MIDI, gestion des nuances ... 674
MIDI, indications de nuance ... 674
MIDI, instrument ... 678
MIDI, le bloc ... 673
MIDI, métadonnées ... 628
MIDI, nuance personnalisée ... 674
MIDI, pistes ... 680
MIDI, réglage du volume ... 675
MIDI, un canal par voix ... 681

midiBalance ... 682
midiChannelMapping ... 680
midiChorusLevel ... 683
midiDrumPitches ... 506
midiExpression ... 682
midiInstrument ... 678
midiPanPosition ... 682
midiReverbLevel ... 682
min-systems-per-page ... 700, 985
mineur ... 25
minimum, hauteur ... 10
minimum-Y-extent ... 719
minimumFret ... 447, 492
\minor ... 25
minorChordModifier ... 533
mirroring markup ... 878
mise en forme ... 705, 707
\mixolydian ... 25
mixolydien ... 25
\mm ... 803, 985
modèle de musique arabe ... 596
modale, inversion ... 19
modale, transformation ... 18
modale, transposition ... 18
\modalInversion ... 19, 967
\modalTranspose ... 18, 967
mode ... 25
mode ancien ... 25
mode *markup* et caractères spéciaux ... 330
mode *markup* et guillemets ... 330
mode relatif et accord ... 222
mode relatif et \autoChange ... 431
modern ... 34
modern cautionary, style d'altération ... 35
modern voice, style d'altération ... 35
modern voice, style d'altération de précaution ... 35
modern, style d'altération ... 34
modern, style d'altération de précaution ... 35
modern-cautionary ... 35
modern-cautionary, style d'altération ... 34
modern-voice ... 35
modern-voice-cautionary ... 35
moderntab, clef ... 467
modification du style par défaut des barres de mesure ... 132
modification, respiration, signe ... 179
modifier des propriétés ... 786
\mordent ... 158, 944
motet ... 401
mouvements, plusieurs ... 609
\mp ... 162
multi-measure rest, within text, by duration ... 886
multi-measure rest, within text, by number of measures ... 884
\multi-measure-rest-by-number ... 884
multiligne, *markup* ... 340
multiligne, texte ... 340
MultiMeasureRestScript ... 74
MultiMeasureRestText ... 74
multimesure, note, contraction ... 282
multimesure, note, expansion ... 282
multimesure, silence, contraction ... 283
multimesure, silence, expansion ... 283
multimesure, silence, nombre ... 284
multimesure, silence, style ... 284
multipage, *markup* ... 332
multiple, liaison de phrasé ... 178
multiples voix ... 233
multiples, liaisons d'articulation ... 174
multivoix, altérations ... 35, 37
musica ficta ... 561
musical, élément ... 740
musicale, citation ... 407
\musicglyph ... 884
\musicLength ... 967
\musicMap ... 968
musicologie, crochet d'analyse ... 313
musicQuotes ... 957
musique échelonnée ... 63
musique ancienne, clef ... 941
musique ancienne, masquage de portée ... 265
musique ancienne, transcription ... 253, 587
musique arabe ... 591
musique arabe, exemple ... 596
musique dans un *markup* ... 345
musique en parallèle ... 244
musique entremêlée ... 244
musique et note de bas de page ... 629
musique non mesurée ... 80, 94
musique non mesurée et saut de ligne ... 95
musique non mesurée et saut de page ... 95
musique ottomane ... 597
musique répétitive ... 193
musique turque ... 597
mutable ... 956
mute bongo ... 946
mute conga ... 946
mute timbale ... 946

N

\n ... 162
n-olet ... 56
n-olet, chiffrage inhabituel ... 58
n-olet, crochet côté tête de note ... 60
n-olet, formatage ... 58
n-olet, liaison ... 57
n-olet, ligature, saut de ligne ... 61
n-olet, modification du chiffre ... 58
n-olet, positionnement du crochet ... 56
n-olet, regroupment ... 56
n-olet, visibilité du crochet ... 57, 59
n-olet, visibilité du nombre ... 57
n-olets successifs ... 58
N.C., personnalisation du symbole ... 530
N.C., symbole ... 528
\name ... 776
\natural ... 884
naturel ... 8
naturel, harmonique ... 441
neo-modern ... 36
neo-modern, style d'altérations ... 36
neo-modern-cautionary ... 37
neo-modern-cautionary, style d'altérations ... 37
neo-modern-voice ... 37
neo-modern-voice, style d'altération ... 37
neo-modern-voice-cautionary ... 37
neomensural ... 553
neume carré et ligature ... 573
\new ... 763

\newSpacingSection ... 737
niente, al, soufflet ... 168
no reset, style d'altérations ... 38
no-reset ... 38
\noBeam ... 117
\noBeamBreak ... 117
\noBreak ... 713
noChordSymbol ... 530
nom d'instrument ... 270
nom d'instrument, autres contextes ... 272
nom d'instrument, centrage ... 271
nom d'instrument, complexe ... 271
nom de note ... 3
nom de note, hollandais ... 8
nom de note, langue ... 11
nom de note, par défaut ... 8
nom de personnage ... 393
nom du chanteur ... 393
nombre de portées ... 262
noms d'instrument, centrés ... 271
non mesurée, musique ... 80
non musical, symbole ... 343
non spécifiée, hauteur ... 10
non traditionnelle, armure ... 26
non-arpeggiato ... 187
non-ASCII, caractères ... 664
non-musical, élément ... 740
\nonArpeggiato ... 187, 403, 434
NonMusicalPaperColumn ... 726, 740
nonstaff-nonstaff-spacing ... 719
nonstaff-relatedstaff-spacing ... 719
nonstaff-unrelatedstaff-spacing ... 719
\noPageBreak ... 715, 968
\noPageTurn ... 718, 968
\normal-size-sub ... 841
\normal-size-super ... 335, 841
\normal-text ... 841
\normal-weight ... 842
\normalsize ... 287, 289, 336, 842
\noStanza ... 389, 968
notation dans un *markup* ... 345
notation et graphisme ... 343
notation facile ... 46
notation, explication ... 309
notation, taille ... 286
\note ... 885
note étouffée et cordes frettées ... 495
note colorisée dans un accord ... 300
note d'ornement ... 150
note d'ornement en fin de note ... 151
note d'ornement et retouche ... 152
note d'ornement, mise en forme ... 152
note d'ornement, synchronisation ... 155
note en couleur ... 298
note fantôme ... 304
note invisible ... 297
note masquée ... 297
note penchée ... 52
note pointée ... 54
note pointée, déplacement horizontal ... 237
note pointée, nombre de points ... 55
note profilée ... 48
note profilée, Aiken, variante fine ... 50
note profilée, fusion ... 50
note silencieuse ... 69
note tenue au début du système suivant, masquage de l'altération ... 9
note, décalage ... 233
note, décallage ... 237
note, division ... 100
note, impression du nom ... 307
note, longueur ... 53
note, nom selon la langue ... 11
note, prolongation ... 549
note, tête ancienne ... 558
note, taille standard ... 290
note, within text, by duration ... 885
note, within text, by log and dot-count ... 885
\note-by-number ... 885
note-event ... 276
Note_heads_engraver ... 100
Note_name_engraver ... 307
NoteColumn ... 740
\notemode ... 606
noteNameFunction ... 307
NoteNames ... 307, 763
noteNameSeparator ... 307
notes inter-portée ... 428
notes simultanées et altération ... 39
notes, espacement horizontal ... 737
notes, nom en arabe ... 592
nouvelle portée ... 248
nuance personnalisée ... 171
nuance textuelle, style ... 170
nuance, alignement vertical ... 169
nuance, extension, personnalisation ... 321
nuance, ligne de prolongation, masquage ... 170
nuance, MIDI, personnalisation ... 674
nuances ... 162
nuances éditoriales ... 171
nuances entre les portées d'un système pianistique ... 427
nuances entre parenthèses ... 171
nuances et MIDI ... 674
nuances et MIDI, indication ... 674
nuances successives sur une note tenue ... 164
nuances suggérées ... 171
nuances, positionnement vertical ... 166
nuances, rupture d'alignement vertical ... 170
\null ... 907
NullVoice ... 386, 763
numéro de corde ... 445
numéro de corde ou doigté ... 445
numéro de corde, positionnement ... 292
numéro de couplet ... 388
numéro de mesure ... 80, 135, 138
numéro de mesure à intervalle régulier ... 136
numéro de mesure et cadence ... 94
numéro de mesure et collision ... 143
numéro de mesure et mesure tronquée ... 138
numéro de mesure et reprise ... 207, 209
numéro de mesure, alignement ... 141
numéro de mesure, centrage ... 142
numéro de mesure, formatage ... 139
numéro de mesure, supression ... 142
numéro de page, référencement ... 639
numéro, notation *easy play* ... 46
numérotation des mesures, suppression ... 94
numérotation des pages ... 702
\number ... 842
number-footnote-table ... 985

\numericTimeSignature 82

O

objet altérable 956
objet de propriété 958
objet de rendu 956
objet graphique, interface 956
objet graphique, propriétés 788
objet graphique, tracé 342
objet *markup* 317
objet Scheme 958
objet texte 317
objet, couleur 813
objet, rotation 819
objet, surimpression 813
objet, visibilité 812
objets graphiques 783
objets, alignement 820
octave absolue 3
octave et clef 21
octave relative et accord 7
octave relative et fonction de substitution 662
octave relative et transposition 6
octave relative, indication 4
octave, spécification 3
octave, vérification 13
\octaveCheck 13, 968
octaviation 27, 819
octaviation optionnelle 21
octaviation, allure du prolongateur 30
octaviation, texte 29
octaviation, une seule voix 29
oddFooterMarkup 985
oddHeaderMarkup 985
\offset 793, 968
offset (décalage) 793
\omit 812, 968
\on-the-fly 907
on-the-fly (à la volée) 627
\once 790, 795, 805, 968
OneStaff 251, 761
\oneVoice 227
opéra 404
opérette 404
\open 158, 441, 510, 945
open bongo 946
open conga 946
open high hat 158, 946
open timbale 946
OpenType collections (.otc) 348
OpenType, fonctionnalités 350
OpenType, font feature 843
OpenType, fonte (.otf) 347
optional material 304
optional passage 304
optionnelle, octaviation 21
opus 618
oratorio 401
orchestrale, musique 404
orchestre, cordes 440
orchestre, matériel d' 270
orchestre, notation pour 404
orenement, espacement 154
orgue 436
orgue, indication de pédalier 436
orgue, indication, pédalier 158
orgue, marque de pédale 945
orgue, point d' 944
\oriscus 573, 579
ornement 150, 943
ornement et ligature 117
ornement et paroles 394
ornement personnalisé 153
ornement, « grupetto barré » 158
ornement, « grupetto Haydn » 158
ornement, « grupetto renversé » 158
ornement, « grupetto » 158
ornement, « mordant ascendant » 158
ornement, « mordant descendant » 158
ornement, « mordant » 158
ornement, « signum congruentiae » 158
ornement, « trille » 158
ornement, horizontal 159
ornement, script 944
orphan-penalty 985
ossia 262, 270
ossia, positionnement 264
ottava 27
\ottava 27, 968
ottavation 29
ottavation-numbers 28
ottavation-ordinals 28
ottavation-simple-ordinals 28
ottavationMarkups 28
ottomane, musique classique 591
outer-margin 699, 985
output-count 957
output-def 957
output-filename 611, 985
output-scale 985
output-suffix 611, 957, 985
outside-staff-horizontal-padding 734
outside-staff-padding 734
outside-staff-priority 733, 952
ouvert 510, 945
ouvert, soufflet 168
\oval 873
\overlay 861
\override 788, 790, 907
override ponctuel 790
override, annulation des effets 789
\override-lines 915
\overrideProperty 790, 968
\overrideTimeSignatureSettings 85, 968
overriding property, within text markup 907
\overtie 844

P

\p 162
pédale de harpe 438
pédale de piano 435
pédale high hat 158
pédale sostenuto 435
pédale *sustain* 435
pédale *sustain*, style 435
pédale, diagramme pour harpe 438
pédale, indication combinée de 435

pédale, indication graphique de ... 435
pédale, indication textuelle de ... 435
pédale, style d'indications de ... 435
pédalier d'orgue, indication ... 436
pédalier, « pointe » ... 158
pédalier, « talon » ... 158
pédalier, orgue, indication ... 158
\pad-around ... 343, 861
\pad-markup ... 343, 861
\pad-to-box ... 343, 862
\pad-x ... 343, 862
\pad-x-left ... 343, 862
\pad-x-right ... 343, 863
padding ... 784
padding text ... 861
padding text horizontally ... 862, 863
page, entête ... 622
page, format ... 693
page, mise en forme ... 746
page, numéro de la première ... 702
page, numérotation ... 702
page, numérotation automatique ... 701
page, numérotation en chiffres romains ... 702
page, numérotation indépendante pour l'introduction ... 702
page, numérotation par bookpart ... 702
page, pied ... 622
page, première ... 702
page, référencement du numéro ... 639
page, saut ... 745
page-breaking ... 700, 985
page-breaking-system-system-spacing ... 700, 985
page-count ... 700, 986
\page-link ... 907
page-number-type ... 702, 986
page-post-process ... 986
\page-ref ... 640, 908
page-spacing-weight ... 701, 986
\pageBreak ... 715, 969
\pageTurn ... 718, 969
pageTurnMinimumRepeatLength ... 718
pageTurnMinimumRestLength ... 718
\palmMute ... 969
\palmMuteOn ... 969
pan, MIDI ... 682
Pango ... 347
\paper ... 612, 693
paper-height ... 694, 986
paper-width ... 697, 986
papersizename ... 986
papier, orientation ... 693
papier, taille ... 693
papier, variable prédéfinie ... 692
\parallelMusic ... 244, 969
parent-alignment-X ... 823
parent-alignment-Y ... 824
parenthèses, altération entre ... 9
parenthèses, note entre ... 304
\parenthesize ... 304, 874, 969
parlato ... 411
parlato, tête de note ... 45
Parmesan, glyphes ... 935
parole séparées, ligne unique ... 372
paroles ... 356
paroles assignées à une voix ... 227
paroles communes à plusieurs voix ... 386
paroles divisées (reprises) ... 385
paroles et barre de mesure ... 376
paroles et liaison de prolongation ... 382
paroles et ligature ... 107
paroles et mélodies ... 359
paroles et *markup* ... 357
paroles et ornement ... 394
paroles et \partCombine ... 241
paroles et répétition ... 377
paroles, alignement horizontal ... 375
paroles, alignement sur la mélodie ... 357
paroles, alignement sur une mélodie épisodique .. 768
paroles, blanc ... 71
paroles, garder dans les marges ... 376
paroles, gestion de l'espacement ... 374
paroles, identificateur ... 368
paroles, mise en forme ... 356
paroles, positionnement ... 264, 370
paroles, reprise avec alternative ... 381
paroles, saut de note ... 71
paroles, variable ... 368
parser ... 957
parser, variable ... 957
\partCombine ... 238, 386, 969
\partCombine et paroles ... 241, 386
\partCombineApart ... 240
\partCombineAutomatic ... 240
\partCombineChords ... 240
\partCombineDown ... 969
\partCombineForce ... 969
partCombineListener ... 957
\partCombineSoloI ... 240
\partCombineSoloII ... 240
\partCombineUnisono ... 240
\partCombineUp ... 970
\partial ... 80, 93, 195, 970
partie vocale ... 401
parties combinées, texte ... 243
parties, combiner des ... 238
partition à la française ... 265
partition incluse dans un *markup* ... 346
\partocombine, texte ... 243
\path ... 874
path, drawing ... 874
\pattern ... 908
paysage, papier ... 693
PDF, intégration de fichiers ... 669
PDF, métadonnées ... 628
pedal high hat ... 946
pedalSustainStyle ... 435
percussion, clef ... 500
percussion, fausse note ... 507
percussion, note fantôme ... 507
percussion, note silencieuse ... 507
percussion, portée ... 248
percussions ... 500, 502
percussions, clef ... 940
percussions, nom des notes ... 946
percussions, personnalisation ... 505
Performer_group ... 778
perse, altérations (sori, koron) ... 599
perse, makam ... 591
Perse, musique classique ... 599
personnage, indication ... 405

personnalisation de tablature 465
personnalisation, diagramme de fret 484
\pes 573, 579
petite note 150
petites notes 274, 512
petites notes, formater des 277
Petrucci 552, 553
Petrucci, clef 557, 941
PetrucciStaff 564, 762
PetrucciVoice 564, 763
phrasé, liaison de 177
phrasé, pour des paroles 364
phrase, barre de mesure 134
phrase, barre de mesusre, hymnes 421
\phrasingSlurDashed 178
\phrasingSlurDashPattern 178, 970
\phrasingSlurDotted 178
\phrasingSlurDown 177
\phrasingSlurHalfDashed 178
\phrasingSlurHalfSolid 178
\phrasingSlurNeutral 177
\phrasingSlurSolid 178
\phrasingSlurUp 177
\phrygian 25
phrygien 25
piano 35
piano cautionary, style d'altérations 36
piano et altérations 35, 36
piano et pédale 435
piano, nuances entre les portées 427
piano, style d'altérations 35
piano, système 251
piano, système pour 427
piano-cautionary 36
PianoStaff 427, 430, 761
piece 618
pied de page 615, 622
pied, indication 158
pipe, symbole 144
Pitch_squash_engraver 101
\pitchedTrill 190, 970
pitchnames 957, 966
pizzicato, Bartók 443
pizzicato, *snap* 443
place-fret 894
placing horizontal brackets, around text 873
placing parentheses, around text 874
placing vertical brackets, around text 868
plaqué, accord 187
plat, crochet 120
plat, soufflet 168
plein, trait de liaison 174, 178
plusieurs lignes de texte 341
plusieurs mouvements 609
plusieurs pages de texte 332
PNG image 873
poet 618
point 54
point d'alignement 820
point d'arrêt 179, 944
point d'augmentation, modification du nombre 55
point d'orgue 158, 944
point d'orgue et silence multimesure 74
point d'orgue sur barre de mesure 79
point de contrôle et *tweak* 793
point de référence 820
pointée, note 54
\pointAndClickOff 970
\pointAndClickOn 970
\pointAndClickTypes 970
pointe 945
pointillé, liaison 176
pointillé, trait de liaison 174, 178
pointillée, barre de mesure 134
points de contrôle, courbe de Bézier 830
polices, choix par défaut 351
\polygon 875
polymétrie 96
polymétrie et ligature 96
\polymetric 96, 970
polyphone et tablature 457
polyphonie 233
polyphonie, mêmes paroles 386
polyphonie, portée simple 227
polyphonie, voix additionnelle 236
ponctuation 356
ponctuation et paroles 356
\popContextProperty 787, 971
portée à la française 262
portée à quatre mesures 713
portée de batterie 248
portée de percussion 248
portée de piano 427
portée multiple 250
portée rythmique 249
portée rythmique, masquage 265
portée simple 248
portée simple et polyphonie 227
portée unique avec crochet de regroupement 252
portée vide 265
portée, changement automatique 430
portée, division 268
portée, initialisation 248
portée, instanciation 248
portée, lignes de 258
portée, nouvelle 248
portée, reprise 258
portée, surlignage 300
portée, suspension 258
portée, taille 707
portée, transcription de grégorien 249
portées pour instrument à clés 427
portées pour instrument à clavier 427
portées, espacement 719
portées, groupe de 250
portées, nombre variable de 262
portées, positionnement exact 726
portées, regroupement 250
portées, regroupement et imbrication 254
\portato 158, 944
position dans la mesure 80, 93
position, vertical, *grobs* 801
positionnemenet, diagramme de fret 476
positionnement des paroles 370
positionnement des silences multimesure 75
positionnement vertical des *grobs* 801
positionnement, doigté 292
positionnement, doigté main droite 493
positionnement, numéro de corde 292
positionnement, ossia 264

positionnement, paroles ... 264
postévénement ... 943
postscript ... 344
\postscript ... 344, 876
pouce *(thumb)* ... 944
pouce, doigté ... 291
pourcent, compteur de reprise ... 215
pourcent, répétition ... 213
pourcent, reprise isolée ... 215
pousser l'archet ... 945
power chords ... 497
\pp ... 162
\ppp ... 162
\pppp ... 162
\ppppp ... 162
précaution, altération de ... 9
prédéfini, diagramme de fret, définition ... 485
préfabriqué, matériel ... 740, 743
présentation mensurale ... 253
\prall ... 158, 944
\pralldown ... 158, 944
\prallmordent ... 158, 944
\prallprall ... 158, 944
\prallup ... 158, 944
\preBend ... 453, 971
\preBendHold ... 453, 971
predefinedDiagramTable ... 488
\predefinedFretboardsOff ... 491
\predefinedFretboardsOn ... 491
\premeasure ... 971
première fois ... 194
première page ... 702
prima volta ... 195
print-all-headers ... 702, 986
print-first-page-number ... 702, 986
print-page-number ... 702, 986
printAccidentalNames ... 307
printNotesLanguage ... 307
printOctaveNames ... 307
prob ... 958
prologation de note ... 549
prolongateur ... 367
prolongateur, octaviation ... 30
prolongation de texte ... 321
property-defaults ... 986
\property-recursive ... 908
\propertyOverride ... 971
\propertyRevert ... 971
\propertySet ... 971
\propertyTweak ... 971
\propertyUnset ... 972
proportionnel, espacement ... 746
propriété altérable ... 956
propriété commune ... 956
propriété objet ... 958
propriété partagée ... 956
propriétés d'objet graphique ... 788
propriétés d'un *grob* ... 788
propriétés, modification ... 786, 792
psalmodie ... 413
psaume ... 417
pseudo-arpège ... 187
\pt ... 803, 986
pull off ... 463
pulsation, regroupement ... 113
pupitre, partition de ... 270
\push-to-tag ... 655, 908
\pushContextProperty ... 787, 972
\pushToTag ... 654, 972
\pushToTagMarkup ... 656, 972
\put-adjacent ... 863
putting space around text ... 861

Q

q, répétition d'accord ... 223, 448
\qr-code ... 909
QR code ... 909
quadrillage temporel ... 311
quadrillage temporel, apparence ... 312
qualité d'accord ... 523
quart de ton ... 9, 10
quart de ton, tablature ... 467
\quilisma ... 573, 579
quotedCueEventTypes ... 276
quotedEventTypes ... 276
\quoteDuring ... 274, 277, 972
\quoteDuring et balise ... 649

R

r (*rest*, silence) ... 69
référence, point de ... 820
réglage fin d'un luth ... 499
réglage par défaut, modification ... 771
réglages, globalisation ... 658
répétition ... 126, 193
répétition courte ... 213
répétition d'accord, utilisation de q ... 223
répétition de mesure ... 213
répétition et liaison de tenue ... 65
répétition, pourcent ... 213
répétition, utilisation de q ... 448
réplique ... 274
réplique, fin ... 281
réplique, formatage ... 277
rétrograde, transformation ... 17
rôle ... 393
rôle, indication ... 405
R (mesure de silence) ... 73
Référence des propriétés internes ... 759
raccordement dans une balise ... 654
ragged, aligné ... 691
ragged-bottom ... 695, 986
ragged-last ... 698, 746, 987
ragged-last-bottom ... 695, 987
ragged-right ... 698, 745, 987
\raise ... 338, 863
\raiseNote ... 527, 972
raising text ... 863
rappel ... 955
Ratisbona, Editio ... 553
recouvrement de contextes ... 251
rectangle en front de regroupement ... 252
rectiligne, crochet ... 119, 120
\reduceChords ... 102, 972
referencing page label, in text ... 913
referencing page number, in text ... 907, 908
registre, symbole pour accordéon ... 437

regroupement de balises 651
regroupement de n-olets 56
regroupement, rectangle en front 252
relatif 4
\relative 4, 6, 17, 431, 662, 972
religieuse, musique 413
reliure 699
\remove 770
remove-empty 266
remove-first 266
remove-grace-property 153
remove-layer 268
\remove-with-tag 655, 909
\RemoveAllEmptyStaves 266, 979
\RemoveEmptyStaves 266, 980
\removeFromTagGroup 653, 972
\removeWithTag 647, 972
renaissance, musique 253
rendu, interfaces de 783
rendu, objet de 956
repère manuel 145
repère textuel 323
repère, format 145
repère, indication de 144
repère, personnalisation 145
repère, sous la portée 91
repère, style 145
repère, sur toutes les portées 146
\repeat 193, 194, 195
\repeat percent 213
\repeat segno 134, 200
\repeat tremolo 217
\repeat unfold 193
\repeat volta 134, 194, 195
repeatCommands 210
repeatCountVisibility 215
\repeatTie 65, 382
\replace 845
\replace-with-tag 655, 910
\replaceWithTag 655, 973
\replaceWithTagMarkup 656, 973
reprise 126, 193
reprise *al fine* 201
reprise *alla coda* 201
reprise ambiguë 209
reprise avec alternative et liaison de prolongation . 65
reprise avec alternative et paroles 381
reprise avec levée 195
reprise courante 194
reprise développée 679
reprise de portée 258
reprise *e poi la coda* 201
reprise et accord 535
reprise et anacrouse 195
reprise et *Coda* 200
reprise et contrôle de barre de mesure 195
reprise et glissando 184, 209
reprise et liaison 209
reprise et métrique 87
reprise et numéro de mesure 207
reprise et paroles 377
reprise et *Segno* 200
reprise manuelle 210
reprise sous-jascente, barre de mesure 134
reprise, barre de mesure 134
reprise, barre en début de pièce 194
reprise, crochet raccourci 206
reprise, D.C. 202
reprise, D.C., manuel 213
reprise, D.S. 202
reprise, D.S., forçage au début 204
reprise, D.S., manuel 213
reprise, da capo 202
reprise, da capo, manuel 213
reprise, dal segno 202
reprise, dal segno, forçage au début 204
reprise, dal segno, manuel 213
reprise, expansion 193
reprise, fin alternative 195
reprise, pourcent, compteur 215
reprise, script 945
reprise, segno 204
reprises imbriquées 209
reprises successives, barre de mesure 135
reset-footnotes-on-new-page 703, 987
\resetRelativeOctave 7, 973
\resetTagGroup 652, 973
\resetTagGroups 652, 973
respiration, indication 179
respiration, liste des indicateurs 946
respiration, modification du symbole 179
\responsum 574, 973
\rest 69, 886
rest, within text, by duration 886
rest, within text, by log and dot-count 887
\rest-by-number 887
rest-event 276
restNumberThreshold 284
restrainOpenStrings 447
retouche (*tweak*) 791
retouche de note d'ornement 152
retour au propriétés par défaut de la métrique 85
\retrograde 17, 973
reverb MIDI 682
\reverseturn 158, 944
\revert 789
\revertTimeSignatureSettings 85, 973
\rfz 162
rgb-color 299
RGB, couleur 299
\rheel 436
\rheeltoe 436
\rhythm 761, 762, 763, 887
rhythm, in text 887
RhythmicStaff 249, 761
ride bell 946
ride cymbal 946
\right-align 337, 863
right-aligning text 863
\right-brace 910
\right-column 864
right-margin 698, 987
RIGHT 822, 823
\rightHandFinger 493, 973
\romanStringNumbers 441, 445
\rotate 864
rotating text 864
\rounded-box 342, 877
\rtoe 436
\rtoeheel 436

RVB, couleur 299
rythmique d'une mélodie 101

S

s (silence invisible) 71
sélection de la taille (notation) 286
séparateur d'accord 535
\sacredHarpHeads 48
\sacredHarpHeadsMinor 49
saisie, ignorer des passages 667
\sans 845
SATB 401
saut 180
saut de durée 71
saut de ligne 123
saut de ligne et cadences 95
saut de ligne et ligature 105
saut de ligne et musique non mesurée 95
saut de ligne manuel 711
saut de ligne régulier 713
saut de page 745
saut de page et cadences 95
saut de page et musique non mesurée 95
saut, gestion sur voix dédiée 714
scalable vector graphics 668
\scale 878
\scaleDurations 63, 96, 973
scaling markup 878
scaling text 865
Scheme, objet 958
Scheme, variable 957
\score 608, 612, 888
Score 760
Score, remplacement du contexte 779
\score-lines 915
score-markup-spacing 696, 987
score-system-spacing 697, 987
score-title 987
score-title-properties 987
Score_engraver 779
Score_performer 779
scoreTitleMarkup 624, 987
Scottish highland bagpipe 512
script 943
script et silence multimesure 74
script textuel, alignement vertical 169
script-priority 161, 953
scripts, ordre vertical 161
seconda volta 195
seconde fois 194
\section 134, 150, 581
section, barre de mesure 134, 150
section, marque 323
\sectionLabel 201, 323, 973
segno 125, 127, 202
\segno 158, 889, 945
Segno et reprise 200
segno, barre de mesure 134
segno, forçage au début 204, 212
segno, signe 945
segno, signe, barre de mesure 204
segno, signe, manuel 212
segno, signe, reprises 204
\segnoMark 134, 204, 212, 973
self-alignment-interface 822
self-alignment-X 719, 823
self-alignment-Y 824
semi-transparence et couleurs 298
\semicirculus 572, 945
semicirculus, grégorien, articulation 572
\semiflat 889
\semiGermanChords 531
\semisharp 889
\senzaMisura 973
septième 523
septième majeure, symbole 533
\serif 845
sesqui-bémol 12
sesqui-dièse 12
\sesquiflat 889
\sesquisharp 889
\set 107, 786, 790
set-global-staff-size 707
set-octavation 27
\setDefaultMeasureLength 973
\setMeasureLengthFromHere 974
setting extent of text object 913
setting horizontal text alignment 855
setting subscript, in standard font size 841
setting superscript, in standard font size 841
\sf 162
\sff 162
\sfz 162
\shape 830, 974
\sharp 890
\shiftDurations 974
\shiftOff 233
\shiftOn 233
\shiftOnn 233
\shiftOnnn 233
short-indent 272, 699, 987
\shortfermata 158, 944
showFirstLength 667, 957
\showKeySignature 512
showLastLength 667, 957
\showStaffSwitch 432
side-axis 822
side-position-interface 822
sidestick 946
signe « snappizzicato » 158
signe de mensuration 558
signe de respiration, modification 179
signe, « segno » 158
signet 639
signets 641
\signumcongruentiae 158, 945
silence 69
silence ancien 560
silence d'église 284
silence d'espacement 71
silence invisible 71
silence multimesure 69, 73
silence multimesure et doigté 77
silence multimesure et *markup* 76
silence multimesure et point d'orgue 74
silence multimesure, étiquette 74
silence multimesure, ajout de texte 74
silence multimesure, longueur 75

silence multimesure, positionnement 75
silence multimesure, script 74
silence multimesure, style 284
silence, décalage automatique 233
silence, division 100
silence, mesure entière 73
silence, spécification du positionnement vertical ... 69
silences et collisions 77
silences, condenser les 77
silences, fusion 238
silencieuse, note, percussion 507
\simple 846
simple text string, with tie characters 891
simple, barre de mesure 134
simultanée, liaison de phrasé 178
simultanées, liaisons d'articulation 174
\single 633, 791, 792, 795, 974
\skip 71, 974
skipBars 284
skipTypesetting 667
slashChordSeparator 532
slashed digit 910
\slashed-digit 910
\slashedGrace 151, 974
\slashSeparator 703
\slashturn 158, 944
slur-event 276
\slurDashed 174
\slurDashPattern 175, 974
\slurDotted 174
\slurDown 174
\slurHalfDashed 174
\slurHalfSolid 174
\slurNeutral 174
\slurSolid 174
\slurUp 175
\small 289, 336, 846
\smallCaps 846
\smaller 334, 336, 846
smob 958
snap pizzicato 443
\snappizzicato 158, 945
snare 946
sol, clef de 20
Solesmes 553
solfège 307
solo 238
son 672
soprano, clef 939
sori 599
sortie, définition 760, 957
sos. 435
sostenuto, pédale 435
\sostenutoOff 435
\sostenutoOn 435
soufflet 163
soufflet et barre de mesure 167
soufflet Ferneyhough 168
soufflet ouvert 168
soufflet penché 819
soufflet plat 168
soufflet, al niente 168
soufflet, alignement directionnel sur NoteColumn . 167
soufflet, déplacement de l'extrémité 168
soufflet, ligature 121
soufflet, longueur minimale 167
souligné 333
\sourcefileline 614
\sourcefilename 614
sourdine 945
sous-ligature, orientation 113
sous-mesure, barre 134
\southernHarmonyHeads 48
\southernHarmonyHeadsMinor 49
\sp 162
spécification d'un repère 145
space-alist 741
Span_stem_engraver 433
spanner, bandeau 804, 958
spatialisation (*pan*), MIDI 682
splash cymbal 946
\spp 162
Sprechgesang 411
stéréo MIDI, balance 682
\staccatissimo 158, 944
\staccato 158, 944
stacking text, in a column 851
Staff 761
staff brace, in markup 906
staff-affinity 719
staff-height 987
staff-padding 292
\staff-space 803, 987
staff-staff-spacing 719
Staff_collecting_engraver 146, 327
Staff_symbol_engraver 265
StaffGroup 761
staffgroup-staff-spacing 719
StaffGrouper 719
\staffHighlight 300, 974
StandaloneRhythmScore 761
StandaloneRhythmStaff 762
StandaloneRhythmVoice 763
standard, taille de note 290
\stanza 388, 974
stanzaReminders 389
stanzaReminderText 389
start-repeat 210
startAcciaccaturaMusic 153
startAppoggiaturaMusic 153
startGraceMusic 153
\startGradualTempoChange 974
\startGroup 313
\startOptionalMaterial 304
\startStaff 258, 262, 761
\startTextSpan 321, 811, 812
\startTrillSpan 189
\stdBass 898
\stdBassIV 899
\stdBassV 900
\stdBassVI 901
Stem 433
stem-spacing-correction 736
\stemDown 306
stemLeftBeamCount 118, 119
\stemNeutral 306
stemRightBeamCount 118, 119
\stemUp 306
stencil 958
\stencil 911

stencil, suppression ... 812
stopAcciaccaturaMusic ... 153
stopAppoggiaturaMusic ... 153
stopGraceMusic ... 153
\stopGradualTempoChange ... 975
\stopGroup ... 313
\stopOptionalMaterial ... 304
\stopped ... 158, 505, 510, 945
\stopStaff ... 258, 262, 265, 761
\stopStaffHighlight ... 300
\stopTextSpan ... 321, 811, 812
\stopTrillSpan ... 189
\storePredefinedDiagram ... 483, 488, 975
strictBeatBeaming ... 113
\string-lines ... 915
stringNumberOrientations ... 292
\stringTuning ... 466, 975
stringTunings ... 465, 479
strokeFingerOrientations ... 292, 493
\stropha ... 573, 579
\strut ... 911
style d'accidentelle ... 32
style d'altération *modern accidental* ... 35
style d'altération *modern voice* ... 35
style de métrique ... 82
style de repère ... 145
style de trait, liaison ... 174, 178
style de voix ... 232
style, nuance textuelle ... 170
style, silence multimesure ... 284
style, tête de note ... 45
\styledNoteHeads ... 975
\sub ... 335, 846
subbass, clef ... 939
subdivideBeams ... 112
subdivision de ligature ... 112
\submeasureBarsOn ... 134
submeasureStructure ... 84, 134
subscript text ... 846
substituant pour événement ... 222
substitution de doigt ... 290
substitution, fonction ... 659
subsubtitle ... 618
subtitle ... 618
succession de n-olets ... 58
suggestAccidentals ... 161, 561
\super ... 335, 847
superscript text ... 847
suppression, accords répétés ... 487
surimpression d'objet ... 813
surimpression de *grob* ... 66
sus ... 526
suspension de portée ... 258
sustain, pédale ... 435
sustain, style de pédale ... 435
\sustainOff ... 435
\sustainOn ... 435
SVG, format de sortie ... 668
swing ... 887
swing.ly script ... 684
syllabe, durée automatique ... 359
symbole arabe d'un demi-bémol ... 593
symbole d'alignement sur rupture ... 741
symbole de portée ... 258
symbole de septième majeure ... 533
symbole non musical ... 343
symbole, arpège ... 187
synchronisation des notes d'ornement ... 155
syntaxe du mode *markup* ... 329
système ... 250
système choral ... 250
système pianistique ... 251
système, début de ... 250
système, espacement des portées ... 719
système, grand ... 250
système, indicateur de séparation ... 257
système, rectangle en front ... 252
systèmes imbriqués ... 254
system-count ... 700, 988
system-separator-markup ... 703, 988
system-system-spacing ... 697, 988
systems-per-page ... 700, 988

T

ténor, clef ... 21
ténor, varC, clef ... 939
tête de note ... 286
tête de note allongée ... 52
tête de note en losange ... 441
tête de note et improvisation ... 52
tête de note, Aiken ... 48
tête de note, allure ... 48
tête de note, apprentissage ... 46
tête de note, Christian Harmony ... 48
tête de note, Funk ... 48
tête de note, Harmonia Sacra ... 48
tête de note, harpe sacrée ... 48
tête de note, spéciale ... 45
tête de note, style ... 232, 937
tête de note, Walker ... 48
tête profilée, fusion ... 50
\tabChordRepeats ... 448, 975
\tabChordRepetition ... 975
\tabFullNotation ... 447
tablature ... 249, 444
tablature et désinence ... 453
tablature et glissando ... 461, 462
tablature et hampe ... 457
tablature et harmonique ... 458
tablature et indication d'harmonique ... 451
tablature et micro-intervalle ... 467
tablature et polyphonie ... 457
tablature et quart de ton ... 467
tablature par défaut ... 446
tablature pour banjo ... 444, 498
tablature pour guitare ... 444
tablature, accordage prédéfini ... 465
tablature, base ... 446
tablature, clef ... 467, 940
tablature, *hammer on* ... 463
tablature, luth ... 499
tablature, *pull off* ... 463
\table ... 915
table de doigtés ... 510
table des matières, personnalisation ... 642
\table-of-contents ... 643, 916
TabStaff ... 249, 446, 762
TabVoice ... 446, 763

tag ... 647
\tag ... 647, 911, 975
\tag et citation de musique ... 649
\tag-list ... 657, 916
\tagGroup ... 651, 975
\tagGroupRef ... 651, 975
tagline ... 618, 988
taille des notes ... 286
taille et fonte ... 708
talon ... 945
tam tam ... 946
tambourine ... 946
\taor ... 512
taqasim ... 595
teaching ... 38
teaching, style d'altérations ... 38
\teeny ... 289, 336, 847
tempo ... 88
\tempo ... 88
tempo en *markup* ... 92
tempo sous la portée ... 91
tempo, changement masqué ... 92
tempo, with rhythm ... 887
tempoCountPrecision ... 88
\temporary ... 795, 805, 975
temporel, quadrillage ... 311
temporelle, note de bas de page ... 632
temps, gestion du ... 80
tenor G, clef ... 939
tenor, clef ... 939
tenue et arpeggio ... 67
tenue et nuances successives ... 164
tenue, gravure manuelle ... 67
tenue, liaison ... 64
\tenuto ... 158, 944
tessiture ... 41
test de mesure ... 144
text column, left-aligned ... 860
text column, right-aligned ... 864
text, with a tie above ... 844, 847
text, with a tie below ... 847, 848
\text-accidental ... 890
\text-common-time ... 890
\text-cut-time ... 890
\text-doubleflat ... 890
\text-doublesharp ... 891
\text-flat ... 891
text-font-size ... 334, 988
\text-natural ... 891
\text-sharp ... 891
texte (*markup*), taille ... 334
texte ajouté ... 329
texte en colonnes ... 328, 340
texte en préambule ... 328
texte et alternative ... 211
texte et extenseur ... 321
texte et liaison ... 176
texte et rembourrage ... 343
texte hors marges ... 320
texte indépendant ... 328
texte indépendant et note de bas de page ... 635
texte indiquant le nombre de mesures vides ... 74
texte isolé ... 328
texte justifié ... 341
texte multiligne ... 340
texte sur plusieurs lignes ... 341
texte, alignement ... 337
texte, alignement horizontal ... 337
texte, alignement vertical ... 338
texte, autres langues ... 317
texte, centrage sur la page ... 341
texte, codage ... 664
texte, décoration ... 342
texte, en début de ligne ... 323
texte, en fin de partition ... 323
texte, encadrement ... 342
texte, entre des notes ... 323
texte, inclusion dans une liaison ... 176
texte, largeur de ligne ... 341
texte, maintien dans les marges ... 320
texte, mise en forme des extenseurs ... 321
texte, mise en forme des prolongations ... 321
texte, nuance, style ... 170
texte, objet ... 317
texte, octaviation ... 29
texte, sur barre de mesure ... 323
texte, *top-level* ... 328
\textEndMark ... 323, 976
\textLengthOff ... 75, 76, 320
\textLengthOn ... 75, 76, 169, 320
\textMark ... 323, 976
\textSpannerDown ... 321
\textSpannerNeutral ... 321
\textSpannerUp ... 321
\thumb ... 158, 291
\tie ... 847
tie, above text ... 844, 847
tie, below text ... 847, 848
TieColumn ... 67
\tied-lyric ... 891
\tieDashed ... 65
\tieDashPattern ... 65, 976
\tieDotted ... 65
\tieDown ... 65
\tieHalfDashed ... 65
\tieHalfSolid ... 65
\tieNeutral ... 65
\tieSolid ... 65
\tieUp ... 65
tieWaitForNote ... 67
timbale ... 946
\time ... 81, 107, 976
\timeAbbrev ... 976
\times ... 976
timeSignature ... 96
Timing ... 80, 761
timing ... 80
\tiny ... 289, 336, 847
tirer l'archet ... 945
tiret, trait de liaison ... 174, 178
tirettes d'accordéon, symboles ... 437
title ... 618
titre ... 615
tocFormatMarkup ... 643, 988
tocIndentMarkup ... 643, 988
\tocItem ... 643, 976
tocItemMarkup ... 643, 988
\tocItemWithDotsMarkup ... 642
tocTitleMarkup ... 643, 988
\toe ... 158, 437, 945

toeHeelStyle ... 436
tom tom ... 946
top-level, texte ... 328
top-margin ... 694, 988
top-markup-spacing ... 697, 988
top-system-spacing ... 697, 988
toplevel-bookparts ... 957
toplevel-scores ... 957
trémolo ... 217
trémolo interportée ... 218
trémolo, indication de ... 217
trémolo, ligature de ... 217
tracé d'objet graphique ... 342
trait d'union ... 367
transcription de musique ancienne ... 253
transcription, grégorien ... 418, 584
transcription, musique ancienne ... 587
transformation rétrograde ... 17
\translate ... 338, 864
\translate-scaled ... 338, 865
translating text ... 864, 865
transparence, semi ... 298
\transparent ... 911
transparent, objet ... 813
transparente, note ... 297
\transpose ... 6, 14, 17, 976
\transposedCueDuring ... 281, 976
transpositeur, instrument ... 14
transposition ... 14
\transposition ... 30, 274, 977
transposition des hauteurs ... 14
transposition et clef ... 21
transposition et diagramme de fret ... 481
transposition et MIDI ... 30
transposition et octave relative ... 6
transposition, instrument ... 30
tre corde ... 435
treble, clef ... 939
\treCorde ... 435
triade ... 523
triangle ... 946
\triangle ... 343, 878
\trill ... 158, 189, 944
trille ... 189
trille avec hauteur explicite ... 190
trille avec hauteur explicite et altération ... 191
triolet ... 56
triolet, formatage ... 58
triolet, liaison ... 57
\tripletFeel ... 684
tronqué, glissando ... 183
true (#t) ... 989
\tuplet ... 56, 96, 977
tuplet-slur ... 57
\tupletDown ... 56
\tupletNeutral ... 56
TupletNumber ... 58
\tupletSpan ... 58, 977
tupletSpannerDuration ... 58
\tupletUp ... 56
turc, nom de note ... 598
\turn ... 158, 944
turque, exemple de makam ... 598
turque, makam ... 591
turque, musique ... 597
turque, musique classique ... 591
\tweak ... 790, 791, 977
tweak (retouche, affinage) ... 791
tweak et point de contrôle ... 793
\tweak, relation avec \override ... 790
two-sided ... 699, 988
\type ... 776, 777
type de caractère ... 956
\typewriter ... 848

U

U.C. ... 435
ukulele ... 470
ukulele, tablature ... 465
ultima volta ... 195
una corda ... 435
\unaCorda ... 435
\underline ... 333, 848
underlining text ... 848
\undertie ... 848
\undo ... 795, 977
une pause par mesure ... 73
\unfolded ... 197, 977
\unfoldRepeats ... 197, 679, 977
\unHideNotes ... 297
Unicode ... 664
universal-color ... 299, 935
\unless ... 627, 892
\unset ... 786
UP ... 822, 824
\upbow ... 158, 441, 945
\upmordent ... 158, 944
\upprall ... 158, 944
\upright ... 849
URL link, as QR code ... 909
ut, clef d' ... 20
UTF-8 ... 664

V

vérification d'octave ... 13
vérification des limites de mesure ... 144
varC, clef ... 939
\varcoda ... 158, 891, 945
\varheel ... 158, 437, 945
variable ... 613
variable et \book ... 610
variable et \bookpart ... 610
variable prédéfinie, papier ... 692
variables, utilisation de ... 645
variante ... 262
variante rythmique ... 385
Variation Font, format (VF) ... 348
\vartoe ... 158, 437, 945
Vaticana, Editio ... 552, 553
VaticanaLyrics ... 569, 763
VaticanaScore ... 569, 761
VaticanaStaff ... 249, 569, 762
VaticanaVoice ... 569, 762
\vcenter ... 865
vents ... 509
vents, doigtés ... 510
vents, doigtés, diagramme ... 517

\verbatim-file ... 912
\verboseBarNumbers ... 977
\version ... 614
\versus ... 574, 978
vertical, alignement de nuance ... 169
vertical, alignement de script textuel ... 169
vertical, alignement des nuances, rupture ... 170
vertical, espacement ... 719, 746
vertical, ordre des scripts ... 161
vertical, positionnement des nuances ... 166
vertical, positionnement forcé des *grobs* ... 801
VerticalAxisGroup ... 719
verticale, étendue des objets ... 820
verticale, liaison ... 187
vertically centering text ... 865
\verylongfermata ... 158, 944
\veryshortfermata ... 158, 944
vibraslap ... 946
vide, accord ... 153, 164, 428
violin, clef ... 939
\virga ... 573, 579
\virgula ... 572
visibilité d'objets ... 812
visibilité d'une clef transposée ... 819
visibilité des hampes ... 306
visibilité des n-olets ... 57
visibilité, compteur, pourcent ... 215
visibilité, crochet de n-olet ... 59
vocale, partition ... 404
vocalise ... 364
voice ... 34
Voice ... 227, 762
voice, style d'altérations ... 34
\voiceFour ... 227
\voiceFourStyle ... 232
\voiceNeutralStyle ... 232
\voiceOne ... 227
\voiceOneStyle ... 232
\voices ... 231, 978
\voiceThree ... 227
\voiceThreeStyle ... 232
\voiceTwo ... 227
\voiceTwoStyle ... 232
\void ... 686, 978
voix ... 227
voix dédiée aux sauts ... 714
voix entre deux portées ... 432
voix et ambitus ... 42
voix multiples ... 233
voix multiples et altérations ... 35, 37
voix, arpège, enjambement ... 188
voix, \autoBeamOff et \partCombine ... 106
voix, citation ... 277
voix, décalage ... 233
voix, division ... 403
voix, octaviation d'une seule ... 29
voix, polyphonie, additionnelle ... 236
voix, réplication ... 277
voix, style ... 232
volta ... 194, 195
\volta ... 195, 197, 978
volta, barre en début de pièce ... 194
volta, prima ... 195
volta, seconda ... 195
volta, ultima ... 195
\volta-number ... 849
Volta_engraver ... 207
\vowelTransition ... 368
voyelle, transition ... 368
\vshape ... 831, 978
\vspace ... 339, 865

W

Walker shape, tête de note ... 48
\walkerHeads ... 48
\walkerHeadsMinor ... 49
whistle ... 946
\whiteout ... 912
whiteout (blanchiment) ... 813
\whiteTriangleMarkup ... 532
\with ... 770, 773
\with-color ... 298, 913
\with-dimension ... 913
\with-dimension-from ... 913
\with-dimensions ... 913
\with-dimensions-from ... 913
\with-link ... 913
\with-outline ... 913
\with-string-transformer ... 849
\with-true-dimension ... 913
\with-true-dimensions ... 914
\with-url ... 879
\withMusicProperty ... 978
\withRelativeDir ... 978
woodblock ... 946
\woodwind-diagram ... 896
\wordwrap ... 341, 866
\wordwrap-field ... 866
\wordwrap-lines ... 332, 917
\wordwrap-string ... 867

X

x11-color ... 299, 300, 931
X ... 822
X-offset ... 719, 821
X11, couleur ... 299, 300
\xNote ... 45, 978
\xNotesOff ... 45
\xNotesOn ... 45

Y

Y ... 822
Y-offset ... 821