LilyPond

Le système de gravure musicale

Manuel d'initiation

L'équipe de développement de LilyPond

(For LilyPond version 2.11.39)

Table des matières

Préface 1

1 Introduction 2
1.1 Gravure 2
1.2 Gravure automatisée 3
1.3 Gravure des symboles musicaux 5
1.4 Représentation de la musique 6
1.5 Exemples d'application 8
1.6 À propos de ce manuel 9

2 Tutoriel 11
2.1 Premiers pas 11
2.1.1 Compilation d'un fichier 11
2.1.2 Notation simple 12
2.1.3 Travail sur des fichiers texte 16
2.1.4 Bien lire le tutoriel 17
2.2 Notation sur une seule portée 17
2.2.1 Relative note names 17
2.2.2 Altérations et armure 17
2.2.3 Liaisons 19
2.2.4 Articulations et nuances 20
2.2.5 Barres de ligature automatiques et manuelles 21
2.2.6 Commandes rythmiques avancées 22
2.3 Notes simultanées 23
2.3.1 Les expressions musicales en clair 23
2.3.2 Plusieurs portées 25
2.3.3 Double portée 26
2.3.4 Combinaison de notes en accords 27
2.3.5 Polyphonie sur une portée 27
2.4 Chansons 28
2.4.1 Gravure de paroles 28
2.4.2 Partition d'une chanson 30
2.5 Dernières précisions 31
2.5.1 Numéro de version 31
2.5.2 Ajout de titres 31
2.5.3 Noms de note absolus 31
2.5.4 Organisation du code source avec des variables 33
2.5.5 Après le tutoriel 34
2.5.6 Bien lire le manuel 34

3 Concepts fondamentaux 35
3.1 Organisation des fichiers LilyPond 35
3.1.1 Introduction à la structure de fichier LilyPond 35
3.1.2 La partition est une unique expression musicale composée 36
3.1.3 Expressions musicales imbriquées 38
3.1.4 Non-imbrication des crochets et liaisons 38
3.2 Les voix contiennent la musique 38

3.2.1 J'entends des Voix ... 38
3.2.2 Instantiation explicite des voix ... 38
3.2.3 Voix et paroles ... 40
3.3 Contextes et graveurs ... 41
3.3.1 Tout savoir sur les contextes ... 41
3.3.2 Création d'un contexte ... 42
3.3.3 Tout savoir sur les graveurs ... 43
3.3.4 Modification des propriétés de contexte ... 43
3.3.5 Ajout et suppression de graveurs ... 43
3.4 Extension des modèles ... 43
3.4.1 Soprano et violoncelle ... 43
3.4.2 Partition pour chœur à quatre voix mixtes ... 46
3.4.3 Écriture d'une partition à partir de zéro ... 46

4 Retouche des partitions ... 47
4.1 Déplacement d'objets ... 47
4.2 Correction des collisions d'objets ... 50
4.3 Retouches courantes ... 50
4.4 Fichiers fournis avec le logiciel ... 52
4.5 Réduction du nombre de pages de la partition ... 53
4.6 Retouches avancées avec Scheme ... 54
4.7 Options ralentissant le traitement ... 55

5 Travail sur des projets LilyPond ... 56
5.1 Suggestions de saisie des fichiers LilyPond ... 56
5.1.1 Suggestions générales ... 56
5.1.2 Gravure de musique existante ... 57
5.1.3 Projets d'envergure ... 57
5.2 Économies de saisie grâce aux identificateurs et fonctions ... 57
5.3 Feuilles de style ... 59
5.4 Mise à jour d'anciens fichiers ... 63
5.5 Résolution de problèmes — tout remettre à plat ... 63
5.6 Exemples minimaux ... 64

Annexe A Modèles ... 65
A.1 Portée unique ... 65
A.1.1 Notes seules ... 65
A.1.2 Notes et paroles ... 65
A.1.3 Notes et accords ... 65
A.1.4 Notes, paroles et accords ... 65
A.2 Modèles pour claviers ... 65
A.2.1 Piano seul ... 65
A.2.2 Chant et accompagnement ... 65
A.2.3 Piano et paroles entre les portées ... 65
A.2.4 Piano et nuances entre les portées ... 65
A.3 Quatuor à cordes ... 65
A.3.1 Quatuor à cordes ... 65
A.3.2 Parties pour quatuor à cordes ... 65
A.4 Ensemble vocal ... 65
A.4.1 Partition pour chœur à quatre voix mixtes ... 65
A.4.2 Partition pour chœur SATB avec réduction pour piano ... 65
A.4.3 Partition pour chœur SATB avec alignement des contextes ... 65
A.5 Exemples de notation ancienne ... 65

A.5.1 Transcription de musique mensurale 65
A.5.2 Transcription du grégorien 65
A.6 Symboles de jazz 65
A.7 Squelettes pour lilypond-book 66
A.7.1 LaTeX 66
A.7.2 Texinfo 66

Annexe B Tutoriel Scheme 67

Annexe C Licence GNU de documentation libre 68
C.0.1 SUPPLÉMENT : comment utiliser cette licence pour vos documents 73

Annexe D Index de LilyPond 74

Préface

Ce doit être pendant une répétition de l'Orchestre des Jeunes d'Eindhoven, un jour de 1995, que Jan, du pupitre des altistes tordus, aborda Han-Wen, un corniste déjanté, pour lui parler d'un mirifique projet dans lequel il venait de se lancer. Il s'agissait d'un système automatisé de gravure musicale — le préprocesseur MPP pour MusiXTeX pour être précis. Han-Wen, qui voulait justement imprimer certaines parties d'une oeuvre, jeta un premier coup d'oeil à ce programme, et devint très vite accro. Le MPP s'avéra bientôt une voie sans issue. Après avoir ratiociné et échangé un grand nombre de courriels enflammés, Han-Wen lança le projet LilyPond en 1996, dans lequel, cette fois, ce fut au tour de Jan de se sentir entraîné.

De bien des points de vue, coder un programme informatique, c'est comme apprendre à jouer d'un instrument. Au début c'est sympa, on découvre comment ça fonctionne, et on aborde tout ce qu'on n'arrive pas encore à faire comme autant de défis. L'exaltation de la nouveauté s'estompant, on doit s'entraîner encore et encore. Les gammes, les études deviennent vite ennuyeuses, et peuvent, si l'on n'est pas encouragé par d'autres — professeurs, chefs ou public — en décourager plus d'un. Pourtant, pour peu que l'on persévère, l'instrument devient progressivement une partie de notre vie. Si certains jours en jouer semble naturel, c'est un vrai bonheur. Et si d'autres jours on ne peut tout simplement rien en tirer, on continue quand même à travailler, jour après jour.

De même, développer LilyPond peut être une tâche harassante. Certains jours, c'est un monceau de bugs duquel il faut se dépêtrer. Pourtant, il fait maintenant partie de notre vie, et nous nous accrochons. Notre principale motivation est sans doute que notre logiciel est véritablement utile aux gens. En flânant sur Internet, nous trouvons beaucoup de gens qui se servent de LilyPond, et réalisent des partitions très impressionnantes : c'est incroyable, mais en même temps très flatteur.

Les utilisateurs ne se contentent pas de nous encourager en utilisant notre logiciel ; nombre d'entre eux nous aident aussi en faisant des suggestions et en signalant des bogues. Aussi, nous voudrions remercier ici tous les utilisateurs qui nous ont signalé des bugs, ont fait des suggestions ou ont contribué d'une façon ou d'une autre au développement de LilyPond.

Jouer de la musique ou en graver, il y a là plus qu'une comparaison séduisante. Même si l'on s'amuse beaucoup en programmant tous ensemble, et qu'on éprouve une satisfaction profonde à aider les gens, au bout du compte, notre travail sur LilyPond est avant tout une manière d'exprimer notre amour sincère de la musique. Puisse-t-il vous aider à créer de nombreuses et belles oeuvres !

Han-Wen et Jan

Utrecht/Eindhoven, Pays-Bas, juillet 2002.

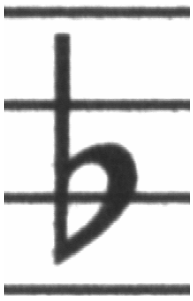
1 Introduction

1.1 Gravure

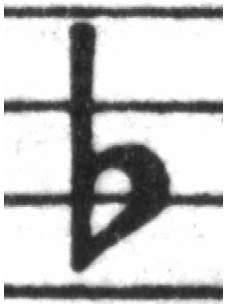
L'art de la typographie musicale se nomme la *gravure*. Ce terme est issu du processus traditionnel d'impression musicale. Il y a seulement quelques dizaines d'années, on faisait les partitions en coupant et en embossant une plaque de zinc ou d'étain en image miroir. Cette plaque était ensuite encrée, les dépressions créées par les creux et les bosses retenant l'encre. Une image était formée en pressant du papier sur la plaque. La découpe et l'embossage étaient entièrement faits à la main. Il était pénible d'appliquer une correction, quand celle-ci n'était pas impossible, la gravure devait donc être parfaite du premier coup. La gravure demandait une qualification hautement spécialisée : un artisan devait accomplir environ cinq ans de formation avant de mériter le titre de maître graveur, et il lui fallait cinq années d'expérience supplémentaires pour devenir vraiment habile.

De nos jours, toutes les partitions récentes sont produites avec des ordinateurs. Ceci a des avantages évidents : le coût des impressions a diminué, et le travail d'éditeur peut être envoyé par courriel. Malheureusement, l'utilisation dominante des ordinateurs a également diminué la qualité graphique des partitions. L'impression informatisée leur donne un aspect fade et mécanique qui les rend désagréables à jouer.

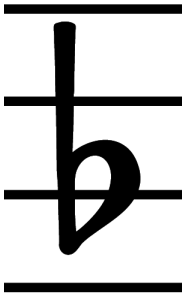
Les images ci-dessous illustrent la différence entre la gravure traditionelle et l'impression typique par ordinateur, et la troisème image montre comment LilyPond mime l'aspect traditionnel. L'image de gauche est une numérisation d'un symbole bémol d'une édition publiée en 2000. Celle du centre montre un bémol d'une gravure à la main de l'édition Bärenreiter de la même musique. L'image de gauche illustre des défauts typiques de l'impression informatique : les lignes de portée sont minces, l'épaisseur de trait du bémol est la même que les lignes fines, et il y a un aspect rigide avec des angles pointus. Par contraste, le bémol Bärenreiter possède un aspect gras et arrondi, presque voluptueux. Notre symbole bémol est créé, entre autres, à partir de celui-là. Il est arrondi, et son épaisseur de trait s'harmonise avec nos lignes de portée, lesquelles sont également plus épaisses que celles de l'édition informatique.



Henle (2000)

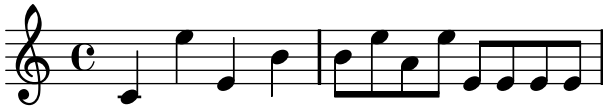


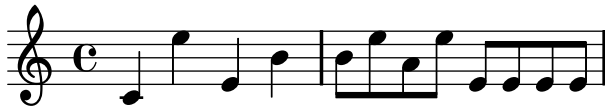
Bärenreiter (1950)



Fonte Feta de LilyPond (2003)

En matière d'espacement, la répartition de l'espace devrait refléter les durées entre les notes. Cependant, beaucoup de partitions modernes se contentent des durées avec une précision mathématique, ce qui mène à de mauvais résultats. Dans l'exemple suivant, un motif est imprimé deux fois : une fois en utilisant un espacement mathématique exact, et une autre fois avec des corrections. Pouvez-vous les repérer ?





L'extrait n'utilise que des notes de même durée ; l'espacement devrait le refléter. Malheureusement, notre oeil nous trompe quelque peu ; il ne se contente pas de remarquer la distance entre les têtes de notes, il prend en compte également la distance entre les hampes consécutives. Ainsi, par compensation, les notes avec une combinaison « hampe vers le haut »/« hampe vers le bas » doivent être éloignées l'une de l'autre, et les notes avec une combinaison « hampe vers le bas »/ « hampe vers le haut » rapprochées, le tout dépendant de la position verticale des notes. Les deux premières mesures sont imprimées avec cette correction, les deux suivantes sans. Les notes dans les deux dernières mesures forment des blocs de notes « hampe vers le bas »/« hampe vers le haut ».

Les musiciens sont généralement plus absorbés par l'exécution que par l'étude de l'aspect graphique d'une partition, donc discutailler sur les détails typographiques peut paraître peu important. Il n'en est rien. Dans de longues pièces avec des rythmes monotones, les corrections d'espacement engendrent de subtiles variations dans la mise en forme de chaque ligne, donnant à chacune une signature visuelle distincte. Sans cette signature, toutes les lignes auraient le même aspect, et ressembleraient à un labyrinthe. Si un musicien regarde ailleurs un instant ou se déconcentre momentanement, il peut avoir du mal à se retrouver sur la page.

De même, l'aspect robuste des symboles sur d'épaisses lignes de portée ressort mieux quand la partition est éloignée du lecteur, comme sur un pupitre par exemple. Une organisation minutieuse des espaces vides permet de minimiser l'espace qu'occupe la musique, tout en évitant que les symboles s'amassent les uns contre les autres. Le résultat permet de réduire le nombre de pages à tourner, ce qui est un grand avantage.

Ceci est une caractéristique commune à toute typographie. La disposition doit être belle, non seulement pour des raisons esthétiques, mais également pour l'aide apportée au lecteur dans la tâche qu'il doit accomplir. Pour du matériel d'exécution comme les partitions de musique, cela prend une double importance : les musiciens ont une quantité limitée d'attention. Moins ils en ont besoin pour lire, plus ils peuvent se concentrer sur la musique elle-même. Autrement dit, une meilleure typographie permet une meilleure interprétation.

Ces exemples démontrent que la typographie musicale est un art subtil et complexe, et que la produire demande une expertise considérable, que les musiciens n'ont généralement pas. LilyPond représente notre effort pour apporter l'excellence graphique de la gravure à la main à l'ère de l'ordinateur, et la rendre accessible à tous les musiciens. Nous avons conçu nos algorithmes, fontes et paramètres de programme pour retrouver la qualité d'édition des anciennes partitions que nous aimons tant lire et jouer.

1.2 Gravure automatisée

Comment pouvons-nous implémenter la typographie ? Si les artisans ont besoin de plus de dix ans pour devenir de vrais maîtres, comment nous, simples programmeurs, pourrions-nous jamais écrire un programme pour faire leur travail ?

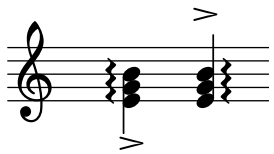
La réponse est : nous ne le pouvons pas. La typographie se base sur le jugement visuel humain, donc les humains ne peuvent pas être complètement remplacés. Si LilyPond arrive à résoudre la plupart des situations correctement, ce sera déjà une grande avancée sur les logiciels existants. Les autres situations peuvent être résolues à la main. Au fil des ans, le logiciel peut être affiné pour faire de plus en plus de choses automatiquement, pour que les ajustements manuels soient de moins en moins nécessaires.

Quand nous avons commencé, nous avons écrit le programme Lilypond entièrement dans le language de programmation C++ ; les fonctions du programme étaient figées par les développeurs. Ceci s'est avéré insatisfaisant pour plusieurs raisons :

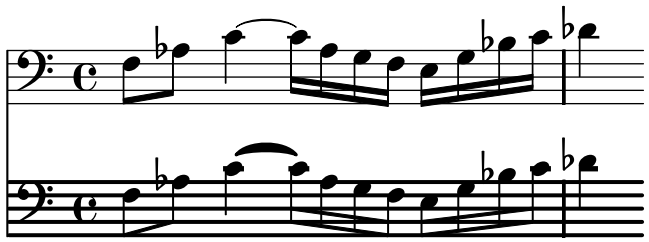
- Quand Lilypond fait des erreurs, les utilisateurs ont besoin de contredire les décisions de formatage. Les utilisateurs doivent donc avoir accès au moteur de formatage. Par conséquent, les règles et les propriétés ne peuvent pas être fixées par nous au moment de la compilation, mais doivent être accessibles aux utilisateurs au moment de l'exécution.
- La gravure est une question de jugement visuel, et donc de goût. Aussi bien informés que nous le sommes, les utilisateurs peuvent être en désaccord avec nos décisions personnelles. Par conséquent, les définitions du modèle typographique doivent également être accessibles à l'utilisateur.
- Enfin, nous affinons continuellement les algorithmes de formatage, donc nous avons besoin d'une approche souple des règles. Le language C++ oblige à une certaine méthode de groupage des règles qui ne convient pas bien au fonctionnement de la notation musicale.

Ces problèmes ont été résolus en intégrant un interpréteur pour le language de programmation Scheme, et en réécrivant des parties de LilyPond en Scheme. L'architecture actuelle de formatage est construite autour de la notion d'objets graphiques, décrits par des fonctions et des variables Scheme. Cette architecture comprend les règles de formatage, le style typographique, et des décisions individuelles de formatage. L'utilisateur a un accès direct à la plupart de ces contrôles.

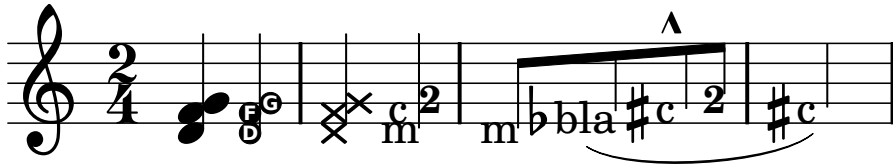
Les variables Scheme contrôlent les décisions de mise en page. Par exemple, beaucoup d'objets graphiques ont une variable de direction qui encode le choix entre haut et bas (ou gauche et droite). Vous pouvez voir ici deux accords, avec des accents, et des arpèges. Dans le premier accord, les objets graphiques sont tous dirigés vers le bas (ou la gauche). Dans le second accord ils sont tous dirigés vers le haut (droite).



Le processus de formatage d'une partition consiste à lire et écrire les variables d'objets graphiques. Certaines variables ont une valeur prédéfinie. Par exemple, l'épaisseur d'un grand nombre de lignes – une caractéristique du style typographique – est une variable avec une valeur prédéfinie. Vous êtes libres d'altérer cette valeur, ce qui vous donne une partition avec une impression typographique différente.



Les règles de formatage ont aussi des variables prédéfinies : chaque objet possède des variables contenant des procédures. Ces procédures exécutent le formatage, et en les substituant par d'autres, nous pouvons changer l'apparence des objets. Dans l'exemple suivant, la règle du choix de têtes de notes est changée au cours de l'extrait de musique.



1.3 Gravure des symboles musicaux

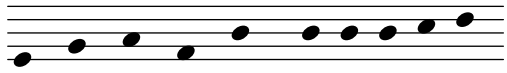
Le processus de formatage décide où placer les symboles. Cependant, cela ne peut être fait qu'à partir du moment où il a été décidé *quels* symboles doivent être imprimés, c'est-à-dire quelle notation utiliser.

La notation musicale usuelle est un système d'écriture qui a évolué à travers les dix derniers siècles. La forme qui est aujourd'hui communément utilisée date du début de la Renaissance. Bien que la forme basique — les têtes de notes sur une portée de cinq lignes — n'a pas changé, les détails continuent d'évoluer pour exprimer les innovations de la notation contemporaine. Par conséquent, elle comprend quelque 500 ans de musique, avec des applications allant des mélodies monodiques à de monstrueux contrepoints pour grand orchestre.

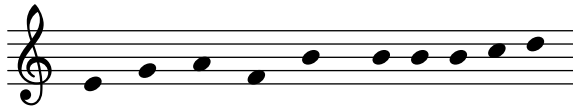
Comment pouvons nous appréhender un tel monstre à plusieurs têtes, et le confiner dans l'espace réduit d'un programme informatique ? Notre solution consiste à diviser le problème de la notation — par opposition à la gravure, ou typographie — en morceaux digestes et programmables : chaque type de symbole est géré par un module séparé, couramment appelé greffon[1]. Chaque greffon est entièrement modulaire et indépendant, et donc peut être développé et amélioré séparément. De tels greffons sont nommés `graveurs`[2], par analogie avec les artisans qui traduisent les idées musicales en symboles graphiques.

Dans l'exemple suivant, voyons comment nous commençons avec un greffon pour les têtes de notes, le graveur de têtes de note (`Note_heads_engraver`) :

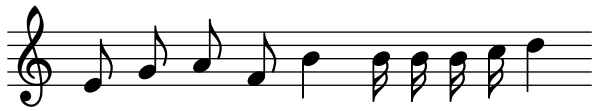
Ensuite, le graveur du symbole de portée (`Staff_symbol_engraver`) ajoute la portée



le graveur de clef (`Clef_engraver`) définit un point de référence pour la portée



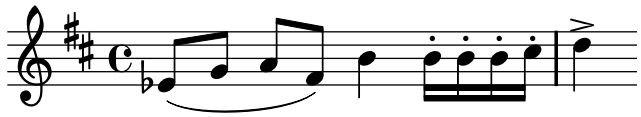
et le graveur de hampes (`Stem_engraver`) ajoute les hampes :



[1] traduction de l'anglais *plug-in*.

[2] `engraver`s en anglais.

Le graveur de hampe est notifié de chaque tête de note qui survient. Chaque fois qu'une tête de note — plusieurs pour un accord — est rencontrée, un objet hampe est créé et connecté à la tête de note. En ajoutant des graveurs pour les barres de ligature, les liaisons, les accents, les altérations accidentelles, les barres de mesure, la métrique, et les armures, nous obtenons un jeu de notation complet.

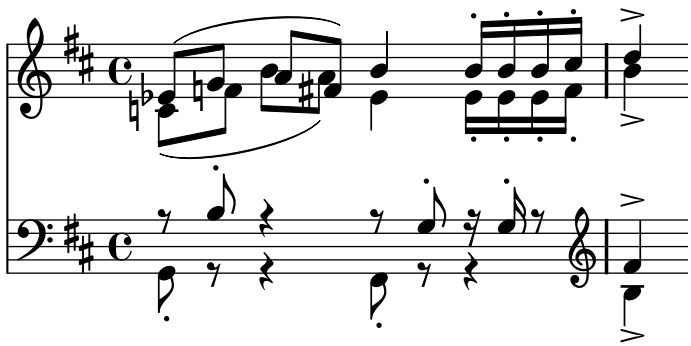


Ce système fonctionne bien pour de la musique monodique, mais qu'en est-il de la polyphonie ? En notation polyphonique, plusieurs voix peuvent partager une portée.

Dans cette situation, la portée et les altérations accidentelles sont partagées, mais les hampes, liaisons etc., sont spécifiques à chaque voix. Par conséquent, les graveurs doivent être groupés. Les graveurs des têtes de notes, hampes, liaisons etc., vont dans un groupe appelé « contexte de Voix »[3], alors que les graveurs des clés, altérations accidentelles, barres de mesure etc., vont dans un groupe appelé « contexte de Portée ». Dans le cas de la polyphonie, un seul contexte de Portée contient plusieurs contextes de Voix. De même, plusieurs contextes de Portée peuvent être inclus dans un seul contexte de Partition. Le contexte de Partition est le contexte de notation de plus haut niveau.

Voir aussi

Program reference: `Contexts`.



1.4 Représentation de la musique

Idéalement, le format d'entrée pour n'importe quel système de formatage est une description abstraite du contenu. Dans ce cas-ci, ce serait la musique elle-même. Cela pose un formidable problème : comment pouvons-nous définir ce que la musique est réellement ? Plutôt que d'essayer

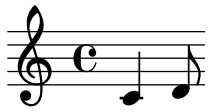
[3] 'Voice context' en anglais, 'Voice' commence par une majuscule comme tous les noms de contexte dans le programme LilyPond.

de trouver une réponse, nous avons renversé la question. Nous écrivons un logiciel capable de produire de la musique écrite, et adaptons le format pour atteindre la plus grande concision possible. Quand le format ne peut plus être simplifié, il nous reste par définition le contenu lui-même. Notre logiciel sert de définition formelle d'un document de musique.

La syntaxe est également l'interface utilisateur pour LilyPond, par conséquent il est facile de saisir

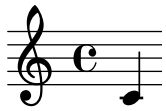
```
c'4 d'8
```

c'est-à-dire un do central noire et, juste au-dessus un ré croche



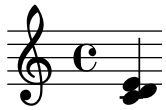
Sur une échelle microscopique, une telle syntaxe est facile à utiliser. A plus grande échelle, la syntaxe a besoin aussi de structure. Comment serait-il possible autrement de rentrer des pièces complexes comme des symphonies ou des opéras ? La structure est formée par le concept d'expression musicale : en combinant de petits fragments de musique pour en former de plus grands, on peut exprimer de la musique plus complexe. Par exemple

```
c4
```



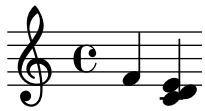
Des accord peuvent être construits avec `<<` et `>>` autour des notes.

```
<<c4 d4 e4>>
```



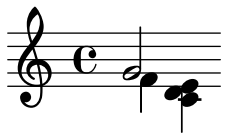
Cette expression est mise dans une séquence grace à l'encadrement par des accolades `{ ... }`

```
{ f4 <<c4 d4 e4>> }
```



Ceci est également une expression, et peut donc encore une fois être combinée avec d'autres expressions simultanées (une blanche) en utilisant `<<`, `\\`, et `>>`

```
<< g2 \\ { f4 <<c4 d4 e4>> } >>
```

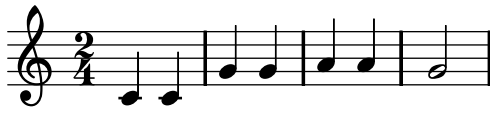


De telles strucutres récursives peuvent être spécifiées formellement et de manière ordonnée dans une grammaire indépendante de tout contexte. Le code d'analyse est aussi générée à partir de cette grammaire. Autrement dit, la syntaxe de LilyPond est définie clairement et sans ambiguité.

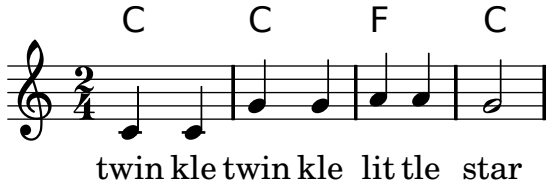
L'interface utilisateur et la syntaxe sont ce que les gens voient et manipulent le plus. Elles sont en partie une affaire de goût, et aussi sujettes à beaucoup de discussions. Même si ces discussions sur les goûts ont leur mérite, elles ne sont pas très productives. D'un point de vue plus large sur LilyPond, l'importance de la syntaxe est minime : il est facile d'inventer une syntaxe concise, alors qu'écrire un code de formatage décent est beaucoup plus difficile. Ceci est également illustré par le nombre de lignes de codes pour les composants respectifs : l'analyse et la représentation constituent moins de 10% du code source.

1.5 Exemples d'application

Nous avons conçu LilyPond comme une expérimentation visant à concentrer l'art de la gravure musicale dans un logiciel. Grâce à tout ce dur labeur, le programme peut maintenant être utilisé pour accomplir des travaux utiles. L'application la plus simple est d'imprimer des notes :



En ajoutant des noms d'accords et des paroles, nous obtenons une partition de chanson :

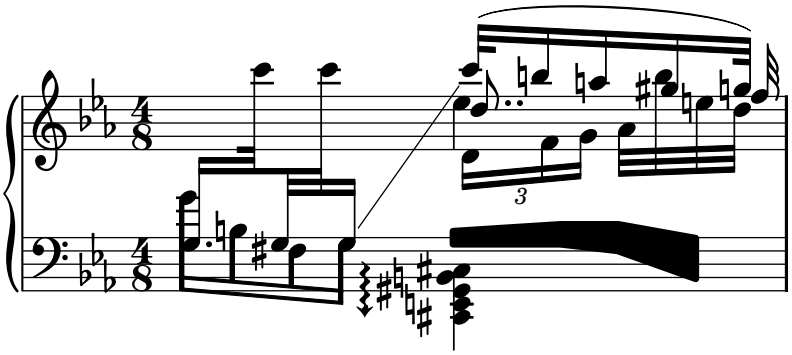


La notation polyphonique et la musique pour piano peuvent également être générées. L'exemple suivant associe quelques constructions plus exotiques :

Screech and boink

Random complex notation

Han-Wen Nienhuys



Les extraits exposés ici ont tous été écrits à la main, mais ce n'est pas une obligation. Puisque le moteur de formatage est en grande partie automatique, il peut servir de sortie pour d'autres programmes qui manipulent la musique. Par exemple, il peut être utilisé pour convertir des bases de données d'extraits musicaux en images pour des sites Internet et des présentations multimédias.

Ce manuel montre également une application : le format d'entrée est du texte, et peut donc facilement être intégré dans d'autres formats basés sur le texte comme LaTeX, HTML, ou dans le cas de ce manuel, Texinfo. À l'aide d'un programme spécial, les extraits de code peuvent être remplacés par des images de musiques dans les fichiers de sortie PDF ou HTML. Cela donne la possibilité de mélanger de la musique et du texte dans les documents.

1.6 À propos de ce manuel

Deux manuels traitent de LilyPond : le *manuel de l'utilisateur* — que vous lisez actuellement — et le *manuel d'utilisation du programme.*

Manuel de l'utilisateur

Ce manuel se divise en trois livres.

Manuel d'apprentissage

Ce livre explique comment débuter avec LilyPond, et expose de manière simple quelques concepts clés. Il est conseillé de lire ces chapitres de manière linéaire.

- *Manuel de l'utilisateur, Tutoriel* propose une introduction en douceur à la typographie musicale. Les utilisateurs débutants sont invités à commencer ici.
- *Manuel de l'utilisateur, Assemblage* explique des concepts généraux du format de fichier ly. Si vous n'êtes pas certain de l'endroit où placer une commande, lisez ce chapitre !
- *Manuel de l'utilisateur, Travail sur des projets LilyPond* montre des utilisations pratiques de LilyPond et donne des conseils afin d'éviter les problèmes les plus courants.
- *Manuel de l'utilisateur, Retouche des partitions* est une introduction aux retouches de gravure avec LilyPond.

Manuel de référence

Ce livre détaille toutes les commandes LilyPond produisant une notation musicale. La lecture de cet ouvrage requiert une bonne compréhension des concepts exposés dans le manuel d'apprentissage.

- *Manuel de l'utilisateur, Basic notation* traite de sujets groupés par type de notation. Cette section détaille la notation de base, qui sera utile dans la plupart des projets de partition.
- *Manuel de l'utilisateur, Instrument-specific notation* traite de sujets groupés par type de notation. Cette section détaille des notations spéciales qui ne seront utiles que pour des types particuliers d'instruments ou la voix.
- *Manuel de l'utilisateur, Advanced notation* traite de sujets groupés par type de notation. Cette section donne des précisions à propos de notations compliquées et inhabituelles
- *Manuel de l'utilisateur, Modification des réglages prédéfinis* explique comment ajuster finement la mise en page.
- *Manuel de l'utilisateur, Objets non musicaux* traite de sorties non musicales comme les titres, les mouvements multiples, et la sélection des instruments MIDI.
- *Manuel de l'utilisateur, Gestion de l'espace* traite de sujets affectant la sortie globale, comme sélectionner la taille de papier ou spécifier les sauts de page.
- *Manuel de l'utilisateur, Interfaces pour les programmeurs* explique comment créer des fonctions de musique.

Annexes

Ce livre contient des tables de référence pratiques.

Autre documentation

D'autres documents constituent de précieuses sources d'information.

Le glossaire de musique explique les termes musicaux, et inclut leur traduction dans diverses langues. C'est un document séparé, disponible aux formats HTML et PDF. Si vous n'êtes pas familier avec la notation musicale ou la terminologie, il est conseillé de consulter le glossaire, notamment pour les parties non encore traduites de la documentation.

- Les Exemples de code sont une vaste sélection de petits exemples montrant des trucs, astuces et fonctionnalités particulières de LilyPond. La plupart de ces exemples sont également disponibles sur le LilyPond Snippet Repository. Ce site Web possède également le manuel de LilyPond avec une fonctionnalité de recherche.
- La référence du programme est un ensemble de pages HTML étroitement liées entre elles, qui documente les moindres petits détails de chaque classe, objet et fonction de LilyPond. Cette documentation est produite directement à partir des définitions de formatage utilisées.

  Presque toutes les fonctions de formatage utilisées en interne sont directement disponibles pour l'utilisateur. Par exemple, toutes les variables qui contrôlent les épaisseurs, les distances etc., peuvent être modifiées dans les fichiers d'entrée. Il y a un grand nombre d'options de formatage, et elles sont toutes décrites dans ce document. Chaque section du manuel de notation a une section **Voir aussi** qui renvoie à la documentation générée automatiquement. Dans la documentation au format HTML, ces sous-sections ont des liens cliquables.

Lorsque vous serez un utilisateur expérimenté, vous pourrez consulter le manuel comme une référence : il y a un index complet[4], mais le manuel est aussi disponible en une seule grande page, ce qui facilite la recherche avec la fonction de recherche de votre navigateur.

Dans tous les documents HTML qui incluent des fragments musicaux, le code Lilypond utilisé pour produire l'image peut être vu en cliquant l'image.

L'emplacement des fichiers de documentation mentionnés ici peut varier d'un système à l'autre. De temps en temps, ce manuel fait référence aux fichiers d'exemple et d'initialisation. Tout au long de ce manuel, nous donnons les emplacements des fichiers d'entrée relativement au répertoire racine de l'archive source. Par exemple, '`input/test/bla.ly`' peut référer au fichier '`lilypond2.x.y/input/test/bla.ly`'. Dans les paquets binaires pour les plateformes Unix, la documentation et les exemples se trouvent généralement sous '`/usr/share/doc/lilypond/`'. Les fichiers d'initialisation, par exemple '`scm/lily.scm`', ou '`ly/engraver-init.ly`', se trouvent généralement dans le répértoire '`/usr/share/lilypond/`'.

Pour finir, ce manuel et les autres sont disponibles en ligne, à la fois aux formats PDF et HTML, à partir du site Web, accessible à l'adresse `http://www.lilypond.org/`.

[4] Si vous cherchez quelque chose sans le trouver dans la documentation, c'est un bogue. Dans ce cas, merci d'envoyer un rapport de bogue.

2 Tutoriel

Ce tutoriel commence par une introduction au langage musical utilisé par LilyPond, qui vous permettra de faire fonctionner le logiciel pour produire une partition. Après ce premier contact, nous verrons comment créer des partitions utilisant une notation musicale courante.

2.1 Premiers pas

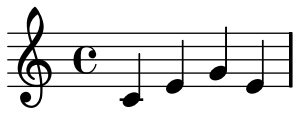
Cette section présente sommairement la façon de travailler avec LilyPond.

2.1.1 Compilation d'un fichier

Le premier exemple montre comment débuter avec LilyPond. Pour créer une partition, on écrit un fichier de texte qui décrit la notation musicale. Par exemple, si l'on écrit

```
{
  c' e' g' e'
}
```

le résultat ressemblera à



Il est aussi possible d'utiliser les noms de notes français '`do re mi fa sol la si`', en insérant au début du fichier la ligne '`\include "italiano.ly"`'

Attention : tout extrait de code LilyPond doit être entouré d'une `{`paire d'accolades`}`. De plus, pour éviter toute ambiguïté, il est préférable d'entourer les accolades par des espaces ou retours à la ligne. Bien que certains exemples de ce manuel ne comportent pas d'accolades, ne les oubliez pas dans vos partitions !

De plus, LilyPond est sensible à la casse. `{ c d e }` est un code valide ; `{ C D E }` produira un message d'erreur.

Saisie de la musique et visualisation de la partition produite

Dans cette section nous expliquerons quelles commandes exécuter et comment voir ou imprimer le résultat de LilyPond.

MacOS X

Si vous double-cliquez sur `LilyPond.app`, un fichier d'exemple s'ouvrira. Sauvegardez-le, par exemple, sous '`test.ly`' sur votre bureau, et traitez-le ensuite avec la commande du menu '`Compile > Typeset File`'. Le fichier PDF résultant sera alors affiché sur votre écran.

Notez que le premier démarrage peut prendre une minute ou deux, car toutes les polices système doivent être d'abord analysées.

À l'avenir, vous aurez certainement recours aux commandes « Nouveau » ou « Ouvrir ». Vous devez enregistrer votre fichier avant de lancer la création de la partition. Si une erreur advient pendant le traitement, vous la trouverez dans la fenêtre « log ».

Windows

Sous Windows, lorsque vous double-cliquez sur l'icône LilyPond qui se trouve sur le Bureau, un fichier d'exemple s'ouvre dans un simple éditeur de texte. Enregistrez-le, par exemple en tant que '`test.ly`' sur votre Bureau, puis double-cliquez sur son icône (qui montre une note de musique)

pour le traiter. Après quelques secondes, vous obtiendrez un fichier '`test.pdf`' sur votre Bureau, fichier que vous pourrez ouvrir pour voir la partition imprimée. Une autre méthode pour lancer le traitement du fichier '`test.ly`' est de le glisser avec votre souris sur l'icône de LilyPond.

Pour modifier un fichier '`.ly`' existant, faites un clic droit dessus et sélectionnez « Éditer la source ». Pour partir d'un fichier vide, lancez l'éditeur en ouvrant un fichier existant et utilisez la commande « New » du menu « File ».

En double-cliquant sur le fichier, vous obtiendrez, en plus du fichier PDF, un fichier '`.log`' qui récapitule les opérations que LilyPond a effectuées sur votre fichier. Si une erreur survient, c'est ce fichier qu'il vous faudra étudier.

Notez qu'il existe d'autres éditeurs de texte, certains disposant d'un meilleur support pour LilyPond ; reportez-vous à *Manuel d'utilisation du programme, LilyPond et les éditeurs de texte*.

Unix

Commencez par ouvrir une fenêtre de terminal et un éditeur de texte. Par exemple, vous pouvez ouvrir un xterm et exécuter `joe`.[1] Dans votre éditeur, entrez le texte suivant et sauvegardez le fichier sous '`test.ly`'

```
{
  c' e' g' e'
}
```

Pour traiter '`test.ly`', procédez comme ceci :

```
lilypond test.ly
```

Vous verrez quelque chose ressemblant à :

```
lilypond test.ly
GNU LilyPond 2.10.0
Processing `test.ly'
Parsing...
Interpreting music...  [1]
Preprocessing graphical objects...
Calculating line breaks...  [2]
Layout output to `test.ps'...
Converting to `test.pdf'...
```

Suivant votre installation, ces messages peuvent être traduits.

De tout cela résulte un fichier '`test.pdf`', que vous pouvez imprimer ou visualiser avec les outils standards de votre système d'exploitation.[2]

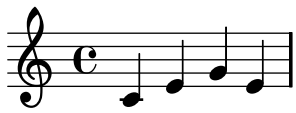
2.1.2 Notation simple

Il y a certains éléments graphiques que LilyPond ajoute automatiquement. Dans l'exemple suivant, nous n'avons fourni que quatre hauteurs, mais LilyPond a ajouté une clé, un chiffre de mesure et du rythme.

```
{
  c' e' g' e'
}
```

[1] Il existe des fichiers de macros pour les fans de VIM et un `LilyPond-mode` pour les fans d'Emacs. S'ils ne sont pas encore installés, consultez le fichier '`INSTALL.txt`'. L'outil d'édition le plus facile d'utilisation est de loin '`LilyPondTool`'. Vous trouverez plus d'informations dans *Manuel d'utilisation du programme, LilyPond et les éditeurs de texte*.

[2] Si votre système ne dispose pas des outils nécessaires, vous pouvez essayer Ghostscript, un programme pour afficher et imprimer librement les fichiers PDF et PostScript.

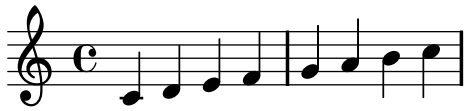


Ce comportement peut être modifié, mais dans bien des cas ces attributions automatiques s'avèrent utiles.

Hauteurs

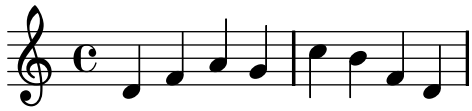
Le moyen le plus simple d'entrer des notes est d'utiliser le mode `\relative`. Avec ce mode, l'`intervalle` entre la note et celle qui la précède est supposé inférieur ou égal à une `quarte`. Commençons par entrer la partition la plus élémentaire qui soit, une `gamme`.

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



La note de départ est `do central`. Chacune des notes qui suivent est à moins d'une quarte de la note précédente — en d'autres termes, le premier '`c`' est le do central, entre la clé de sol et la clé de fa, puis est suivi par le ré le plus proche, et ainsi de suite. On peut bien sûr créer des mélodies d'intervalles plus étendus :

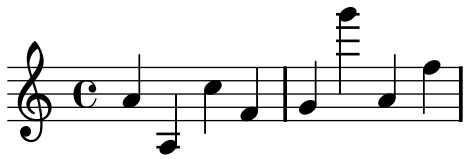
```
\relative c' {
  d f a g
  c b f d
}
```



Comme vous l'aurez remarqué, cet exemple ne commence plus sur le do du milieu. La première note — le '`d`' — est le ré qui en est le plus proche.

Pour ajouter des intervalles supérieurs à une quarte, il suffit d'indiquer si la note est à l'octave supérieure ou inférieure, en ajoutant respectivement une apostrophe `'` ou une virgule `,` au nom de la note.

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

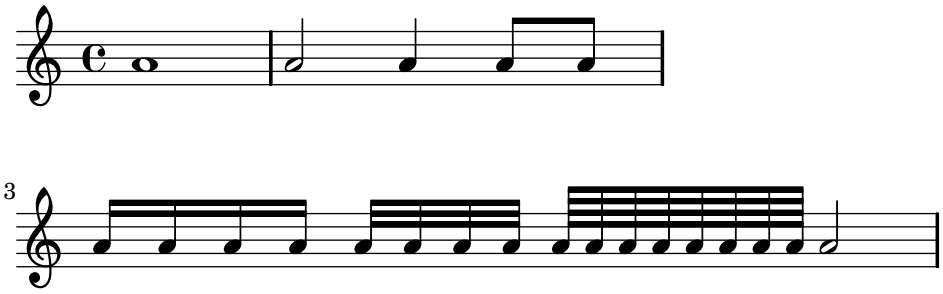


Pour déplacer une note deux octaves (ou davantage !) plus haut ou plus bas, il suffit de mettre plusieurs `''` ou plusieurs `,,` — attention cependant à bien mettre deux apostrophes `''`, et non un guillemet `"` ! C'est de cette même manière que l'on peut modifier la valeur de départ de `\relative c'`.

Durées (rythme)

La `durée` d'une note est indiquée par un nombre qui suit son nom : '`1`' pour une `ronde`, '`2`' pour une `blanche`, '`4`' pour une `noire` et ainsi de suite. Les hampes sont ajoutées automatiquement.

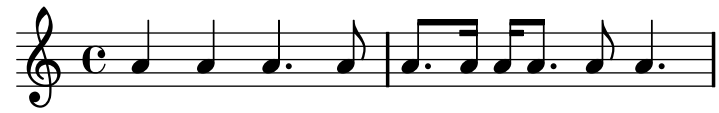
```
\relative c'' {
  a1
  a2 a4 a8 a
  a16 a a a a32 a a a a64 a a a a a a a a2
}
```



Si aucune durée n'est indiquée, la dernière durée entrée sera utilisée pour les notes suivantes. En l'absence d'indication, la première note est une noire.

Une `note pointée` s'obtient en ajoutant un point '`.`' à la valeur rythmique.

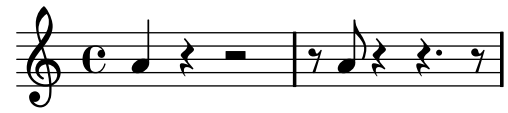
```
\relative c'' {
  a a a4. a8
  a8. a16 a a8. a8 a4.
}
```



Silences

On saisit un `silence` tout comme une note, mais avec le caractère '`r`'.

```
\relative c'' {
  a r r2
  r8 a r4 r4. r8
}
```

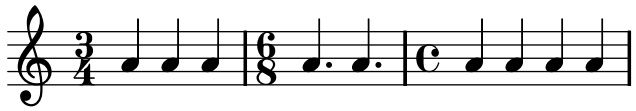


Métrique

La `métrique` peut être définie à l'aide de la commande `\time` :

```
\relative c'' {
  \time 3/4
  a4 a a
  \time 6/8
  a4. a
```

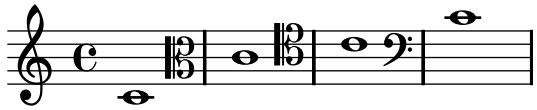
```
  \time 4/4
  a4 a a a
}
```



Clefs

La `clef` peut être définie à l'aide de la commande `\clef` :

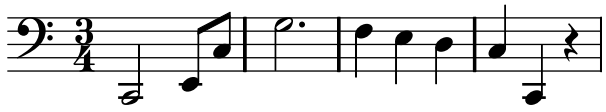
```
\relative c' {
  \clef treble
  c1
  \clef alto
  c1
  \clef tenor
  c1
  \clef bass
  c1
}
```



Assemblage

Voici un bref exemple qui montre tous ces éléments ensemble :

```
\relative c, {
  \time 3/4
  \clef bass
  c2 e8 c' g'2.
  f4 e d c4 c, r4
}
```



Voir aussi

Entrer des hauteurs et des durées
: voir *Manuel de l'utilisateur, Hauteurs* and *Manuel de l'utilisateur, Durées*.

Les silences
: voir *Manuel de l'utilisateur, Silences*.

Les chiffres de mesure et autres commandes de métrique
: voir *Manuel de l'utilisateur, Métrique*.

Les clés
: voir *Manuel de l'utilisateur, Clefs*.

2.1.3 Travail sur des fichiers texte

Le traitement des fichiers source de LilyPond est semblable à celui du code de nombreux langages de programmation répandus : la casse est prise en compte, et les caractères considérés comme espaces ont généralement peu d'importance. Les expressions sont délimitées par des accolades `{ }`, et les commentaires par `%` ou `%{ ... %}`.

Si cette phrase vous paraît incompréhensible, ne vous en faites pas ! Tous ces termes vont être expliqués :

- **La casse** : LilyPond est sensible à la casse, c'est à dire qu'une lettre capitale n'a pas la même valeur qu'une lettre minuscule. Les notes, par exemple, doivent être entrées en minuscules : `{ c d e }` est un code valide, alors que `{ C D E }` produira un message d'erreur.
- **Les espaces multiples** : LilyPond ne tient pas compte du nombre d'espaces, ou de retours à la ligne. `{ c d e }` a le même sens que `{ c d e }` ou que

```
{
c                       d
 e }
```

  Bien sûr, ce dernier exemple est difficile à lire. Une bonne habitude à prendre est d'indenter les blocs de code avec soit des tabulations soit des doubles espaces :

```
{
  c d e
}
```

- **Expressions musicales** : Tout morceau saisi dans LilyPond doit être placé entre `{` **accolades** `}`. Ces caractères indiquent à LilyPond que ce bloc de texte est une et une seule expression musicale, tout comme les parenthèses '`()`' en mathématiques. Il est préférable, pour éviter toute ambiguïté, d'entourer tous ces crochets d'espaces, à moins qu'ils se trouvent au début ou à la fin d'une ligne.

  Une fonction — `\relative { }` par exemple — compte également comme une seule expression musicale.
- **Les commentaires** : Un commentaire est une indication pour tout lecteur humain d'un fichier de musique ; il est ignoré par l'ordinateur, et n'a donc aucun effet sur la partition imprimée. On distingue deux types de commentaires :
  - la ligne de commentaire, introduite par le symbole '`%`' : tout ce qui suit ce symbole sur cette ligne sera ignoré.
  - le bloc de commentaire, qui peut être de plusieurs lignes voire de toute une section : tout ce qui se trouve entre `%{` et `%}` est ignoré. Les blocs de commentaires ne peuvent s'imbriquer.

  Le fragment suivant met en évidence quelques usages possibles des commentaires :

```
% voici les notes de "ah vous dirai-je maman"
 c4 c g' g a a g2

%{
   Ces lignes et les notes qui suivent
   seront ignorées, car elles se trouvent
   dans un bloc de commentaire.

   g g f f e e d d c2
%}
```

Vous trouverez plus d'astuces pour organiser vos fichiers LilyPond dans *Manuel de l'utilisateur, Suggestions de saisie des fichiers LilyPond*.

2.1.4 Bien lire le tutoriel

Comme nous l'avons vu dans *Manuel de l'utilisateur, Travail sur des fichiers texte*, un code LilyPond doit être encadré par des `{ }` ou bien par `\relative c'' { ... }` afin d'être compris. Cependant, dans la suite de ce manuel, la plupart des exemples ne feront pas apparaître ces signes.

Si vous consultez la documentation au format HTML, et que vous souhaitez voir la source exacte d'un exemple, il vous suffit de cliquer sur l'image. Si vous ne disposez pas de la version HTML, il vous est possible de simplement copier et coller le code affiché, mais **à condition** d'ajouter `\relative c'' { }` de la façon suivante :

```
\relative c'' {
 ...  collez ici votre exemple...
}
```

Pourquoi avoir omis les accolades ? La plupart des exemples de ce manuel peuvent être insérés au milieu d'un morceau de musique plus long. Il n'y a donc aucune raison d'ajouter `\relative c'' { }` à ces exemples — en effet, il n'est pas possible d'insérer un `\relative` à l'intérieur d'un autre `\relative`. Il vous serait donc devenu impossible de copier un bref exemple de la documentation et de le coller dans une pièce de votre cru.

2.2 Notation sur une seule portée

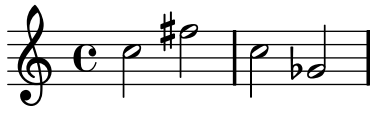
Cette section présente la notation courante dont on a besoin pour écrire une seule voix sur une seule portée.

2.2.1 Relative note names

Comme nous l'avons vu dans *Manuel de l'utilisateur, Notation simple*, LilyPond calcule la hauteur de chaque note en fonction de la précédente[3]. Si aucune indication supplémentaire d'octaviation n'est ajoutée, il en concluera que chaque hauteur est située à une quarte au plus de la note précédente.

Lilypond tient compte des intervalles induits par les noms des notes — en d'autres termes, une quarte augmentée n'est *pas* équivalente à une quinte diminuée. Ainsi, si l'on part d'un Do, un Fa dièse sera placé au-dessus, tandis qu'un Sol bémol sera placé au-dessous.

```
c2 fis
c2 ges
```



Voir aussi

Les hauteurs de note relatives
: voir *Manuel de l'utilisateur, Octaves relatives*.

Les vérifications d'octaves
: voir *Manuel de l'utilisateur, Vérification d'octave*.

2.2.2 Altérations et armure

[3] Il existe un autre mode de saisie des hauteurs, le mode *Manuel de l'utilisateur, Noms de note absolus*, mais en pratique il est bien plus aisé et sûr d'avoir recours au mode de hauteurs relatives.

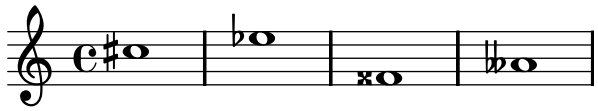
Altérations

Dans la notation par défaut, un `dièse` s'obtient en ajoutant '`is`' au nom de la note, et un `bémol` en ajoutant '`es`'. Comme vous pouvez vous y attendre, un double dièse ou double bémol est alors obtenu en ajoutant '`isis`' ou '`eses`'[4].

Cependant, si vous utilisez la commande '`\include "italiano.ly"`' pour entrer les noms de notes français au lieu des noms hollandais, il faudra ajouter un '`d`' pour un dièse, et un '`b`' pour un bémol. Le double dièse et le double bémol s'obtiennent en ajoutant respectivement '`dd`' et '`bb`'

Pour en savoir plus sur les autres langues disponibles, voir *Manuel de l'utilisateur, Noms de note dans d'autres langues*.

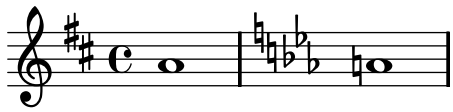
```
cis1 ees fisis, aeses
```



Armures

L'armure est déterminée par la commande `\key`, suivie d'une hauteur puis de `\major` (majeur) ou `\minor` (mineur) :

```
\key d \major
a1
\key c \minor
a
```



Attention aux armures et aux hauteurs

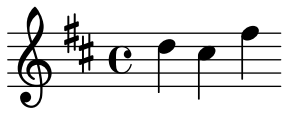
La combinaison de l'armure et des hauteurs de note — y compris les altérations — permet à LilyPond de déterminer dans quel cas afficher les altérations accidentelles. L'armure n'affecte que les altérations *imprimées*, et non les hauteurs réelles ! Cette fonctionnalité est souvent source de confusion pour les nouveaux utilisateurs, aussi expliquons-la en détail.

LilyPond fait une nette distinction entre le contenu musical et la mise en forme. L'altération d'une note — bémol, bécarre ou dièse — fait partie de sa hauteur, et relève donc du contenu musical. La présence ou non d'une altération accidentelle — un *signe* bémol, bécarre ou dièse — devant la note correspondante est une question qui relève de la mise en page. Mettre en page une partition se fait selon des règles ; les altérations accidentelles seront donc automatiquement imprimées suivant ces règles. Les hauteurs de note, en revanche, relèvent de ce que vous voulez entendre ; et, dans la mesure où la musique que vous entrez est censée être celle que vous voulez entendre, LilyPond (qui n'est chargé que de la gravure) ne les choisira pas à votre place.

Dans cet exemple,

```
\key d \major
d cis fis
```

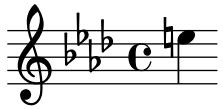
[4] Cette syntaxe est dérivée de la convention de dénomination des notes dans les langues nordiques et germaniques, comme l'allemand ou le hollandais.



aucune note n'a d'altération accidentelle, et pourtant vous devrez entrer le '`is`' pour les notes `cis` et `fis`.

Le code '`d`' ne veut pas dire « Imprimez-moi un point noir juste en-dessous de la portée. » Cela signifie plutôt : « Ici se trouve une note dont la hauteur est un ré naturel. » Avec l'armure de la bémol majeur, ce ré sera flanqué d'un bécarre accidentel :

```
\key aes \major
e
```



Ajouter explicitement toutes les altérations requiert un peu plus d'effort dans la phase de saisie, mais la transposition en sera grandement facilitée. De plus les altérations accidentelles peuvent être imprimées suivant plusieurs conventions. Regardez *Manuel de l'utilisateur, Altérations accidentelles automatiques* pour connaître les différentes manières dont les altérations accidentelles peuvent être imprimées, suivant les règles que vous choisirez.

Voir aussi

Les altérations
: voir *Manuel de l'utilisateur, Altérations* and *Manuel de l'utilisateur, Altérations accidentelles automatiques*.

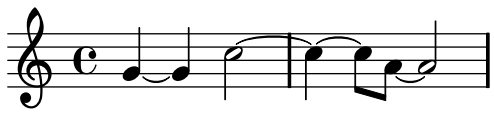
Les armures
: voir *Manuel de l'utilisateur, Armure*.

2.2.3 Liaisons

Liaisons de prolongation

Une `liaison de prolongation`[5] se crée en ajoutant un tilde '`~`' à la première note liée.

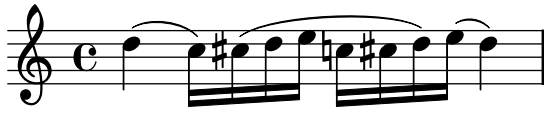
```
g4~ g c2~
c4 ~ c8 a8 ~ a2
```



Liaisons d'articulation

Une `liaison d'articulation` d'articulation (ou « legato ») peut englober plusieurs notes. Les notes de départ et d'arrivée reçoivent respectivement un signe '`(`' et '`)`'.

```
d4( c16) cis( d e c cis d) e( d4)
```

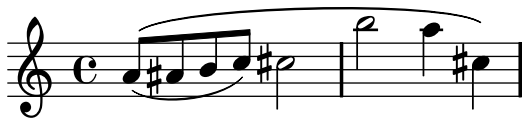


[5] parfois aussi appelée liaison de tenue

Liaisons de phrasé

De plus longues liaisons, dites de phrasé, sont délimitées par `\(` et `\)`. Il est possible d'avoir en même temps des legatos et des phrasés, mais pas plusieurs liaisons de phrasé ou de legato à la fois.

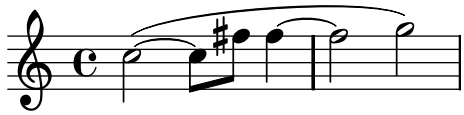
```
a8(\( ais b c) cis2 b'2 a4 cis,\)
```



Attention aux types de liaison

Une liaison d'articulation ou de phrasé ressemble à une liaison de tenue, mais n'a pas la même signification. Alors qu'une liaison de tenue ne peut relier que deux notes de même hauteur, le legato indique une articulation de plusieurs notes, éventuellement nombreuses. Les liaisons de tenue peuvent être enchâssées dans un legato ou un phrasé.

```
c2~( c8 fis fis4 ~ fis2 g2)
```



Voir aussi

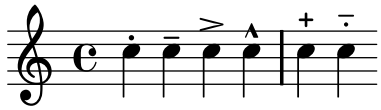
Manuel de l'utilisateur, Liaisons de prolongation,
Manuel de l'utilisateur, Liaisons d'articulation,
Manuel de l'utilisateur, Liaisons de phrasé.

2.2.4 Articulations et nuances

Articulations

Des `articulations` peuvent être ajoutées à une note, au moyen d'un tiret '`-`' suivi d'un caractère :

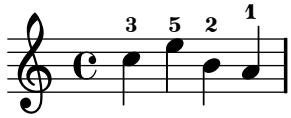
```
c-. c-- c-> c-^ c-+ c-_
```



Doigtés

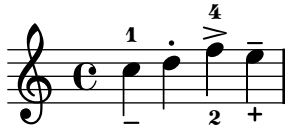
De même, des indications de doigté peuvent être ajoutées à une note en utilisant un tiret ('`-`') et le chiffre à écrire :

```
c-3 e-5 b-2 a-1
```



Articulations et doigtés sont habituellement placés automatiquement, mais vous pouvez spécifier une direction en utilisant '^' (en haut) ou '_' (en bas). Vous pouvez aussi utiliser plusieurs articulations sur la même note. Dans la plupart des cas, cependant, il est mieux de laisser LilyPond déterminer l'emplacement de l'articulation.

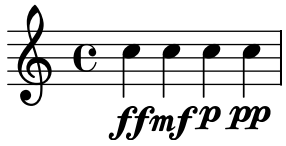
```
c_-^1 d^. f^4_2-> e^-_+
```



Nuances

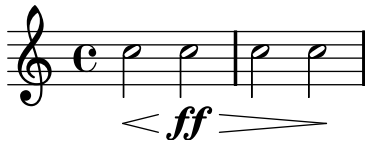
Les signes de nuances sont obtenus en ajoutant à la note les noms des nuances, précédées d'un anti-slash '\' :

```
c\ff c\mf c\p c\pp
```



Crescendos et decrescendos débutent avec les commandes `\<` et `\>`. Ils se terminent soit par une nuance d'arrivée, par exemple `\f`, soit par la commande `\!` :

```
c2\< c2\ff\> c2 c2\!
```



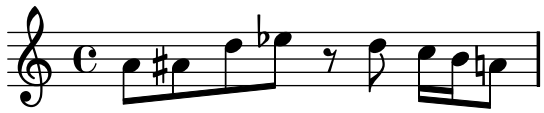
Voir aussi

Manuel de l'utilisateur, Articulations.
Manuel de l'utilisateur, Doigtés.
Manuel de l'utilisateur, Nuances.

2.2.5 Barres de ligature automatiques et manuelles

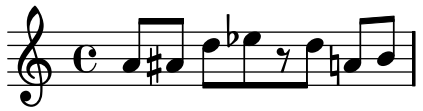
Toutes les `barre de ligature` sont dessinées automatiquement :

```
a8 ais d ees r d c16 b a8
```



Lorsqu'on n'aime pas la manière dont les notes sont automatiquement groupées, il est possible de les ligaturer manuellement, en marquant la première note à attacher d'un '`[`' et la dernière d'un '`]`'.

```
a8[ ais] d[ ees r d] a b
```



Voir aussi

Groupements de notes et ligatures automatiques
: voir *Manuel de l'utilisateur, Barres de ligature automatiques*.

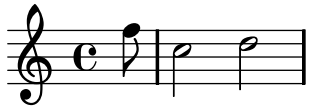
Groupements et ligatures manuels
: voir *Manuel de l'utilisateur, Barres de ligature manuelles*.

2.2.6 Commandes rythmiques avancées

Mesure incomplète

Une levée (ou `anacrouse`) est entrée avec la commande `\partial`, suivie d'une durée : `\partial 4` est une levée d'une noire et `\partial 8` d'une croche.

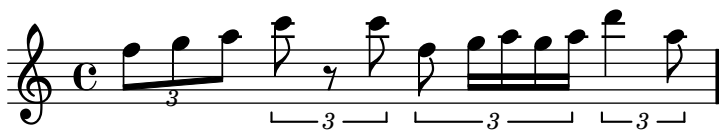
```
\partial 8
f8 c2 d
```



Nolets

Les nolets sont créés avec la commande `\times`, qui prend deux arguments : une fraction et une expression musicale. La durée des notes de l'expression musicale est multipliée par la fraction. Par exemple les notes d'un triolet durent les deux tiers du temps de leur notation réelle, cette fraction est donc de 2/3 pour les triolets :

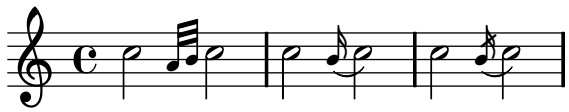
```
\times 2/3 { f8 g a }
\times 2/3 { c r c }
\times 2/3 { f,8 g16[ a g a] }
\times 2/3 { d4 a8 }
```



Notes d'ornement

Des notes d'ornement sont produites par la commande `\grace`, mais aussi en préfixant une expression musicale avec le mot-clé `\appoggiatura` ou `\acciaccatura` :

```
c2 \grace { a32[ b] } c2
c2 \appoggiatura b16 c2
c2 \acciaccatura b16 c2
```



Voir aussi

Notes d'ornement
: voir *Manuel de l'utilisateur, Notes d'ornement*,

nolets
: voir *Manuel de l'utilisateur, Nolets*,

levées
: voir *Manuel de l'utilisateur, Partial measures*.

2.3 Notes simultanées

Cette section traite des situations où l'on a plus d'une note à la fois : plusieurs instruments, plusieurs portées pour un même instrument (le piano, par exemple), et les accords.

La polyphonie, en théorie musicale, est la notion d'une musique constituée de plusieurs voix ; dans lilypond, ce terme désigne les situations où il y a plus d'une voix sur une même portée.

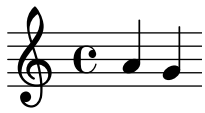
2.3.1 Les expressions musicales en clair

Dans les fichiers source LilyPond, la musique est représentée par ce qu'on appelle des *expressions musicales*. En soi, une seule note peut constituer une expression musicale, si tant est qu'elle soit correctement encadrée :

```
a4
```

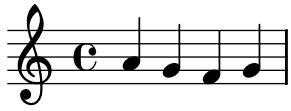
Mettre un groupe de notes entre accolades crée une nouvelle expression musicale :

```
{ a4 g4 }
```



Placer une séquence d'expressions musicales — des notes par exemple — entre accolades signifie qu'elles doivent être jouées successivement, les unes après les autres. Le résultat est une expression, qui peut elle-même être regroupée séquentiellement avec d'autres expressions. Ici, l'expression de l'exemple précédent est combinée à deux notes :

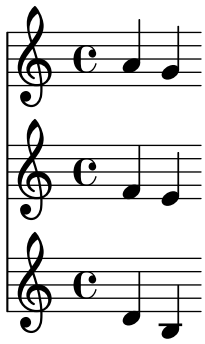
```
{ { a4 g } f g }
```



Expressions musicales simultanées – plusieurs portées

Cette technique est utile pour de la musique polyphonique. Pour entrer une musique avec plusieurs voix ou plusieurs portées, nous pouvons aussi combiner en parallèle les expressions. Deux voix qui doivent être jouées en même temps, sont entrées comme une combinaison simultanée de deux expressions. Une expression musicale 'simultanée' est formée en entourant les expressions entre `<<` et `>>`. Dans l'exemple suivant, trois expressions (contenant chacune deux notes distinctes) sont combinées simultanément.

```
\relative c'' {
  <<
    { a4 g }
    { f e }
    { d b }
  >>
}
```



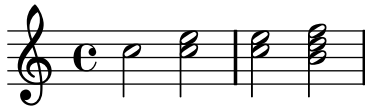
Notez que nous avons ici indenté chaque niveau du fichier d'entrée avec un nombre d'espaces différent. LilyPond se moque de l'espace qu'il peut y avoir (ou pas) au début d'une ligne, mais en indentant votre code vous le rendrez bien plus facile à lire pour les humains.

Attention : chaque note saisie est relative à la précédente, mais pas au `c''` de la commande `\relative` de départ.

Expressions musicales simultanées – une seule portée

Pour déterminer le nombre de portées, LilyPond regarde le premier élément autre qu'une accolade. Si c'est une seule note, il y a une portée ; si c'est une expression simultanée, il y aura plus d'une portée.

```
\relative c'' {
  c2 <<c e>>
  << { e f } { c <<b d>> } >>
}
```



Analogie avec les expressions mathématiques

Ce mécanisme est similaire au formules mathématiques : une grosse formule est créée en assemblant plusieurs petites formules. Ces types de formules, appelées expressions, ont une définition récursive, de telle sorte que vous pouvez fabriquer des expressions arbitrairement longues et complexes. Par exemple,

```
1

1 + 2

(1 + 2) * 3

((1 + 2) * 3) / (4 * 5)
```

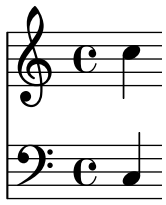
Ceci est une suite d'expressions, où chacune est contenue dans la suivante. Les expressions les plus simples sont les nombres, et de plus grandes expressions sont produites en combinant des expressions avec des opérateurs — comme '`+`', '`*`' et '`/`' — et des parenthèses. Tout comme les expressions mathématiques, les expressions musicales peuvent être imbriquées avec une profondeur arbitraire, ce qui est nécessaire pour de la musique complexe comme des partitions polyphoniques.

2.3.2 Plusieurs portées

Comme nous l'avons vu dans *Manuel de l'utilisateur, Les expressions musicales en clair*, un fichier d'entrée LilyPond est fait d'expressions musicales. Si la partition commence par plusieurs expressions simultanées, LilyPond créera plusieurs portées. Cependant, il est plus facile de voir ce qu'il advient si l'on crée explicitement chacune des portées.

Pour créer plus d'une portée, chaque partie de la musique constituant une portée est entrée en la faisant précéder de `\new Staff`. Ces éléments `Staff` sont ensuite combinés en parallèle avec `<<` et `>>`, comme ceci :

```
\relative c'' {
  <<
    \new Staff { \clef treble c }
    \new Staff { \clef bass c,, }
  >>
}
```



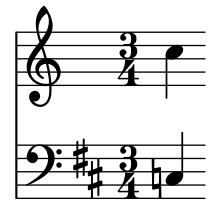
La commande `\new` introduit un « contexte de notation ». Un contexte de notation est un environnement dans lequel les événements musicaux — comme les notes ou les commandes `\clef` — sont interprétés. Pour des pièces simples, ces contextes sont créés automatiquement. Pour des pièces plus complexes, il est préférable de spécifier explicitement les contextes. Cela assure que chaque fragment aura sa propre portée.

Il existe différents types de contextes. Les contextes `Staff` (portée), `Voice` (voix) et `Score` (partition) gèrent la notation de la mélodie, alors que `Lyrics` gère les paroles et `ChordNames` imprime le nom des accords.

En termes de syntaxe, faire précéder une expression musicale de `\new` crée une plus grosse expression musicale. En reprenant la comparaison, cela ressemble au signe « moins » en mathématiques. La formule $(4+5)$ est une expression, donc $-(4+5)$ est une plus grosse expression.

Les chiffres de métrique indiqués sur une portée affectent toutes les autres portées[6]. En revanche l'armure d'une portée n'affecte *pas* les autres portées.

```
\relative c'' {
  <<
    \new Staff { \clef treble \time 3/4 c }
    \new Staff { \clef bass \key d \major c,, }
  >>
}
```



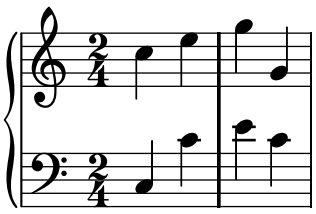
2.3.3 Double portée

La musique pour piano s'écrit sur deux portées reliées par une accolade. Imprimer ce type de portée revient au même que dans l'exemple de musique polyphonique de *Manuel de l'utilisateur, Plusieurs portées*, mais maintenant cette expression entière doit être interprétée dans un contexte `PianoStaff` :

```
\new PianoStaff <<
  \new Staff ...
  \new Staff ...
>>
```

Voici un bref exemple :

```
\relative c'' {
  \new PianoStaff <<
    \new Staff { \time 2/4 c4 e g g, }
    \new Staff { \clef bass c,, c' e c }
  >>
}
```



Voir aussi

Voir *Manuel de l'utilisateur, Musique pour piano*.

[6] Ce comportement peut être modifié si nécessaire, voir *Manuel de l'utilisateur, Notation polymétrique*.

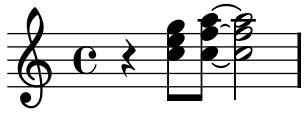
2.3.4 Combinaison de notes en accords

Des accords peuvent être produits en entourant les hauteurs de notes par des angles gauche et droit — '<' et '>' —

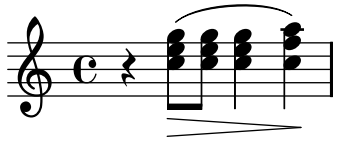
```
r4 <c e g>4 <c f a>2
```

Vous pouvez combiner les indications comme les liaisons et les ligatures de croches avec les accords. Ils doivent cependant être placés en dehors des angles :

```
r4 <c e g>8[ <c f a>]~ <c f a>2
```



```
r4 <c e g>8\>( <c e g> <c e g>4 <c f a>\!)
```

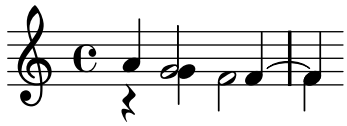


2.3.5 Polyphonie sur une portée

Quand différentes lignes mélodiques sont combinées sur une seule et même portée, elles sont imprimées comme des voix polyphoniques ; chaque voix a ses propre hampes[7], liaisons et ligatures, la voix supérieure ayant les hampes vers le haut, la voix inférieure vers le bas.

Ce type de partition est réalisé en entrant chaque voix comme une séquence (avec `{...}`), en combinant simultanément les voix et en les séparant par `\\` :

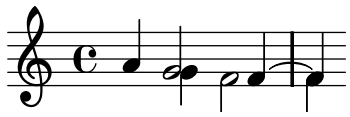
```
<<
  { a4 g2 f4~ f4 } \\
  { r4 g4 f2 f4 }
>>
```



Pour l'écriture de musique polyphonique, les silences invisibles s'avèrent bien pratiques : ce sont des silences qui ne s'impriment pas. Ils sont utiles pour remplir des voix qui, temporairement, ne jouent rien. On peut voir ici le même exemple avec un silence invisible (`s`) à la place d'un silence normal (`r`) :

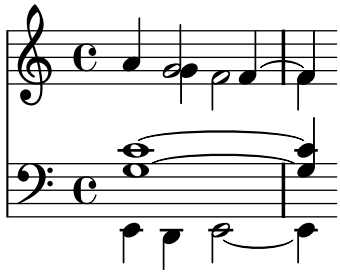
[7] familièrement appelées queues de note.

```
<<
  { a4 g2 f4~ f4 } \\
  { s4 g4 f2 f4 }
>>
```



Là encore, ces expressions peuvent s'imbriquer arbitrairement :

```
<<
  \new Staff <<
    { a4 g2 f4~ f4 } \\
    { s4 g4 f2 f4 }
  >>
  \new Staff <<
    \clef bass
    { <c g>1 ~ <c g>4 } \\
    { e,,4 d e2 ~ e4}
  >>
>>
```



Voir aussi

Voir *Manuel de l'utilisateur, Polyphonie basique*.

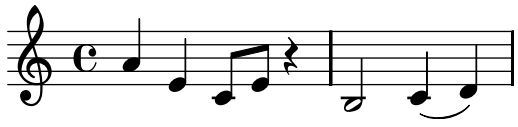
2.4 Chansons

Cette section présente l'écriture vocale et les partitions de variété.

2.4.1 Gravure de paroles

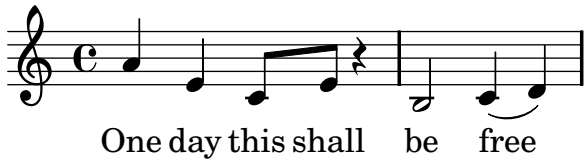
Prenons une mélodie toute simple :

```
\relative c'' {
  a4 e c8 e r4
  b2 c4( d)
}
```



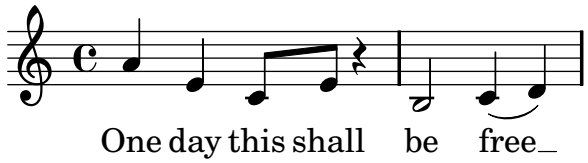
Des paroles peuvent être associées à ces notes, en les combinant avec la commande `\addlyrics`. Les paroles sont entrées en séparant chaque syllable par un espace :

```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free }
>>
```



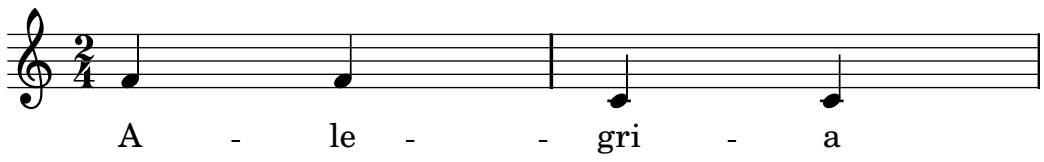
Cette mélodie se termine sur un `mélisme`, c'est-à-dire qu'une seule syllable (« free ») correspond à plus d'une note. Ceci est indiqué avec une *ligne d'extension*. Elle est entrée avec deux caractères souligné (_), c'est-à-dire :

```
<<
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



De la même manière, les séparations syllabiques d'un mot peuvent être entrées avec deux tirets (-), ce qui produit un tiret centré entre les deux syllabes :

```
<<
  \relative c' {
    \time 2/4
    f4 f c c
  }
  \addlyrics { A -- le -- gri -- a }
>>
```



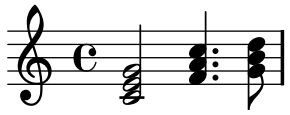
Voir aussi

Plus de possibilités, comme celle d'ajouter plusieurs lignes de paroles en dessous d'une même mélodie sont exposées dans *Manuel de l'utilisateur, Musique vocale*.

2.4.2 Partition d'une chanson

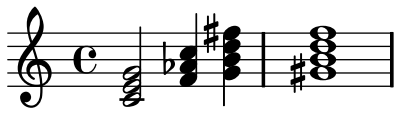
En musique de variété, il est courant d'indiquer l'accompagnement par le nom des accords. De tels accords peuvent être entrés comme les notes :

```
\chordmode { c2 f4. g8 }
```



Maintenant, chaque hauteur est lue comme la base de l'accord à la place de la note. Ce mode est activé avec `\chordmode`. D'autres accords peuvent être créés en ajoutant des modificateurs après deux points. L'exemple suivant montre quelques modificateurs usuels :

```
\chordmode { c2 f4:m g4:maj7 gis1:dim7 }
```

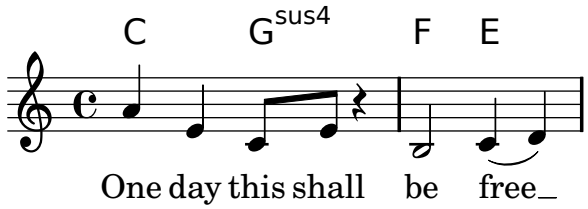


Pour la musique improvisée, les accords ne sont pas imprimés sur des portées mais comme des lignes à part entière. Ceci s'obtient en utilisant `\chords` à la place de `\chordmode`. La même syntaxe sera utilisée que dans le cas de`\chordmode`, mais le rendu des notes interviendra dans un contexte `ChordNames`, avec le résultat suivant :

```
\chords { c2 f4.:m g4.:maj7 gis8:dim7 }
```

Une fois assemblés, les accords, paroles et mélodie forment une partition de chanson :

```
<<
  \chords { c2 g:sus4 f e }
  \relative c'' {
    a4 e c8 e r4
    b2 c4( d)
  }
  \addlyrics { One day this shall be free __ }
>>
```



Voir aussi

Une liste complète de modificateurs et d'autres options de mise en forme se trouve à la section *Manuel de l'utilisateur, Accords*.

2.5 Dernières précisions

L'ultime section de ce tutoriel montre comment ajouter une touche finale à des morceaux simples, et consititue une introduction au reste du manuel.

2.5.1 Numéro de version

La déclaration `\version` stipule le numéro de la version de LilyPond pour laquelle le fichier a été écrit,

```
\version "2.11.38"
```

que l'on place par convention en début de fichier.

Cette annotation permet de faciliter les prochaines mises à jour de LilyPond. Les changements dans la syntaxe sont gérés avec un programme spécial, '`convert-ly`' — voir *Manuel d'utilisation du programme, Mise à jour des fichiers avec convert-ly* — et il utilise `\version` pour déterminer les règles de conversion à appliquer au fichier.

2.5.2 Ajout de titres

Les informations bibliographiques (nom du morceau, du compositeur, etc) sont entrées dans un bloc séparé, le bloc d'en-tête (`\header`), qui existe indépendamment des expressions musicales principales. Le bloc `\header` est habituellement placé en début de fichier.

```
\version "2.11.38"
\header {
  title = "Symphonie"
  composer = "Moi"
  opus = "Op. 9"
}

{
 ... music ...
}
```

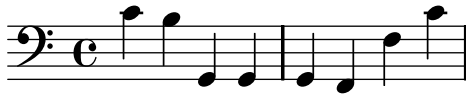
Quand le fichier est traité, le titre et le compositeur sont imprimés en haut de la partition. Vous trouverez plus d'informations sur les titres à *Manuel de l'utilisateur, Création de titres*.

2.5.3 Noms de note absolus

Jusqu'ici nous n'avons utilisé que le mode `\relative` pour définir les hauteurs de notes. Si c'est effectivement le moyen le plus simple d'entrer la majeure partie de votre musique, il existe une autre façon de procéder : le mode des hauteurs absolues.

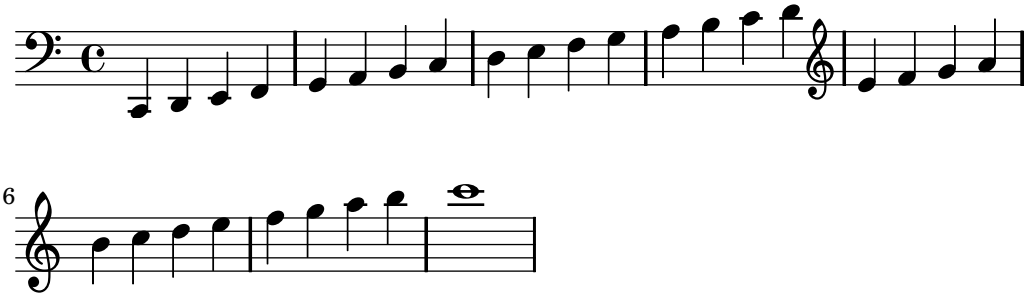
Si vous omettez la commande `\relative`, LilyPond considérera toutes les hauteurs comme des hauteurs absolues. Un `c'` sera toujours un do du milieu, un `b` sera toujours une note au-dessous du précédent, et un `g,` sera toujours la note la plus grave dans la portée de clé de fa.

```
{
  \clef bass
  c' b g, g,
  g, f, f c'
}
```



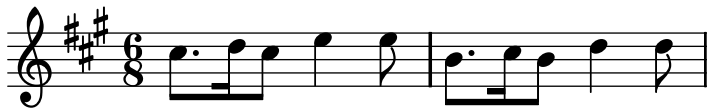
Voici une gamme sur 4 octaves :

```
{
  \clef bass
  c, d, e, f,
  g, a, b, c
  d e f g
  a b c' d'
  \clef treble
  e' f' g' a'
  b' c'' d'' e''
  f'' g'' a'' b''
  c'''1
}
```



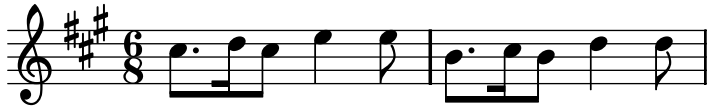
Comme vous pouvez le voir, il faut beaucoup d'apostrophes pour écrire de la musique dans un registre aigu. Regardez cet extrait de Mozart :

```
{
  \key a \major
  \time 6/8
  cis''8. d''16 cis''8 e''4 e''8
  b'8. cis''16 b'8 d''4 d''8
}
```



Toutes ces apostrophes rendent le fichier moins lisible, et c'est donc une source d'erreurs. En mode `\relative`, le même exemple devient bien plus facile à lire :

```
\relative c'' {
  \key a \major
  \time 6/8
  cis8. d16 cis8 e4 e8
  b8. cis16 b8 d4 d8
}
```



Si d'aventure vous faites une erreur d'octaviation, le mode `\relative` la rendra frappante — toutes les notes suivantes seront placées à la mauvaise octave. En mode de hauteurs absolues, une erreur isolée ne serait pas auntant visible, et donc aussi facile à dénicher.

Cependant, le mode de hauteurs absolues reste utile pour les musiques où les intervalles sont étendus, surtout pour les fichiers LilyPond créés par ordinateur.

2.5.4 Organisation du code source avec des variables

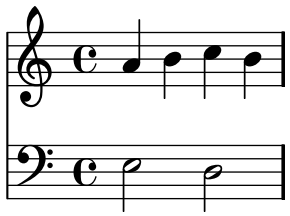
Lorsque l'on combine tous les éléments étudiés plus haut pour produire des fichiers plus volumineux, les blocs `\score` deviennent beaucoup plus gros parce que les expressions musicales sont plus longues et, dans le cas des pièces polyphoniques, profondément imbriquées. De telles expressions imposantes finissent par devenir peu maniables. Cet inconvénient peut être résolu par l'utilisation d'*identificateurs*.

En utilisant ces identificateurs, que l'on pourrait aussi appeler variables ou macros, il est possible de découper des expressions musicales complexes. Un identificateur se définit comme suit :

```
MusiqueToto = { ... }
```

Le contenu de l'expression musicale `MusiqueToto` pourra être utilisé plus loin en faisant précéder son nom d'un anti-slash, c'est-à-dire `\MusiqueToto`, juste comme n'importe quelle commande LilyPond. Tous les identificateurs doivent être définis *avant* l'expression musicale principale.

```
violin = \new Staff { \relative c'' {
  a4 b c b
}}
cello = \new Staff { \relative c {
  \clef bass
  e2 d
}}
{
  <<
    \violin
    \cello
  >>
}
```



Le nom d'un identificateur ne doit comporter que des caractères alphabétiques non accentués, aucun nombre ni tiret.

Il est possible d'utiliser des variables de types variés. Par exemple,

```
width = 4.5\cm
name = "Wendy"
aFivePaper = \paper { paperheight = 21.0 \cm }
```

En fonction de son contenu, un identificateur peut être utilisé à différents endroits. L'exemple suivant utilise la variable ci-dessus :

```
\paper {
  \aFivePaper
  line-width = \width
}
{ c4^\name }
```

2.5.5 Après le tutoriel

Après avoir parcouru ce tutoriel, vous devriez vous essayer à écrire un morceau ou deux. Commencez par copier l'un des *Manuel de l'utilisateur, Modèles* types et ajoutez-y des notes. Si vous voulez employer une notation que vous n'avez pas trouvé dans le tutoriel, consultez la référence de notation, en commençant par la *Manuel de l'utilisateur, Basic notation*. Si vous désirez écrire pour un ensemble instrumental non couvert par les *Manuel de l'utilisateur, Modèles*, lisez la section *Manuel de l'utilisateur, Extension des modèles*.

Après avoir écrit quelques pièces courtes, lisez les chapitres 3 à 5 du manuel d'apprentissage. Rien ne s'oppose à ce que vous consultiez dès à présent les autres chapitres, bien sûr ! Néanmoins, le reste du manuel de l'utilisateur part du principe que vous avez déjà bien assimilé la syntaxe de LilyPond. Vous pouvez toujours survoler le reste du manuel, et y revenir plus tard après avoir acquis de l'expérience.

2.5.6 Bien lire le manuel

Comme nous l'avons déjà vu dans *Manuel de l'utilisateur, Bien lire le tutoriel*, de nombreux exemples du tutoriel n'ont pas fait apparaître `\relative c'' { ... }` dans l'extrait de code affiché.

Dans le reste du manuel, les exemples utilisés sont encore beaucoup plus souples : parfois il leur manque le `\relative c'' { ... }`, mais d'autres fois ils ont recours à une autre hauteur de référence, telle que `c'` ou `c,,`), et dans certains cas c'est même l'exemple entier qui est en mode de hauteurs absolues ! Cependant, de telles ambiguïtés ne se trouvent que dans des contextes où les hauteurs n'ont que peu d'importance. Dans tous les exemples où elles en ont, le mode `\relative` ou absolu `{ }` est explicitement spécifié.

Si vous ne vous y retrouvez toujours pas pour savoir quel code LilyPond produit précisément tel ou tel exemple, consultez la version HTML de ce manuel si ce n'est pas déjà le cas, et cliquez sur l'image de la partition. La source exacte utilisée pour générer ce manuel s'affichera alors.

Pour en savoir plus sur l'organisation de la suite de ce manuel, reportez-vous à *Manuel de l'utilisateur, À propos de ce manuel*.

3 Concepts fondamentaux

3.1 Organisation des fichiers LilyPond

La mise en forme des fichiers d'entrée de LilyPond est vraiment peu astreignante, afin d'offrir assez de souplesse aux utilisateurs expérimentés pour qu'ils puissent organiser leurs fichiers comme ils l'entendent. Cependant, les nouveaux utilisateurs peuvent parfois se perdre en raison de cette souplesse. Cette section présente sommairement l'organisation du code LilyPond, en privilégiant la simplicité au détriment de certains détails. Vous trouverez une description plus complète dans *Manuel de l'utilisateur, Structure de fichier*.

3.1.1 Introduction à la structure de fichier LilyPond

La plupart des exemples de ce manuel sont de courts fragments, par exemple

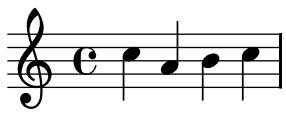
```
c4 a b c
```

Comme vous le savez maintenant (du moins nous l'espèrons), ceci ne peut pas être traité en tant que tel. Il s'agit de formes abrégées des exemples complets ; pour pouvoir être traitées, ces formeulations doivent au moins être encadrées par des accolades :

```
{
  c4 a b c
}
```

La plupart des exemples font aussi intervenir la commande `\relative`, suivie de `c'` ou `c''`. Elle n'est pas à proprement parler nécessaire pour le traitement des exemples, mais dans la plupart des cas le résultat sera vraiment déplorable si vous l'oubliez.

```
\relative c'' {
  c4 a b c
}
```



C'est ici que nous passons aux choses sérieuses : le code LilyPond, sous cette forme, est en réalité un *autre* raccourci. Même s'il est traité sans problème, et aboutit au bon résultat, c'est une forme abrégée de :

```
\score {
  \relative c'' {
    c4 a b c
  }
}
```

Un bloc `\score` doit commencer par une et une seule expression musicale. Rappelez-vous que cette expression peut être ce que vous voulez, d'une note toute seule à un gigantesque

```
{
  \new GrandStaff <<
    collez ici la partition complète de votre opéra de Wagner préféré
  >>
}
```

Dès lors que tout cela est entre accolades : `{ ... }`, c'est une et une seule expression musicale.

Le bloc `\score` peut contenir d'autres éléments, tels que

```
\score {
  { c'4 a b c' }
  \layout { }
  \midi { }
  \header { }
}
```

Certains préfèrent mettre ces commandes en dehors du bloc `\score` — par exemple, on met souvent le `\header` au-dessus. C'est juste là une autre forme abrégée que LilyPond accepte.

Un autre raccourci pratique est la possibilité de définir des variables — également appelées « identificateurs ». Dans tous les modèles, vous trouverez :

```
melodie = \relative c' {
  c4 a b c
}

\score {
  { \melodie }
}
```

Lorsque LilyPond examinera ce fichier, il va prendre la valeur de la variable `melodie`, c'est-à-dire tout ce qui suit le signe `=`, et l'insérer partout où il rencontrera `\melodie`. Vous êtes libre de choisir comment dénommer vos variables[1] ; ce peut être `melodie`, `global`, `maindroitepiano`, ou `laTeteAToto`. Pour plus de détails, voir *Manuel de l'utilisateur, Économies de saisie grâce aux identificateurs et fonctions*.

Pour une description complète du format des fichiers d'entrée, voir *Manuel de l'utilisateur, Structure de fichier*.

3.1.2 La partition est une unique expression musicale composée

Dans la section précédente, nous avons vu l'organisation générale des fichiers d'entrée de LilyPond. Mais c'est comme si nous avions éludé la question essentielle : comment diable peut-on savoir quoi mettre après `\score` ?

En fait, nous ne l'avons pas éludée du tout : le grand mystère est tout simplement qu'il n'y a *pas* de mystère. Allez, expliquons-le en une ligne :

Un bloc `\score` *doit commencer par une et une seule expression musicale.*

Peut-être serait-il judicieux de relire la section *Manuel de l'utilisateur, Les expressions musicales en clair*, dans laquelle vous avez appris à construire de grandes expressions musicales petit bout par petit bout — nous avons vu les notes, puis les accords, etc. Maintenant, nous allons partir d'une grande expression musicale, et remonter la pente.

```
\score {
  {   % cette accolade marque le début de l'expression musicale
    \new GrandStaff <<
    insérez ici votre opéra de Wagner préféré
    >>
  }   % cette accolade marque la fin de l'expression musicale
  \layout { }
}
```

Un opéra de Wagner multiplierait facilement la longueur de ce manuel par deux ou trois, alors faisons-le en version chant/piano. On n'a plus besoin d'une partition d'orchestre — `GrandStaff` — donc laissons cela de côté. Par contre, un chanteur et un piano *pourraient* nous être utiles.

[1] Les noms de variables sont sensibles à la casse, et ne peuvent contenir ni chiffre, ni tiret, ni caractère accentué.

```
\score {
  {
    <<
      \new Staff = "chanteur" <<
      >>
      \new PianoStaff = piano <<
      >>
    >>
  }
  \layout { }
}
```

Vous vous souvenez que nous avons recours à `<<` et `>>` pour mettre en place des musiques simultanées. Et, pour le coup, on aimerait *vraiment* que la partie vocale et l'accompagnement soient imprimés ensemble...

```
\score {
  {
    <<
      \new Staff = "chanteur" <<
        \new Voice = "chant" { }
      >>
      \new Lyrics \lyricsto chant \new Lyrics { }
      \new PianoStaff = "piano" <<
        \new Staff = "mainDroite" { }
        \new Staff = "mainGauche" { }
      >>
    >>
  }
  \layout { }
}
```

On y voit nettement plus clair maintenant. Nous voici donc avec la partie du chanteur, qui contient un ensemble `Voice`, ce qui dans LilyPond correspond à une voix, au sens de voix d'une polyphonie plutôt que de voix chantée — ce pourrait être une partie de violon par exemple.

Nous avons également une partie de piano, qui contient deux portées : une pour la main droite, une autre pour la main gauche.

À ce point, on pourrait commencer à ajouter les notes. Dans les accolades qui suivent `\new Voice = chant`, on pourrait commencer à écrire

```
\relative c'' {
  a4 b c d
}
```

Mais si l'on procédait ainsi, la section `\score` deviendrait vite assez touffue, et très rapidement on ne s'y retrouverait plus. C'est pourquoi on utilisera plutôt des variables, ou identificateurs :

```
melodie = { }
texte = { }
mainDroite = { }
mainGauche = { }
\score {
  {
    <<
      \new Staff = "chanteur" <<
        \new Voice = "chant" { \melodie }
```

```
        >>
        \new Lyrics \lyricsto chant \new Lyrics { \texte }
        \new PianoStaff = "piano" <<
          \new Staff = "mainDroite" { \mainDroite }
          \new Staff = "mainGauche" { \mainGauche }
        >>
      >>
    }
    \layout { }
}
```

Souvenez-vous que vous pouvez donner aux variables le nom que vous voulez, à condition de respecter les caractères autorisés. Ces limitations sont décrites dans *Manuel de l'utilisateur, Structure de fichier*.

Quand on écrit, ou que l'on lit, une section `\score`, mieux vaut y aller lentement et soigneusement. Commencez par le niveau le plus large, puis travaillez sur chaque niveau plus détaillé. À ce propos, une indentation stricte et propre est vraiment d'une aide précieuse : assurez-vous que chaque élément d'un même niveau a le même décalage horizontal dans votre éditeur de texte !

3.1.3 Expressions musicales imbriquées

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.1.4 Non-imbrication des crochets et liaisons

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.2 Les voix contiennent la musique

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.2.1 J'entends des Voix

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.2.2 Instantiation explicite des voix

Les contextes `Voice` peuvent être déclarés manuellement dans un bloc `<< >>` pour créer de la musique polyphonique, en utilisant `\voiceOne`, ... jusqu'à `\voiceFour` pour assigner des directions de hampes et un déplacement horizontal pour chaque partie.

En particulier,

```
<< \upper \\ \lower >>
```

équivaut à

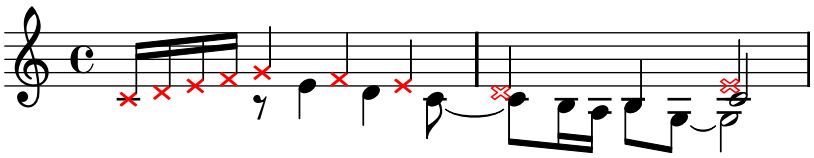
```
<<
 \new Voice = "1" { \voiceOne \upper }
 \new Voice = "2" { \voiceTwo \lower }
>>
```

Les commandes `\voiceXXX` fixent la direction des hampes, des liaisons de prolongations et de phrasé, des articulations, des annotations, des points d'augmentation des notes pointées et des doigtés. `\voiceOne` et `\voiceThree` font pointer ces objets vers le haut, alors que `\voiceTwo` et `\voiceFour` les font pointer vers le bas. La commande `\oneVoice` les ramène aux critères normaux.

Une expression séquentielle qui apparaît en premier dans un `<< >>` appartient à la voix principale. Ceci est utile lorsque des voix supplémentaires apparaissent pendant que la voix principale est jouée. Voici une meilleure réalisation de l'exemple de la section précédente. Les

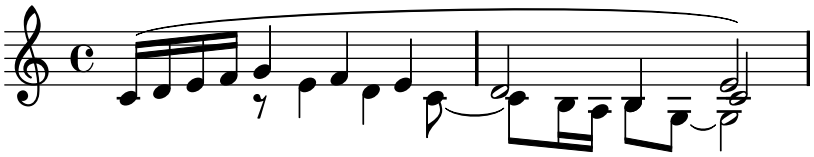
notes colorées et en croix mettent en évidence le fait que la mélodie principale est maintenant dans un seul contexte de voix.

```
\new Staff \relative c' {
  \override NoteHead #'style = #'cross
  \override NoteHead #'color = #red
  c16 d e f
  \voiceOne
  <<
    { g4 f e | d2 e2 }
    \new Voice="1" { \voiceTwo
      r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2
      \oneVoice
    }
    \new Voice { \voiceThree
      s2. | s4 b4 c2
      \oneVoice
    }
  >>
  \oneVoice
}
```



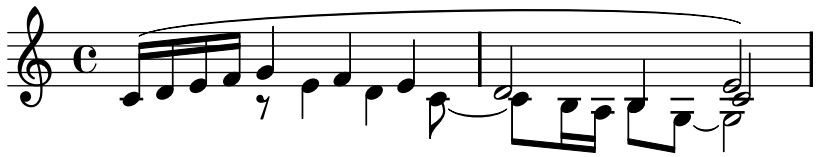
La définition correcte des voix permet à la mélodie d'être liée.

```
\new Staff \relative c' {
  c16^( d e f
  \voiceOne
  <<
    { g4 f e | d2 e2) }
    \context Voice="1" { \voiceTwo
      r8 e4 d c8 ~ | c8 b16 a b8 g ~ g2
      \oneVoice
    }
    \new Voice { \voiceThree
      s2. s4 b4 c2
      \oneVoice
    }
  >>
  \oneVoice
}
```



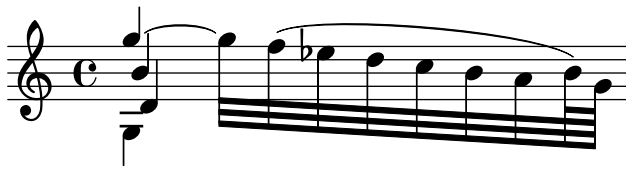
Le fait d'éviter le séparateur \\ permet aussi d'imbriquer des constructions polyphoniques, ce qui peut être une manière plus naturelle de saisir la musique.

```
\new Staff \relative c' {
  c16^( d e f
  \voiceOne
  <<
    { g4 f e | d2 e2) }
    \context Voice="1" { \voiceTwo
      r8 e4 d c8 ~ |
      <<
        {c8 b16 a b8 g ~ g2}
        \new Voice { \voiceThree
          s4 b4 c2
          \oneVoice
        }
      >>
    \oneVoice
    }
  >>
  \oneVoice
}
```



Dans certaines circonstances de polyphonie complexe, vous pourrez être amené à recourir à des voix supplémentaires afin d'éviter des collisions de notes. Ces voix additionnelles s'ajoutent en définissant un identificateur, comme le montre l'exemple suivant :

```
voiceFive = #(context-spec-music (make-voice-props-set 4) 'Voice)

\relative c''' <<
  { \voiceOne g4 ~  \stemDown g32[ f( es d c b a b64 )g] } \\
  { \voiceThree  b4} \\
  { \voiceFive d,} \\
  { \voiceTwo g,}
>>
```



3.2.3 Voix et paroles

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

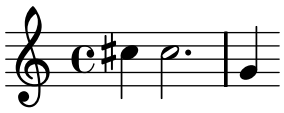
3.3 Contextes et graveurs

Nous avons évoqué rapidement les contextes et graveurs dans les sections précédentes ; examinons en détail ces concepts essentiels dans la maîtrise de LilyPond.

3.3.1 Tout savoir sur les contextes

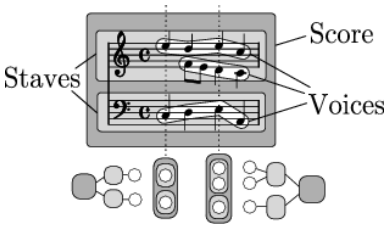
Imprimer de la musique impose d'ajouter un certain nombre d'éléments de notation. Par exemple, voici un fragment de partition, précédé du code qui l'engendre :

```
cis4 cis2. g4
```



Si le code est assez austère, dans la parition ont été ajoutés un chiffre de mesure, des barres de mesure, des altérations et une clé. Pour une bonne raison : LilyPond *interprète* le code. Il le compulse dans l'ordre chronologique, de même qu'on lit une partition de gauche à droite ; et pendant ce traitement, le logiciel garde en mémoire les limites des mesures, ou encore quelles hauteurs de notes demandent des altérations accidentelles. Ces informations se présentent à plusieurs niveaux : ainsi, une altération n'a d'effet que sur une seule portée, tandis qu'une barre de mesure doit être synchronisée sur toute l'étendue verticale de la partition.

LilyPond regroupe ces règles et ces fragments d'information dans des *Contextes*. Certains contextes sont les voix (contexte `Voice`), les portées (contexte `Staff`), ou la partition dans son ensemble (contexte `Score`). Ils sont ordonnés hiérarchiquement : ainsi un contexte `Staff` peut contenir plusieurs contextes `Voice`, et un contexte `Score` peut contenir plusieurs contextes `Staff`.



Chaque contexte est chargé de faire appliquer certaines règles de gravure, de créer certains objets, et de prendre en compte les propriétés qui leur sont associées. Ainsi, le contexte `Voice` peut faire intervenir une altération accidentelle, puis le contexte `Staff` devra déterminer si cette dernière devra être imprimée ou non dans la suite de la mesure. Les barres de mesure, enfin, sont alignées verticalement grâce au contexte `Score`.

En revanche, dans une musique polymétrique, par exemple mélant une portée à 3/4 et une autre à 4/4, les barres de mesures n'ont plus à être alignées : il faut alors modifier les comportement par défaut des contextes `Score` et `Staff`.

Dans une partition très simple, les contextes sont créés implicitement, et peuvent être ignorés. Mais lorsqu'il s'agit de morceaux plus amples – entendons par là tout ce qui s'écrit sur plus d'une portée – il faut les créer explicitement pour être sûr d'obtenir toutes les portées nécessaires, et dans le bon ordre. Enfin pour des morceaux impliquant une notation spéciale, modifier les contextes ou en créer de nouveaux devient extrêment utile.

Une description exhaustive de tous les contextes disponibles peut être trouvée dans la référence du programme : voir Translation ↦ Context.

3.3.2 Création d'un contexte

Dans une partition contenant une seule voix sur une seule portée, les contextes sont automatiquement créés. Dans une partition plus complexe, il faut les créér à la main. Trois commandes le permettent :

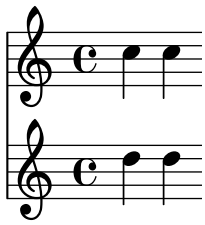
- La plus facile, et la plus rapide à saisir, est `\new` – « nouveau » en français. Elle introduit une expression musicale, comme suit :

  `\new` *`Contexte expression musicale`*

  le choix du *Contexte* pouvant être, par exemple, `Staff` ou `Voice`. Cette commande crée un contexte, puis interprète l'*expression musicale* dans ledit contexte.

  En pratique, la commande `\new` peut servir dans une partition comprenant plusieurs portées. Comme chaque partie doit se trouver sur sa propre portée, il faut la faire précéder de `\new Staff`.

  ```
  <<
    \new Staff { c4 c }
    \new Staff { d4 d }
  >>
  ```

  La commande `\new` peut aussi permettre de nommer le contexte créé :

  `\new` *`Contexte = "inventezUnNom" musique`*

  Le nom que vous choisirez ne pourra être attribué que si aucun autre contexte n'a été créé précédemment avec le même nom.

- Tout comme `\new`, la commande `\context` envoie une expression musicale vers un contexte donné, mais attribue nécessairement un nom à ce contexte. La syntaxe est la suivante :

  `\context` *`Contexte = unNom musique`*

  Cette commande va partir à la recherche d'un contexte déjà existant, de type *Contexte*, et portant le nom *unNom*. Ce qui peut être fort utile pour se référer à un contexte existant. S'il s'avère que le contexte en question n'existe pas encore, il est créé. Dans le cas de musique vocale avec des paroles, cela donne :

  `\context Voice = "tenor"` *`musique`*

  et (pour que le texte soit aligné correctement avec les notes) :

  `\new Lyrics \lyricsto "tenor"` *`paroles`*

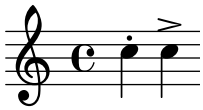
  Une autre utilité de dénommer les contextes est de superposer en un même contexte différentes expressions musicales. Dans l'exemple suivant, les notes et les ponctuations ont été saisies séparément :

  ```
  musique = { c4 c4 }
  ponctuation = { s4-. s4-> }
  ```

  En les envoyant toutes deux dans le même contexte `Voice`, on les combine :

  ```
  <<
    \new Staff \context Voice = "A" \musique
    \context Voice = "A" \ponctuation
  ```

```
>>
```



De cette façon, il est possible d'élaborer une édition Urtext (c'est-à-dire originale, la plupart du temps sans ponctuations), en laissant la possibilité d'ajouter différentes ponctuations sur les mêmes notes.

- La dernière commande pour créer des contextes est

  ```
  \context Contexte musique
  ```

  Elle ressemble à l'emploi de `\context` avec `= unNom`, mais cette fois elle se considèrera chez elle partout où elle trouvera un contexte de type *Contexte*, quel que puisse être son nom.

  Cette variante sert à des expressions musicales qui peuvent être interprétées à plusieurs niveaux. Par exemple, une commande telle que `\applyOutput` (voir `\context`, elle s'applique par défaut dans le contexte `Voice`.

  ```
  \applyOutput #'Contexte #fonction   % s'applique dans le contexte Voice
  ```

  Pour l'appliquer au contexte `Score` ou `Staff`, il faut utiliser :

  ```
  \applyOutput #'Score #fonction
  \applyOutput #'Staff #fonction
  ```

3.3.3 Tout savoir sur les graveurs

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.3.4 Modification des propriétés de contexte

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.3.5 Ajout et suppression de graveurs

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.4 Extension des modèles

Bon, vous avez lu le tutoriel, vous savez écrire de la musique. Mais comment obtenir les portées que vous voulez ? Les Annexe A [Modèles], page 65, c'est bien beau, mais que faire quand ils ne traitent pas ce que l'on veut précisément ?

Les exemples qui suivent vous donneront des méthodes générales pour adapter des modèles.

3.4.1 Soprano et violoncelle

Commencez par le modèle qui vous semblera le plus proche de ce à quoi vous voulez aboutir. Disons par exemple que vous voulez écrire une pièce pour soprano et violoncelle : dans ce cas l'on pourrait commencer par les « notes et paroles », pour la partie de soprano.

```
\version "2.11.38"
melodie = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}
```

```
texte = \lyricmode {
  Laaa Siii Dooo Rééé
}

\score{
  <<
    \new Voice = "voixUn" {
      \autoBeamOff
      \melodie
    }
    \new Lyrics \lyricsto "voixUn" \texte
  >>
  \layout { }
  \midi { }
}
```

Maintenant, on veut ajouter une partie de violoncelle. Jetons un coup d'oeil sur l'exemple avec les notes seules :

```
\version "2.11.38"
melodie = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

\score {
\new Staff \melodie
\layout { }
\midi { }
}
```

On n'a pas besoin de deux commandes `\version`. Ce dont on a besoin, c'est la section `melodie`. De même, on n'a pas besoin de deux sections `\score` — si nous les gardions toutes les deux, on obtiendrait deux parties séparées ; mais nous voulons un vrai duo, avec les deux parties ensemble. Dans la section `\score`, on n'a pas besoin non plus de deux `\layout` ni de deux `\midi`.

Si on se contente de couper et coller les sections `melodie`, on se retrouvera avec deux sections de ce nom ; il nous faut donc les renommer. Appelons la section pour la soprano `musiqueSoprano` et celle pour le violoncelle `musiqueVioloncelle`. Tant qu'on y est, renommons `texte` en `parolesSoprano`. Attention à bien renommer les deux occurrences de chacune de ces dénominations : c'est-à-dire la définition de départ, où l'on trouve `mélodie = relative c' {` , et l'endroit où cette dénomination est utilisée, dans la section `\score`.

Et puis, toujours tant qu'on y est, mettons le violoncelle en clé de Fa, comme le veut l'usage, et donnons-lui d'autres notes.

```
\version "2.11.38"
musiqueSoprano = \relative c' {
  \clef treble
  \key c \major
  \time 4/4
```

```
  a4 b c d
}

parolesSoprano = \lyricmode {
  Laaa Siii Dooo Rééé
}

musiqueVioloncelle = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    \new Voice = "voixUn" {
      \autoBeamOff
      \musiqueSoprano
    }
    \new Lyrics \lyricsto "voixUn" \parolesSoprano
  >>
  \layout { }
  \midi { }
}
```

Voilà qui est mieux, mais la partie de violoncelle n'apparaît pas sur la partition — en effet, nous n'y avons pas fait appel dans la section `\score`. Si l'on veut que la partie de violoncelle s'imprime sous la partie de soprano, on va devoir ajouter :

```
\new Staff \musiqueVioloncelle
```

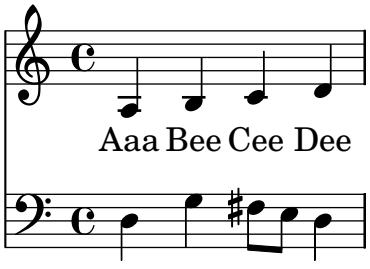
en dessous de tout ce qui concerne la soprano. Il nous faut également encadrer la musique par des `<<` et `>>`, qui feront comprendre à LilyPond que plusieurs évènements — ici, des objets `Staff` — se déroulent en même temps. Le bloc `\score` ressemble maintenant à

```
\score{
  <<
    <<
      \new Voice = "voixUn" {
        \autoBeamOff
        \musiqueSoprano
      }
      \new Lyrics \lyricsto "voixUn" \parolesSoprano
    >>
    \new Staff \musiqueVioloncelle
  >>
  \layout { }
  \midi { }
}
```

C'est un peu le bazar dans tout ça ; mais il vous sera facile de mettre un peu d'ordre dans l'indentation. Voici le modèle pour soprano et violoncelle au complet :

```
\version "2.11.38"
```

```
sopranoMusic = \relative c' {
  \clef treble
  \key c \major
  \time 4/4

  a4 b c d
}

sopranoLyrics = \lyricmode {
  Aaa Bee Cee Dee
}

celloMusic = \relative c {
  \clef bass
  \key c \major
  \time 4/4

  d4 g fis8 e d4
}

\score{
  <<
    <<
      \new Voice = "one" {
        \autoBeamOff
        \sopranoMusic
      }
      \new Lyrics \lyricsto "one" \sopranoLyrics
    >>
    \new Staff \celloMusic
  >>
  \layout { }
  \midi { }
}
```



3.4.2 Partition pour chœur à quatre voix mixtes

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

3.4.3 Écriture d'une partition à partir de zéro

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

4 Retouche des partitions

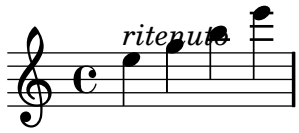
Ce chapitre indique comment modifier le résultat que vous obtiendrez. LilyPond offre de nombreuses possibilités de réglages, permettant de modifier quasiment chaque élément de votre partition.

4.1 Déplacement d'objets

Aussi surprenant que cela puisse paraître, LilyPond n'est pas parfait. Certains éléments sur la partition peuvent se chevaucher, ce qui est regrettable mais, le plus souvent, facile à corriger.

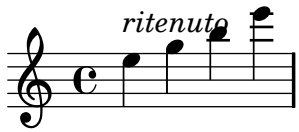
À FAIRE : les modifications de la gestion des espacements de la version 2.12 feront perdre leur pertinence aux exemples suivants. Ils démontrent cependant la puissance de LilyPond, et justifient à ce titre leur présence dans ces lignes, tant que d'autres exemples n'auront pas été proposés.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
e4^\markup{ \italic ritenuto } g b e
```



Le plus simple est ici d'augmenter la distance entre l'objet (du texte comme ici, ou bien des nuances ou des doigtés) et la note. Dans LilyPond, il s'agit de la propriété `padding`, qui se mesure en espaces relatifs à la taille de la portée. Pour la plupart des objets (chacun ayant sa propre valeur), elle est définie à 1.0, ou un peu moins. Nous voulons ici l'augmenter : essayons 1.5.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'padding = #1.5
e4^\markup{ \italic ritenuto } g b e
```

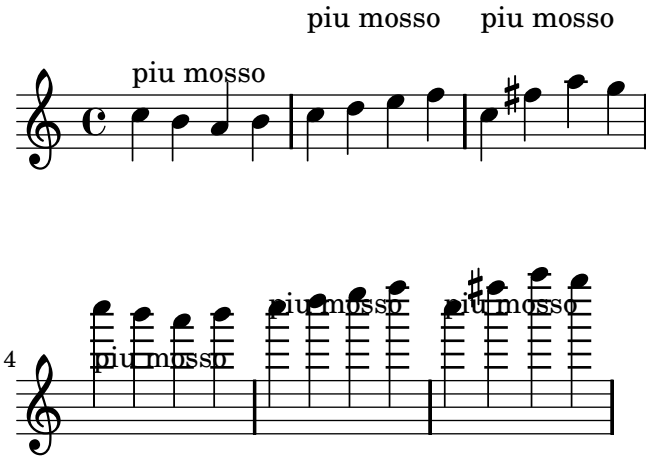


C'est déjà mieux ! Mais on peut certainement encore améliorer le résultat. Il nous semble, après plusieurs essais, que la meilleure valeur dans ce cas soit 2.3. Toutefois, ce constat est le fruit d'expérimentations et de goût personnel en matière de notation. Essayez le même exemple avec 2.3... mais également avec des valeurs plus grandes (ou plus petites). À votre avis, quelle est la meilleure version ?

La propriété `staff-padding` est de nature similaire. `padding` détermine l'espace minimum entre un objet et l'objet le plus proche (le plus souvent une note ou les lignes de la portée) ; `staff-padding` détermine pour sa part l'espace minimum entre un objet et la portée. La différence est subtile, mais vous apparaîtra clairement ici :

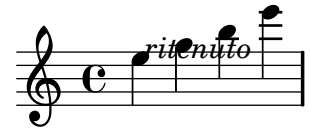
```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b a b
```

```
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
\break
c'4^"piu mosso" b a b
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
\once \override TextScript #'staff-padding = #4.6
c4^"piu mosso" fis a g
```



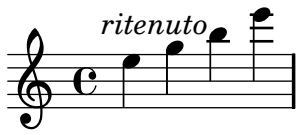
Une autre démarche permet de contrôler totalement la position d'un objet — on peut le déplacer horizontalement ou verticalement. Il suffit d'avoir recours à la propriété `extra-offset`. En fait c'est une méthode plus complexe, qui peut en outre poser des problèmes. Quand on déplace un objet à l'aide de `extra-offset`, le déplacement est effectué après que LilyPond a placé tous les autres objets. Par conséquent, l'objet ainsi déplacé peut venir recouvrir d'autres objets déjà placés.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( 1.0 . -1.0 )
e4^\markup{ \italic ritenuto } g b e
```



Lorsqu'on utilise `extra-offset`, le premier nombre décrit le déplacement horizontal (négatif pour un déplacement vers la gauche) tandis que le deuxième décrit un déplacement vertical (positif pour le haut). Après quelques essais, on peut choisir les valeurs suivantes qui semblent donner un résultat satisfaisant.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
\once \override TextScript #'extra-offset = #'( -1.6 . 1.0 )
e4^\markup{ \italic ritenuto } g b e
```



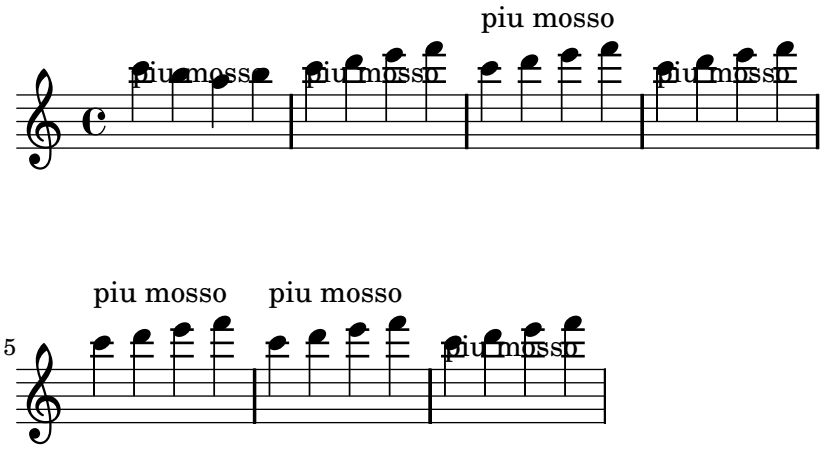
Une fois encore, c'est après quelques tâtonnements que l'on a abouti à ces nombres, au regard du résultat final. Si vous souhaitez que le texte soit plus haut, plus à gauche, etc. essayez vous-même et choisissez aprés avoir regardé le résultat.

Une dernière mise en garde : dans cette section, nous avons eu recours à

```
\once \override TextScript ...
```

ce qui permet de régler le placement du texte pour la note suivante. Mais si cette note n'a pas de texte, le réglage ne s'appliquera pas et n'attendra **pas** le prochain texte. Pour que ce comportement persiste après la commande, ne mettez pas `\once`. Votre réglage s'appliquera alors partout, jusqu'à ce que vous l'annuliez au moyen de la commande `\revert`. Ceci est expliqué en détail dans *Manuel de l'utilisateur, La commande* *override*.

```
  % temporary code to break this example:
  \override TextScript #'outside-staff-priority = ##f
c4^"piu mosso" b
\once \override TextScript #'padding = #4.6
  a4 b
c4^"piu mosso" d e f
\once \override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\break
\override TextScript #'padding = #4.6
c4^"piu mosso" d e f
c4^"piu mosso" d e f
\revert TextScript #'padding
c4^"piu mosso" d e f
```



Voir aussi

Dans ce même manuel : *Manuel de l'utilisateur, La commande* *override*, *Manuel de l'utilisateur, Retouches courantes*.

4.2 Correction des collisions d'objets

Dans la section *Manuel de l'utilisateur, Déplacement d'objets*, nous avons vu comment déplacer un objet `TextScript`. Ce même procédé peut être appliqué à d'autres types d'objet : il vous suffira de remplacer `TextScript` par le nom de l'objet en question.

Pour trouver cette dénomination, regardez les liens '**Voir aussi**' en bas des pages de la documentation. Par exemple, en bas de la page *Manuel de l'utilisateur, Nuances*, nous trouvons

> **Voir aussi**
>
> Référence du programme : `DynamicText`, `Hairpin`. Le placement vertical de ces symboles est contrôlé par `DynamicLineSpanner`.

Ce qui implique que, pour modifier la hauteur d'une nuance, nous utiliserons

```
\override DynamicLineSpanner #'padding = #2.0
```

Nous ne listerons pas ici tous les types d'objets, mais seulement les plus communs :

Type d'objet	**Nom de l'objet**
Nuances (verticalement)	`DynamicLineSpanner`
Nuances (horizontalement)	`DynamicText`
Laisons de tenue	`Tie`
Liaisons	`Slur`
Indications d'articulation	`Script`
Doigtés	`Fingering`
Textes (`^"texte"`)	`TextScript`
Repères	`RehearsalMark`

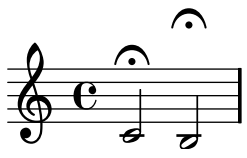
4.3 Retouches courantes

Certains réglages sont si courants que des raccourcis sont fournis sous forme de commandes telles que `\slurUp` ou `\stemDown`. Toutes ces commandes sont décrites dans les différentes sections de la Référence de notation.

La liste complète des modifications possibles pour chaque type d'objet (tel que liaison ou ligature) se trouve dans la Référence du programme. Cependant, certaines propriétés sont communes à de nombreux objets, et on peut de ce fait définir quelques réglages génériques.

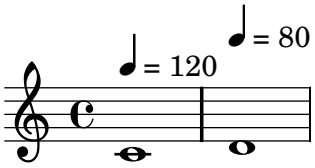
- La propriété `padding` peut être définie de manière à accroître (ou décroître) la distance entre les symboles qui se placent au-dessus ou au-dessous des notes. Ce qui s'applique à tous les objets régis par `side-position-interface`.

```
c2\fermata
\override Script #'padding = #3
b2\fermata
```



```
% This will not work, see below:
\override MetronomeMark #'padding = #3
\tempo 4=120
c1
```

```
% This works:
\override Score.MetronomeMark #'padding = #3
\tempo 4=80
d1
```

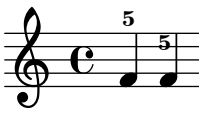


Notez, dans le second exemple, l'importance de savoir à quel contexte correspond l'objet. Dans la mesure où l'objet `MetronomeMark` appartient au contexte `Score`, ses modifications affectées au contexte `Voice` ne l'affecteront pas. Pour plus de détails, voir *Manuel de l'utilisateur, Élaboration d'une retouche*.

- La propriété `extra-offset` permet de déplacer latéralement et verticalement ; c'est pourquoi elle requiert deux nombres. Le premier affecte le placement horizontal (un nombre positif déplace l'objet vers la droite) ; le second le placement vertical (un nombre positif déplace l'objet vers le haut). Cette propriété est de bas niveau : le moteur de formatage ne tient aucun compte des placements qu'elle induit.

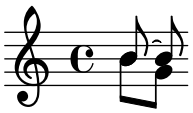
  Dans l'exemple suivant, le second doigté est déplacé un peu vers la gauche, et plus bas de 1.8 espaces :

```
\stemUp
f-5
\once \override Fingering
    #'extra-offset = #'(-0.3 . -1.8)
f-5
```



- La propriété `transparent` imprime les objets avec de l''encre invisible' : l'objet n'est pas visible, mais tous les comportement le concernant s'appliquent quand même. Il occupe une certaine place, intervient dans la gestion des collisions, et on peut lui attacher des liaisons ou des ligatures.

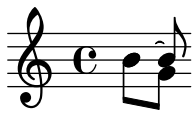
  L'exemple suivant montre comment tenir des notes entre différentes voix, au moyen de liaisons. Ces liaisons de tenue, en principe, ne peuvent relier que deux notes d'une même voix. On introduit donc la liaison dans une autre voix :



et on efface la première croche (hampe vers le haut) de ladite voix ; maintenant la liaison semble passer d'une voix à l'autre :

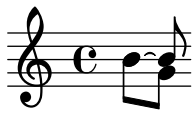
```
<< {
  \once \override Stem #'transparent = ##t
  b8~ b8\noBeam
} \\ {
```

```
  b[ g8]
} >>
```



Pour s'assurer que le crochet de la hampe que nous avons effacée ne raccourcira pas la liaison, nous allons également rallonger cette hampe, en attribuant à la propriété `length` la valeur `8` :

```
<< {
  \once \override Stem #'transparent = ##t
  \once \override Stem #'length = #8
  b8~ b8\noBeam
} \\ {
  b[ g8]
} >>
```



Les distances dans LilyPond sont mesurées dans l'unité staff-space (espace de portée) tandis que la plupart des propriétés relatives aux épaisseurs sont mesurées à l'aide de l'unité line-thickness (épaiseur de ligne). Toutefois, certaines d'entre-elles échappent à cette règle : par exemple l'épaisseur des liens de croches est mesurée à l'aide de l'unité staff-space. Pour de plus amples informations, consultez les sections correspondantes de la Référence du programme.

4.4 Fichiers fournis avec le logiciel

La Référence du programme contient beaucoup d'informations sur LilyPond. Cependant vous pouvez en découvrir encore plus en consultant les fichiers internes de LilyPond.

Des réglages par défaut (tels que les définitions des blocs `\header{}`) sont contenus dans des fichiers `.ly`. D'autres (comme les définitions des commandes « markup ») sont contenus dans des fichiers `.scm` (Scheme). Malheureusement, des explications plus complètes dépassent le cadre de ce manuel. Les utilisateurs qui souhaiteraient comprendre le fonctionnement de ces fichiers de configuration doivent être avertis que des connaissances techniques substantielles et beaucoup de temps sont nécessaires.

- Linux : '*dossierduprogramme*`/lilypond/usr/share/lilypond/current/`'
- Mac OS X : '*dossierduprogramme*`/LilyPond.app/Contents/Resources/share/lilypond/current/`'. Pour accéder à ce dossier, deux possibilités : soit, dans un Terminal, taper `cd` suivi du chemin complet ci-dessus ; soit Control-cliquer (ou clic droit) sur l'application LilyPond et sélectionner 'Afficher le contenu du paquet'.
- Windows : '*dossierduprogramme*`/LilyPond/usr/share/lilypond/current/`'

Les répertoires '`ly/`' et '`scm/`' sont tout particulièrement intéressants. En effet les fichiers du type '`ly/property-init.ly`' ou encore '`ly/declarations-init.ly`' déterminent toutes les définitions avancées communes.

4.5 Réduction du nombre de pages de la partition

Parfois, une partition peut se terminer avec seulement un ou deux systèmes sur la dernière page. Ceci peut être ennuyeux surtout si vous constatez, en regardant les pages précédentes, qu'il reste encore beaucoup de place sur celles-ci.

Si vous vous intéressez aux problèmes de mise en page, `annotate-spacing` peut alors être un outil d'une valeur inestimable. Cette commande imprime les valeurs de nombreuses commandes d'espacement concernant la mise en page. Consultez *Manuel de l'utilisateur, Mise en évidence de l'espacement* pour de plus amples informations. À l'aide des informations données par `annotate-spacing`, on peut voir quelles marges il est souhaitable de modifier afin de résoudre le problème.

En plus d'agir sur les marges, il existe d'autres possibilités qui permettent de gagner de la place.

- Demander à LilyPond de placer les systèmes aussi près que possible les uns des autres (pour en disposer autant que possible sur une page), tout en répartissant les systèmes afin de ne pas laisser de blanc en bas de la dernière page.

```
\paper {
  between-system-padding = #0.1
  between-system-space = #0.1
  ragged-last-bottom = ##f
  ragged-bottom = ##f
}
```

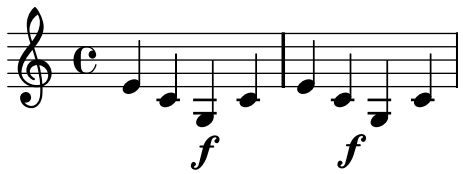
- Obliger LilyPond à mettre un certain nombre de systèmes par page. Par exemple, si LilyPond veut placer onze systèmes dans une page, vous pouvez l'obliger à n'en mettre que dix.

```
\paper {
  system-count = #10
}
```

- Supprimer (ou réduire) les objets qui augmentent la hauteur du système. C'est le cas en particulier de certaines reprises (avec des alternatives) qui placent des crochets au dessus des portées. Si ces crochets de reprise se poursuivent sur deux systèmes, ils prendront plus de place que s'ils sont regroupés sur un même système.

  Un autre exemple : déplacer les nuances qui « débordent » d'un système.

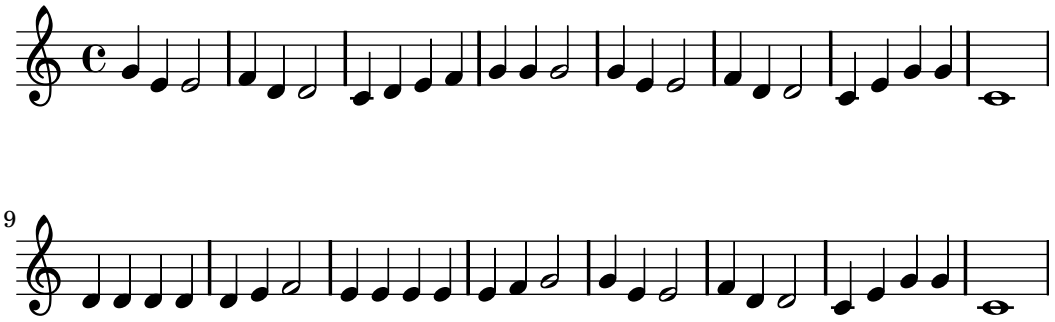
```
\relative c' {
  e4 c g\f c
  \override DynamicLineSpanner #'padding = #-1.8
  \override DynamicText #'extra-offset = #'( -2.1 . 0)
  e4 c g\f c
}
```



- Modifier l'espacement vertical avec `SpacingSpanner`. Reportez-vous à *Manuel de l'utilisateur, Modification de l'espacement horizontal* pour plus de détails.

```
\score {
  \relative c'' {
    g4 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 = #(ly:make-moment 1 4)
    }
  }
}
```

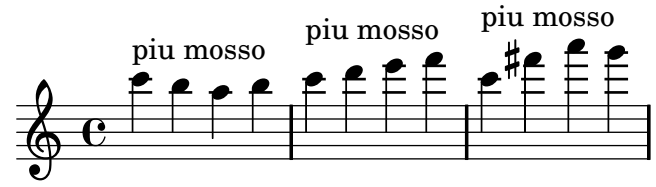


4.6 Retouches avancées avec Scheme

Nous avons déjà vu comment le résultat obtenu avec LilyPond peut être largement personnalisé à l'aide de commandes comme `\override TextScript #'extra-offset = ( 1 . -1)`. Cependant, l'utilisation de Scheme ouvre des possibilités encore plus grandes. Pour des explications complètes là-dessus, consultez le *Manuel de l'utilisateur, Tutoriel Scheme* et les *Manuel de l'utilisateur, Interfaces pour les programmeurs*.

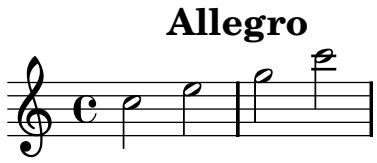
On peut utiliser Scheme simplement à l'aide des commandes `\override`.

```
padText = #(define-music-function (parser location padding) (number?)
#{
  \once \override TextScript #'padding = #$padding
#})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```



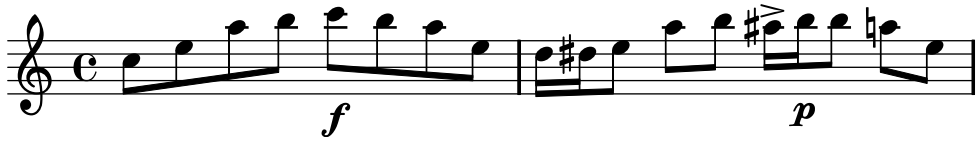
On peut s'en servir pour créer de nouvelles commandes :

```
tempoMark = #(define-music-function (parser location padding marktext)
                                    (number? string?)
#{
  \once \override Score . RehearsalMark #'padding = $padding
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $marktext }
#})

\relative c'' {
  c2 e
  \tempoMark #3.0 #"Allegro"
  g c
}
```



On peut même y inclure des expressions musicales :

```
pattern = #(define-music-function (parser location x y) (ly:music? ly:music?)
#{
  $x e8 a b $y b a e
#})

\relative c''{
  \pattern c8 c8\f
  \pattern {d16 dis} { ais16-> b\p }
}
```



4.7 Options ralentissant le traitement

LilyPond peut effectuer des vérifications supplémentaires lors du traitement des fichiers, cependant le rendu nécessitera alors plus de temps. En contrepartie, il y aura moins d'ajustements manuels à réaliser.

```
%%  Ceci sert à s'assurer que les indications textuelles resteront à l'intérieur des ma
\override Score.PaperColumn #'keep-inside-line = ##t
```

5 Travail sur des projets LilyPond

Cette section explique comment résoudre ou éviter certains problèmes courants. Si vous avez de l'expérience en programmation, beaucoup de ces astuces peuvent vous paraître évidentes, mais vous ne perdrez tout de même pas votre temps à lire ce chapitre.

5.1 Suggestions de saisie des fichiers LilyPond

Maintenant vous êtes prêt à travailler sur de plus gros fichiers LilyPond — des pièces entières, et plus seulement les petits exemples du tutoriel. Mais comment devriez-vous vous y prendre ?

Tant que LilyPond parvient à comprendre vos fichiers et produit le résultat que vous souhaitez, peu importe la manière dont le code est organisé. Néanmoins, quelques critères doivent être pris en compte lorsque l'on écrit un fichier LilyPond.

- Si vous faites une erreur, la structure même du fichier LilyPond peut permettre de la localiser plus ou moins facilement.
- Et si vous souhaitez partager vos fichiers avec quelqu'un d'autre, ou si vous souhaitez modifier vos propres fichiers dans quelques années ? Si certains fichiers LilyPond sont compréhensibles au premier coup d'oeil, d'autres vous feront vous arracher les cheveux pendant une heure.
- Et si vous souhaitez mettre à jour votre fichier pour l'utiliser avec une version plus récente de LilyPond ? La syntaxe du langage d'entrée change parfois lorsque LilyPond s'améliore. La plupart des changements peuvent être appliqués automatiquement avec `convert-ly`, mais quelques-uns peuvent requérir une intervention manuelle. Vos fichiers LilyPond peuvent être structurés de manière à faciliter leur mise à jour.

5.1.1 Suggestions générales

Voici quelques conseils qui peuvent vous éviter certains problèmes ou en résoudre d'autres.

- **Ajoutez le numéro de version dans chaque fichier**. Notez que chaque fichier modèle contient une ligne `\version "2.11.32"`. Nous vous conseillons fortement d'inclure cette ligne, même pour de petits fichiers. Par expérience, il est très difficile de se rappeler quelle version de LilyPond on utilisait quelques années auparavant. L'utilitaire `convert-ly` demande que vous spécifiiez la version de LilyPond vous utilisiez alors.
- **Ajoutez des contrôles**: *Manuel de l'utilisateur, Vérification des limites de mesure*, *Manuel de l'utilisateur, Vérification d'octave* et *Manuel de l'utilisateur, Vérification des numéros de mesure*. Si vous avez ajouté des contrôles de loin en loin, et que vous faites une erreur, vous pourrez la retrouver plus rapidement. « De loin en loin », qu'est-ce à dire ? Cela dépend de la complexité de la musique. Pour de la musique très simple, peut-être une ou deux fois. Pour de la musique très complexe, peut-être à chaque mesure.
- **Une mesure par ligne de texte**. Si la musique en elle-même ou le résultat que vous désirez contient quelque chose de compliqué, il est souvent bon de n'écrire qu'une seule mesure par ligne. Économiser de la place en tassant huit mesures par ligne, ça ne vaut pas vraiment le coup si l'on doît corriger vos fichiers.
- **Ajoutez des commentaires**. Utilisez soit des numéros de mesure (assez souvent), soit des références au contenu musical — « second thème des violons », « quatrième variation », etc. Vous pouvez ne pas avoir besoin des commentaires lorsque vous écrivez une pièce pour la première fois, mais si vous souhaitez y revenir deux ou trois ans plus tard pour changer quelque chose, ou si vous donnez le fichier source à un ami, ce sera beaucoup plus difficile de déterminer vos intentions ou la manière dont votre fichier est structuré si vous n'y avez pas adjoint de commentaires.
- **Indentez les accolades**. Beaucoup de problèmes viennent d'un défaut de parité entre `{` et `}`.

- **Séparez les affinages de mise en forme** de la musique elle-même. Voyez *Manuel de l'utilisateur, Économies de saisie grâce aux identificateurs et fonctions* et *Manuel de l'utilisateur, Feuilles de style*.

5.1.2 Gravure de musique existante

Si vous saisissez de la musique à partir d'une partition existante, c'est-à-dire de la musique déjà écrite,

- n'entrez qu'un seul système de la partition originale à la fois — mais toujours une seule mesure par ligne de texte —, et vérifiez chaque système lorsqu'il est terminé. Vous pouvez utiliser la commande `showLastLength` pour accélérer la compilation — voir *Manuel de l'utilisateur, Ignorer des passages de la partition* ;
- définissez `mBreak = {\break }` et insérez `\mBreak` dans le fichier d'entrée pour obtenir des sauts de ligne identiques à la partition originale. Cela facilite la comparaison entre la partition originale et la partition de LilyPond. Lorsque vous avez fini de relire votre musique, vous pouvez définir `mBreak = { }` pour enlever tous ces sauts de ligne, et laisser LilyPond placer les sauts de ligne selon son propre algorithme.

5.1.3 Projets d'envergure

Lorsque l'on travaille sur un gros projet, il devient vital de structurer clairement ses fichiers LilyPond.

- **Utilisez un identificateur pour chaque voix**, avec un minimum de structure dans la définition. La structure de la section `\score` est la plus susceptible de changer, notamment dans une nouvelle version de LilyPond, alors que la définition du `violon` l'est beaucoup moins.

```
violin = \relative c'' {
g4 c'8. e16
}
...
\score {
 \new GrandStaff {
   \new Staff {
     \violin
   }
 }
}
```

- **Séparez les retouches** des définitions de musique. Ce conseil a été vu dans *Manuel de l'utilisateur, Suggestions générales*, mais pour les projets d'importance c'est absolument vital. Nous pouvons avoir besoin de changer la définition de `fthenp`, mais dans ce cas nous n'aurons besoin de le faire qu'une seule fois, et nous pourrons encore éviter de modifier quoi que ce soit à l'intérieur de la définition du `violon`.

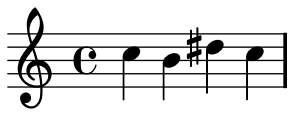
```
fthenp = _\markup{
 \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
g4\fthenp c'8. e16
}
```

5.2 Économies de saisie grâce aux identificateurs et fonctions

Jusqu'à maintenant, vous avez vu ce type de code :

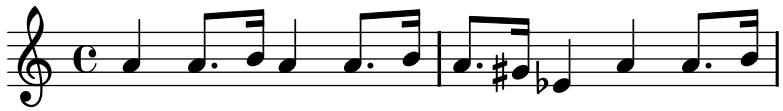
```
hornNotes = \relative c'' { c4 b dis c }
\score {
```

```
  {
    \hornNotes
  }
}
```



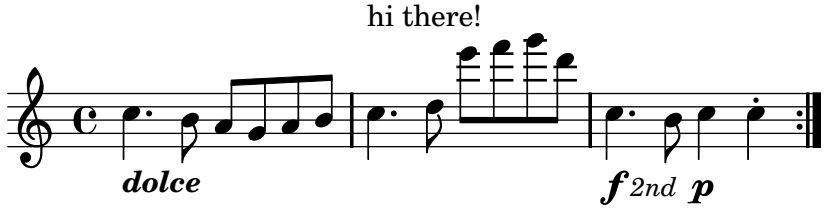
Vous comprendrez combien cela peut être utile pour écrire de la musique minimaliste :

```
fragA = \relative c'' { a4 a8. b16 }
fragB = \relative c'' { a8. gis16 ees4 }
violin = \new Staff { \fragA \fragA \fragB \fragA }
\score {
  {
    \violin
  }
}
```



Cependant, vous pouvez aussi utiliser ces identificateurs — aussi connus sous le nom de variables, macros, ou commandes (définies par l'utilisateur) — pour des retouches :

```
dolce = \markup{ \italic \bold dolce }
padText = { \once \override TextScript #'padding = #5.0 }
fthenp=_\markup{ \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative c'' {
  \repeat volta 2 {
    c4._\dolce b8 a8 g a b |
    \padText
    c4.^"hi there!" d8 e' f g d |
    c,4.\fthenp b8 c4 c-. |
  }
}
\score {
  {
    \violin
  }
\layout{ragged-right=##t}
}
```



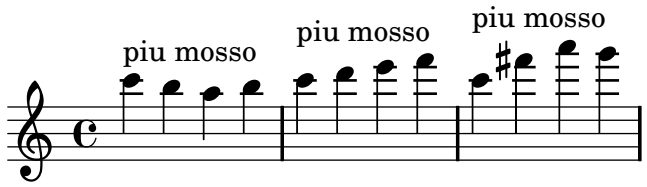
Ces identificateurs sont évidemment utiles pour économiser de la frappe. Mais ils peuvent l'être même si vous ne les utilisez qu'une seule fois : ils réduisent la complexité. Regardons l'exemple précédent sans aucun identificateur. C'est beaucoup plus laborieux à lire, et particulièrement la dernière ligne.

```
violin = \relative c'' {
  \repeat volta 2 {
    c4._\markup{ \italic \bold dolce } b8 a8 g a b |
    \once \override TextScript #'padding = #5.0
    c4.^"hi there!" d8 e' f g d |
    c,4.\markup{ \dynamic f \italic \small { 2nd }
      \hspace #0.1 \dynamic p } b8 c4 c-. |
  }
}
```

Jusqu'ici nous avons vu des substitutions statiques : quand LilyPond rencontre `\padText`, il le remplace par le contenu que nous lui avons défini — c'est-à-dire le contenu à droite de `padText=`.

LilyPond gère également des substitutions non-statiques — vous pouvez les voir comme des fonctions.

```
padText =
#(define-music-function (parser location padding) (number?)
  #{
    \once \override TextScript #'padding = #$padding
  #})

\relative c''' {
  c4^"piu mosso" b a b
  \padText #1.8
  c4^"piu mosso" d e f
  \padText #2.6
  c4^"piu mosso" fis a g
}
```



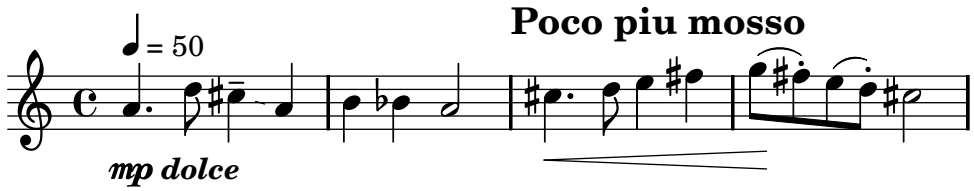
Utiliser les identificateurs est aussi un bon moyen pour vous épargner du travail si la syntaxe de LilyPond change un jour — voir *Manuel de l'utilisateur, Mise à jour d'anciens fichiers*. Si vous avez une seule définition, par exemple `\dolce`, pour tous vos fichiers (voir *Manuel de l'utilisateur, Feuilles de style*), et que la syntaxe change, alors vous n'aurez qu'à mettre à jour votre seule définition `\dolce`, au lieu de devoir modifier chaque fichier `.ly`.

5.3 Feuilles de style

La sortie que produit LilyPond peut être largement modifiée — voir *Manuel de l'utilisateur, Retouche des partitions* pour plus de détails. Mais que faire si vous avez beaucoup de fichiers auxquels vous souhaitez appliquer vos retouches ? Ou si vous souhaitez simplement séparer les retouches de la musique elle-même ? Rien de plus facile.

Prenons un exemple. Ne vous inquiétez pas si vous ne comprenez pas les parties avec tous les `#()`. Celles-ci sont expliquées dans *Manuel de l'utilisateur, Retouches avancées avec Scheme*.

```
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```

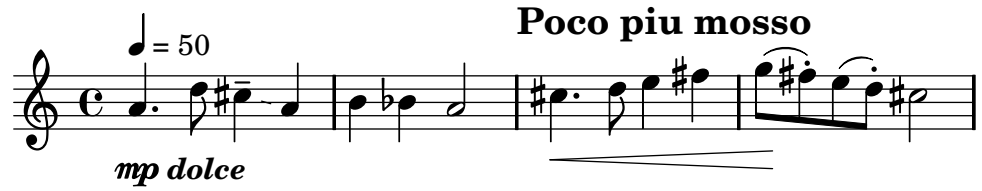


Il y a quelques problèmes de chevauchement ; nous allons arranger cela en utilisant les techniques de *Manuel de l'utilisateur, Déplacement d'objets*. On peut aussi faire quelque chose pour les définitions de `mpdolce` et `tempoMark`. Elles produisent le résultat que nous désirons, mais nous pourrions aussi vouloir les utiliser dans une autre pièce. Il suffirait de les copier et les coller au début de chaque fichier, mais c'est fastidieux. De plus, cela laisse les définitions dans nos fichiers de musique, et je trouve personnellement tous ces `#()` assez laids. Stockons-les dans un autre fichier :

```
%%% enregistrez ceci dans un fichier nommé "definitions.ly"
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line(#:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})
```

Maintenant, modifions notre musique (enregistrez ce fichier sous '`"musique.ly"`').

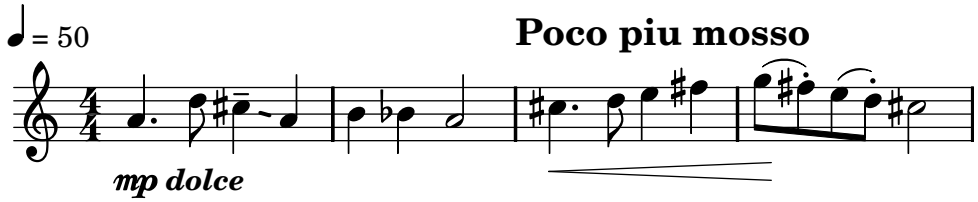
```
\include "definitions.ly"

\relative c'' {
  \tempo 4=50
  a4.\mpdolce d8 cis4--\glissando a | b4 bes a2
  \once \override Score.RehearsalMark #'padding = #2.0
  \tempoMark "Poco piu mosso"
  cis4.\< d8 e4 fis | g8(\! fis)-. e( d)-. cis2
}
```



C'est mieux, mais effectuons encore quelques retouches. Le glissando est peu visible, c'est pourquoi nous allons l'épaissir et le rapprocher des têtes de notes. Déplaçons l'indication métronomique au-dessus de la clef, au lieu de la laisser au-dessus de la première note. Et pour finir, mon professeur de composition déteste les chiffrages de mesure en « C », nous allons donc le transformer en « 4/4 ».

Cependant, ne changez pas le fichier '`musique.ly`'. Remplacez le fichier '`definitions.ly`' par ceci :

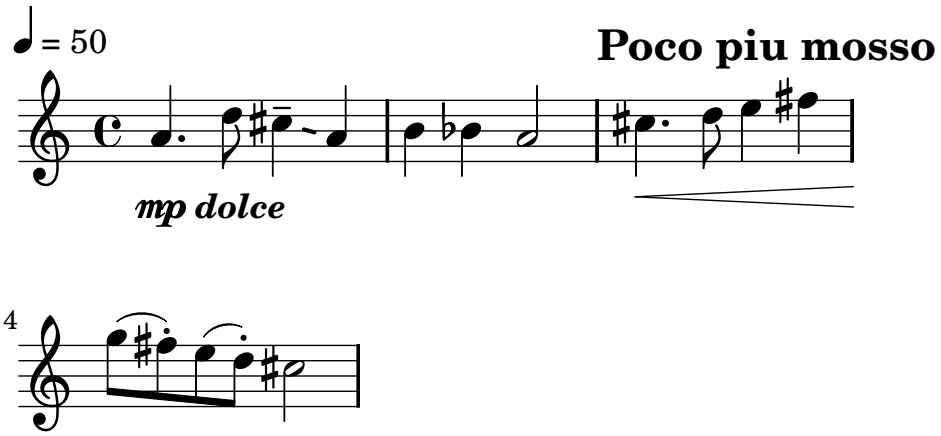
```
%%%  definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 . 0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
    \override TimeSignature #'style = #'numbered
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



C'est encore mieux ! Mais supposons maintenant que je veuille publier cette pièce. Mon professeur de composition n'aime pas les chiffrages de mesure en « C », mais moi je les aime bien. Copions l'actuel '`definitions.ly`' dans le fichier '`publication-web.ly`', et modifions ce dernier. Puisque la musique est destinée à produire un fichier PDF affiché sur écran, nous allons aussi augmenter la taille globale de police.

```
%%%  definitions.ly
mpdolce = #(make-dynamic-script (markup #:hspace 1 #:translate (cons 5 0)
  #:line( #:dynamic "mp" #:text #:italic "dolce" )))
tempoMark = #(define-music-function (parser location markp) (string?)
```

```
#{
  \once \override Score . RehearsalMark #'self-alignment-X = #left
  \once \override Score . RehearsalMark #'extra-spacing-width = #'(+inf.0 . -inf.0)
  \mark \markup { \bold $markp }
#})

#(set-global-staff-size 23)
\layout{
  \context { \Score
    \override MetronomeMark #'extra-offset = #'(-9 .  0)
    \override MetronomeMark #'padding = #'3
  }
  \context { \Staff
  }
  \context { \Voice
    \override Glissando #'thickness = #3
    \override Glissando #'gap = #0.1
  }
}
```



Il ne nous reste plus qu'à remplacer `\include "definitions.ly"` par `\include "publication-web.ly"` dans notre fichier de musique.

Il est possible, bien sûr, de rendre cela encore plus pratique. Nous pourrions créer un fichier '`definitions.ly`' qui ne contiendrait que les définitions de `mpdolce` et de `tempoMark`, un fichier '`publication-web.ly`' qui ne contiendrait que la section `layout` décrite ci-dessus et un fichier '`universite.ly`' qui ne contiendrait que les retouches pour produire le résultat que mon professeur préfère. Le début du fichier '`musique.ly`' ressemblerait alors à

```
\include "definitions.ly"

%%%  Décommentez seulement une de ces deux lignes !
\include "publication-web.ly"
%\include "universite.ly"
```

Cette approche peut être utile même si vous ne produisez qu'un seul jeu de partitions. J'utilise personnellement une demi-douzaine de fichiers de « feuille de style » pour mes projets. Je commence chaque fichier de musique par `\include "../global.ly"` qui contient :

```
%%%   global.ly
\version "2.11.38"
```

```
#(ly:set-option 'point-and-click #f)
\include "../init/init-defs.ly"
\include "../init/init-mise-en-page.ly"
\include "../init/init-en-tetes.ly"
\include "../init/init-papier.ly"
```

5.4 Mise à jour d'anciens fichiers

La syntaxe de LilyPond change de temps en temps. Ces changements de syntaxe du langage d'entrée accompagnent les améliorations du logiciel. Ces changements sont parfois destinés à rendre les fichiers plus faciles à lire et à écrire, ou permettent d'intégrer de nouvelles fonctionnalités.

LilyPond est fourni avec un utilitaire qui facilite cette mise à jour : `convert-ly`. Pour savoir comment utiliser ce programme, voir *Manuel d'utilisation du programme, Mise à jour des fichiers avec convert-ly*.

Malheureusement, `convert-ly` ne peut pas réaliser toutes les modifications. Il s'occupe des changements qui ne requièrent qu'une simple substitution de texte — comme `raggedright` devenant `ragged-right` —, les autres étant trop compliqués à effectuer. Les changements de syntaxe qui ne sont pas gérés par `convert-ly` sont énumérés dans *Manuel d'utilisation du programme, Mise à jour des fichiers avec convert-ly*.

Par exemple, dans les versions 2.4 et antérieures de LilyPond, les accents et les lettres non anglaises étaient entrées en utilisant LaTeX — par exemple, '`No\"el`'. À partir de la version 2.6, le caratère '`ë`' doit être entré directement dans le fichier LilyPond comme caractère UTF-8. `convert-ly` ne peut pas changer tous les caractères LaTeX en caractères UTF-8 ; vous devez mettre à jour vos vieux fichiers LilyPond manuellement.

5.5 Résolution de problèmes — tout remettre à plat

Tôt ou tard, vous écrirez un fichier que LilyPond ne peut pas compiler. Les messages que LilyPond affiche peuvent vous aider à trouver l'erreur, mais dans beaucoup de cas vous aurez besoin de faire quelques recherches pour déterminer la source du problème.

Pour ce faire, les outils les plus puissants sont le commentaire de fin de ligne, indiqué par `%`, et le commentaire multilignes (ou bloc de commentaire), indiqué par `%{ ... %}`. Si vous ne pouvez localiser le problème, commencez par mettre en commentaire de grandes parties de votre fichier d'entrée. Après avoir mis en commentaire une section, essayez de compiler à nouveau. Si cela fonctionne, c'est que le problème se situe dans cette partie du fichier. Si cela ne fonctionne pas, continuez à mettre en commentaire d'autres sections, jusqu'à ce que vous ayez quelque chose qui compile.

Dans un cas extrême, vous pourriez en arriver à

```
\score {
  <<
    % \melodie
    % \harmonie
    % \basse
  >>
  \layout{}
}
```

c'est-à-dire un fichier sans aucune musique.

Si cela arrive, ne vous découragez pas. Décommentez un peu, la partie de basse par exemple, et voyez si ça fonctionne. Si ce n'est pas le cas, placez en commentaire toute la partie de basse, mais laissez `\basse` décommenté dans le bloc `\score`.

```
basse = \relative c' {
%{
  c4 c c c
  d d d d
%}
}
```

Maintenant commencez à décommenter petit à petit le partie de `basse` jusqu'à ce que vous localisiez la ligne qui pose problème.

Une autre technique de déboguage très utile est la construction de *Manuel de l'utilisateur, Exemples minimaux*.

5.6 Exemples minimaux

Un exemple minimal est un exemple de code aussi court que possible. De tels exemples sont bien plus compréhensibles que des exemples longs. Les exemples minimaux sont utilisés pour

- les rapports de bogue,
- les demandes d'aide sur les listes de diffusion,
- un ajout à LilyPond Snippet Repository.

Pour construire un exemple minimal, la règle est très simple : enlevez tout ce qui n'est pas nécessaire. Il est préférable de commenter les lignes non nécessaires plutôt que de les effacer : ainsi, si vous vous apercevez que certaines étaient *réellement* nécessaires, vous pouvez les décommenter au lieu de les resaisir.

Il y a deux exceptions à cette règle du strict nécessaire :

- incluez le numéro de `\version` en début de fichier
- si possible, utilisez `\paper{ ragged-right=##t }` au début de votre exemple.

Tout l'intérêt d'un exemple minimal réside dans sa facilité de lecture :

- évitez d'utiliser des notes, armures ou métriques compliquées, à moins que vous ne vouliez montrer quelque chose en rapport avec celles-ci,
- n'utilisez pas de commandes `\override` sauf si elles font l'intérêt de l'exemple.

Annexe A Modèles

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.1 Portée unique

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.1.1 Notes seules

A.1.2 Notes et paroles

A.1.3 Notes et accords

A.1.4 Notes, paroles et accords

A.2 Modèles pour claviers

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.2.1 Piano seul

A.2.2 Chant et accompagnement

A.2.3 Piano et paroles entre les portées

A.2.4 Piano et nuances entre les portées

A.3 Quatuor à cordes

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.3.1 Quatuor à cordes

A.3.2 Parties pour quatuor à cordes

A.4 Ensemble vocal

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.4.1 Partition pour chœur à quatre voix mixtes

A.4.2 Partition pour chœur SATB avec réduction pour piano

A.4.3 Partition pour chœur SATB avec alignement des contextes

A.5 Exemples de notation ancienne

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.5.1 Transcription de musique mensurale

A.5.2 Transcription du grégorien

A.6 Symboles de jazz

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.7 Squelettes pour lilypond-book

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

A.7.1 LaTeX

A.7.2 Texinfo

Annexe B Tutoriel Scheme

Cette section n'est pas encore traduite ; reportez-vous à l'édition de ce manuel en anglais.

Annexe C Licence GNU de documentation libre

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written 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 that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The 'Document', below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as 'you'.

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. (For example, 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.

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 'Transparent' copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and 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 has been designed to thwart or discourage subsequent modification by readers is not Transparent. 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 designed for human modification. Opaque formats include PostScript, PDF, 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 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.

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 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 publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. 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 less than five).

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', and 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. In any section entitled 'Acknowledgments' or 'Dedications', preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments 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 as 'Endorsements' or to conflict in title with any Invariant Section.

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.

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 'Acknowledgments', 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, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an 'aggregate', and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they 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 quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around 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 provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

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 http://www.gnu.org/copyleft/.

    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.

C.0.1 SUPPLÉMENT : comment utiliser cette licence pour vos 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 1.1
  or any later version published by the Free Software Foundation;
  with the Invariant Sections being list their titles, with the
  Front-Cover Texts being list, and with the Back-Cover Texts being list.
  A copy of the license is included in the section entitled `GNU
Free Documentation License'
.
```

If you have no Invariant Sections, write 'with no Invariant Sections' instead of saying which ones are invariant. If you have no Front-Cover Texts, write 'no Front-Cover Texts' instead of 'Front-Cover Texts being *list*'; likewise for Back-Cover Texts.

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 D Index de LilyPond

\

\context 42
\new 42

A

accents 20
accords 27
accords, noms 30
anacrouse 22
appoggiature 22
armure, définition de 18
articulation 20

B

barre de ligature 21
bémol 18
blanche 14

C

casse, prise en compte de 11, 16
changement de portée manuel 26
chanson, partition complète 30
chansons 28
clef 15
commentaire de fin de ligne 16
commentaire-bloc 16
commentaires 16
Contextes, création de 42
Contexts 6
créeer des contextes 43
crescendo 21

D

decrescendo 21
dièse 18
distances 52
do central 13
documentation du fonctionnement interne 10
doigtés 20
durée 14
DynamicLineSpanner 50
DynamicText 50

E

épaisseur des caractères 2
équilibre 2
espacement optique 2
espacement régulier 3
étendre lilypond 10
exemples de code 10
expression 24
expression musicale 24
extra-offset 48, 51

F

FDL, GNU Free Documentation License 68
fichier PDF 12
fonte 2

G

gamme 13
gravure 5

H

Hairpin 50

I

identificateurs 36, 57
index 10
intervalle 13
invisibles, objets 51

J

jargon 10

L

langage 10
langue 10
langues étrangères 10
levée 22
liaison d'articulation 19
liaison de prolongation 19
liaisons d'articulation 19
liaisons de phrasé 20
liaisons de phrasé et de prolongation, différences 20
liaisons de prolongation 19
ligatures manuelles 21
ligne d'extension 29
lilypond-internals 10
LSR 10

M

masquage d'objets 51
mélisme 29
mesure incomplète 22
métrique 14

N

noire 14
nolets 22
noms d'accords 30
note pointée 14
notes d'ornement 22
nouveaux contextes 42
nuances 21

O

ornementation 22

P

padding 47, 50
paroles 28
partition de chanson 30
police 2
polyphonie 27
propriétés 10

Q

`quarte` 13

R

régulier, espacement 3
régulier, rythme 3
retoucher 10
retouches, distances 52
`ronde` 14

S

Scheme 10
`Score` 41, 43
`silence` 14
staccato 20
`Staff` 41, 43
suppression d'objets 51
symboles musicaux 2

T

terminologie 10
transparents, objets 51
triolets 22
typographie 3, 5

V

variables 10, 36, 57
versions 31
visionnage de la musique 12
`Voice` 38, 41, 42, 43
voix changeant manuellement de portée 26
voix multiples sur une portée 27