GNU LilyPond

Le système de gravure musicale

Manuel de notation

L'équipe de développement de LilyPond

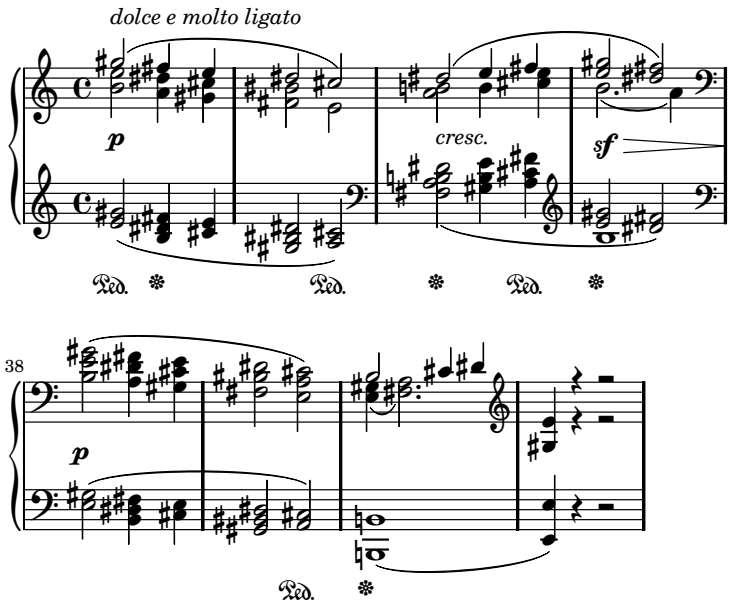
Pour LilyPond version 2.11.58

Table des matières

1 Notation musicale générale

Ce chapitre explique comment créer la notation musicale standard.

1.1 Hauteurs



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

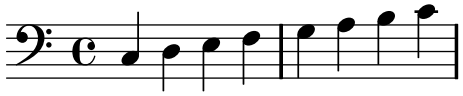
1.1.1 Écriture des hauteurs de note

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

Hauteurs avec octave absolue

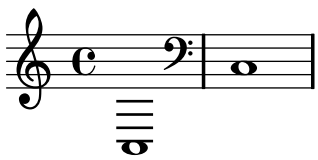
La hauteur s'écrit — à moins de préciser une autre langue — avec la notation anglaise, en utilisant les lettres `a` à `g`. Une gamme ascendante de do majeur s'écrit

```
\clef bass
c d e f g a b c'
```



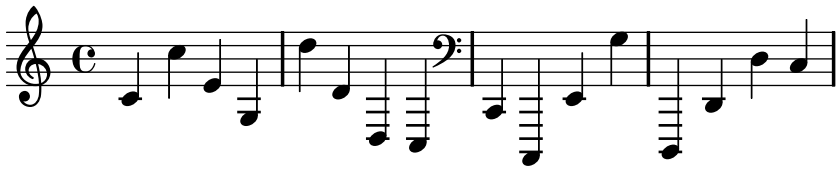
La note `c` est écrite une octave sous le do central.

```
\clef treble
c1
\clef bass
c1
```



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

```
\clef treble
c' c'' e' g d'' d' d c
\clef bass
c, c,, e, g d,, d, d c
```



Il existe une autre méthode pour préciser à quelle octave se situe la note à graver ; cette méthode demande moins d'indications d'octave (`'` ou `,`) — voir [Octaves relatives], page 2.

Octaves relatives

On spécifie les octaves en ajoutant `'` et `,` aux noms de hauteurs. En recopiant de la musique, on a vite fait de mettre une note à la mauvaise octave, et ce genre d'erreur est difficile à retrouver. Le mode d'écriture \relative prévient ces erreurs dans la mesure où elles deviennent beaucoup plus évidentes : une seule erreur décale le reste de la pièce à une mauvaise octave.

```
\relative startpitch musicexpr
```

ou

```
\relative musicexpr
```

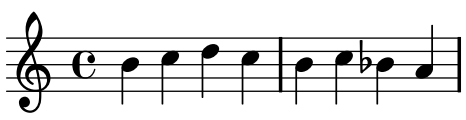
`c'` est utilisé par défaut si aucune hauteur de départ n'est définie.

L'octave des notes mentionnées dans *musicexpr* va être calculée de la manière suivante : si aucun signe de changement d'octave n'est utilisé, l'intervalle de base entre la note actuelle et la précédente sera toujours au plus d'une quarte. Cet intervalle est déterminé sans tenir compte des altérations ; ainsi un `fisis` après un `ceses` sera placé au-dessus du `ceses`. En d'autres termes, une quarte doublement augmentée demeure considérée comme un intervavlle plus petit qu'une quinte diminuée, bien que la quarte doublement augmentée soit de sept demi-tons et la quinte diminuée de seulement six demi-tons.

Les signes de changement d'octave `'` et `,` peuvent être ajoutés pour hausser ou baisser la note d'une octave supplémentaire. Lorsque l'on entre en mode `\relative`, une hauteur absolue de départ peut être spécifiée, et agira dès lors comme si elle précédait la première note de *musicexpr*. Si aucune hauteur de départ n'est spécifiée, le do central sert de point de départ.

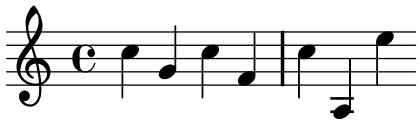
Voici le mode \relative en action.

```
\relative c'' {
  b c d c b c bes a
}
```



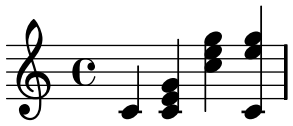
On utilise les signes de changement d'octave pour les intervalles dépassant la quarte.

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



Si l'expression précédente est un accord, c'est la première note de l'accord qui détermine l'emplacement de la première note du prochain accord.

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



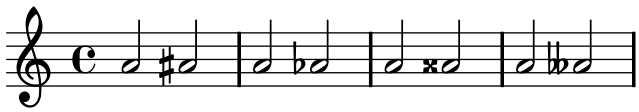
La hauteur après `\relative` contient un nom de note.

La conversion en mode \relative n'affectera pas les sections `\transpose`, `\chordmode` ou `\relative` situées dans son argument. Pour utiliser \relative dans de la musique transposée, un code `\relative` additionnel doit être placé dans `\transpose`.

Altérations

Dans la notation par défaut, un dièse est formé en ajoutant `-is` après le nom de note, un bémol en ajoutant `-es`. Les double-dièses et double-bémols sont obtenus en ajoutant `-isis` ou `-eses` au nom de note.

```
a2 ais a aes
a2 aisis a aeses
```



Ce sont les noms de note hollandais. En hollandais, on élide `aes` pour écrire `as`, mais les deux formes sont acceptées. De manière similaire, on accepte aussi bien `es` que `ees`.

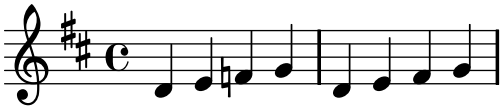
```
a2 as e es
```

Un bécarre annule l'effet d'une altération, qu'elle soit accidentelle ou à l'armure. Cependant, dans la syntaxe des noms de note, les bécarres ne s'écrivent pas avec un suffixe ; un simple nom de note donnera une note bécarre.

```
a4 aes a2
```

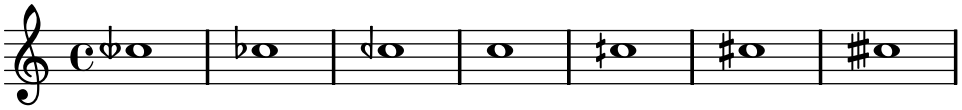
LilyPond interprète l'entrée `d e f` comme « imprimer un ré naturel, un mi naturel et un fa naturel », sans tenir compte de l'armure. Pour plus d'information à propos de la distinction entre le contenu musical et sa représentation, voir Section "Accidentals and key signatures" dans *Manuel d'initiation*.

```
\key d \major
d e f g
d e fis g
```



Les demi-bémols et demi-dièses s'écrivent en ajoutant respectivement `-eh` et `-ih`. Voici une série de dos altérés en hauteurs croissantes :

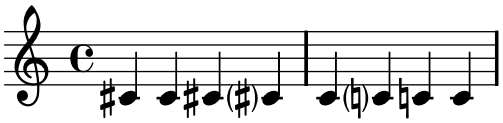
```
ceseh1 ces ceh c cih cis cisih
```



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

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

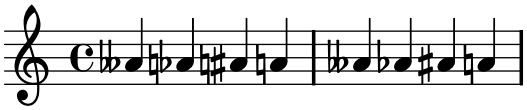
```
cis cis cis! cis? c c? c! c
```



Morceaux choisis

En accord avec les règles standards de l'écriture musicale, on grave un bécarre avant un dièse ou un bémol si on a besoin d'annuler une altération précédente. Pour modifier ce comportement, assignez la propriété `extraNatural` du contexte `Staff` à la valeur `##f` (faux).

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



Voir aussi

Référence du programme : Section "LedgerLineSpanner" dans *Référence des propriétés internes*, Section "NoteHead" dans *Référence des propriétés internes*.

L'impression automatique des altérations peut être affinée de plusieurs manières. Pour plus d'information, voir [Altérations accidentelles automatiques], page 11.

Problèmes connus et avertissements

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

Noms de note dans d'autres langues

Vous disposez de jeux prédéfinis de noms de notes pour plusieurs autres langues. Pour les utiliser, incluez le fichier d'initialisation spécifique à la langue voulue. Par exemple, pour les langues romanes, ajoutez `\include "italiano.ly"` au début du fichier source. Les fichiers de langues disponibles ainsi que les noms de notes utilisés sont les suivants :

	Noms de note								dièse	bémol	double dièse
nederlands.ly	c	d	e	f	g	a	bes	b	-is	-es	-isis
english.ly	c	d	e	f	g	a	bf	b	-s/-sharp	-f/-flat	-ss/-x/ -sharpshsarp
deutsch.ly	c	d	e	f	g	a	b	h	-is	-es	-isis
norsk.ly	c	d	e	f	g	a	b	h	-iss/-is	-ess/-es	-ississ/-isis
svenska.ly	c	d	e	f	g	a	b	h	-iss	-ess	-ississ
italiano.ly	do	re	mi	fa	sol	la	sib	si	-d	-b	-dd
catalan.ly	do	re	mi	fa	sol	la	sib	si	-d/-s	-b	-dd/-ss
espanol.ly	do	re	mi	fa	sol	la	sib	si	-s	-b	-ss

Notez qu'en hollandais, en allemand, en norvégien et en suédois, un '`la`' altéré de bemol se note `aes` et `aeses`. Ces formes sont contractées en `as` et `ases` (ou plus communément `asas`). Dans certains fichiers linguistiques, seules ces formes abrégées ont été définies ; ceci s'applique aussi aux suffixes pour les quarts de ton.

Certaines musiques utilisent des microtonalités, pour lesquelles les altérations sont des fractions de dièse ou bémol « normaux ». Le tableau suivant répertorie les noms de note en quart de ton, tels que définis dans plusieurs fichiers linguistiques. Les préfixes 'semi-' et 'sesqui-' correspondent au 'demi-' et 'trois demis'. À noter qu'aucune définition n'existe à ce jour pour le norvégien, le suédois, le catalan et l'espagnol.

	Noms de note								semi-dièse	semi-bémol	sesqui-dièse	sesqui-bémol
nederlands.ly	c	d	e	f	g	a	bes	b	-ih	-eh	-isih	-eseh
english.ly	c	d	e	f	g	a	bf	b	-qs	-qf	-tqs	-tqf
deutsch.ly	c	d	e	f	g	a	b	h	-ih	-eh	-isih	-eseh
norsk.ly	c	d	e	f	g	a	b	h				
svenska.ly	c	d	e	f	g	a	b	h				
italiano.ly	do	re	mi	fa	sol	la	sib	si	-sd	-sb	-dsd	-bsb
catalan.ly	do	re	mi	fa	sol	la	sib	si				
espanol.ly	do	re	mi	fa	sol	la	sib	si				

1.1.2 Modification de plusieurs hauteurs

Vérifications d'octave

Les tests d'octave rendent la correction d'erreurs d'octave plus facile dans le mode d'octave `relative` : une note peut être suivie de `=`*apostrophes/virgules* pour indiquer à quelle octave absolue elle devrait être. Dans l'exemple suivant,

```
\relative c'' { c='' b=' d,='' }
```

le `d` générera un avertissement, puisqu'on attend un `d''` mais qu'on obtient un `d'` — il n'y a qu'une tierce entre `b'` et `d''`. Sur la partition, l'octave sera corrigée pour donner un `d''` et la prochaine note sera calculée en fonction de `d''` et non de `d'`.

Il existe aussi une vérification d'octave qui ne produit pas de musique imprimée, ayant pour syntaxe

```
\octaveCheck hauteur
```

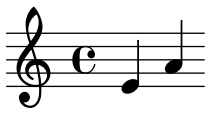
Cette commande vérifie que la *hauteur* considérée sans apostrophe ni virgule et comme hauteur relative, est la même que la *hauteur* absolue considérée avec ses éventuelles apostrophes ou virgules. Sinon, un avertissement est émis et l'octave est corrigée. La *hauteur* n'est pas considérée comme une note, et donc n'est pas imprimée.

Dans l'exemple ci-dessous, le premier test réussit, puisque le `e` — dans le mode `relative` — est au plus à une quarte de `a'`. Cependant, le deuxième test produit un avertissement, puisque le `e` est à quinte de `b'`. Le message d'avertissement est émis, et l'octave est corrigée afin que les notes suivantes soient à nouveau à la bonne octave.

```
\relative c' {
  e
  \octaveCheck a'
  \octaveCheck b'
}
```

L'octave d'une note qui suit un test d'octave est déterminée selon la note précédente. Dans l'exemple suivant, la dernière note est un `a'`, au-dessus du do central. Cela veut dire que le test `\octaveCheck` réussit, et peut donc être enlevé sans changer le résultat sur la partition.

```
\relative c' {
  e
  \octaveCheck b
  a
}
```



Transposition

Une expression musicale peut être transposée avec `\transpose`. Voici la syntaxe :

```
\transpose note_de départ note_d_arrivée musicexpr
```

Cela signifie que *musicexpr* est transposé d'un intervalle entre les notes *note_de_départ* et *note_d_arrivée* : toute note dont la hauteur était `note_de_départ` est changée en `note_d_arrivée`.

Prenons comme exemple une pièce écrite en ré majeur. Si cette pièce est un peu trop basse pour l'interprète, elle peut être transposée en mi majeur avec

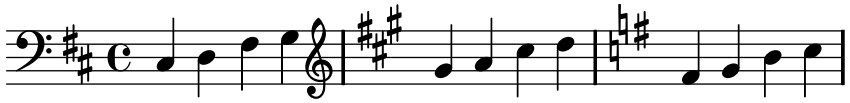
```
\transpose d e ...
```

Regardons maintenant une partie écrite pour violon — un instrument en ut). Si cette partie doit être jouée par une clarinette en la (écrite à la tierce mineure supérieure, un do écrit donnant un la réel), la transposition suivante créera la partie appropriée.

```
\transpose a c ...
```

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

```
mus = { \key d \major cis d fis g }
\new Staff {
  \clef "F" \mus
  \clef "G"
  \transpose c g' \mus
  \transpose c f' \mus
}
```



On peut aussi utiliser `\transpose` pour entrer des notes écrites pour un instrument transpositeur. Normalement, les hauteurs dans LilyPond sont écrites en ut, i.e. en sons réels, mais elles peuvent être écrites dans un autre ton. Quand, par exemple, on écrit pour une trompette en si bémol, commençant sur ré note réelle, on pourrait écrire

```
\transpose c bes { e4 ... }
```

Pour imprimer cette musique en si bémol à nouveau — et de ce fait produire une partie de trompette, au lieu d'un conducteur en notes réelles, on utilisera un deuxième `transpose`

```
\transpose bes c { \transpose c bes { e4 ... } }
```

Voir aussi

Référence du programme : Section "TransposedMusic" dans *Référence des propriétés internes*.

Exemples : Section "Pitches" dans *Exemples de code*.

Problèmes connus et avertissements

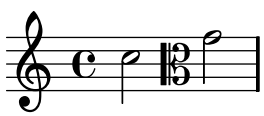
Si vous voulez utiliser en même temps `\transpose` et `\relative`, vous devez mettre `\transpose` en dehors de `\relative`, puisque `\relative` n'aura aucun effet sur la musique apparaissant dans un `\transpose`.

1.1.3 Gravure des hauteurs

Clefs

La clé indique quelles lignes de la portée correspondent à telles hauteurs. Elle est réglée par la commande `\clef`.

```
{ c''2 \clef alto g'2 }
```



Les clés prises en charge sont

Clef	**Position**
`treble`, violin, G, G2	Clé de sol 2e ligne
`alto`, `C`	Clé d'ut 3e ligne
`tenor`	Clé d'ut 4e ligne

`bass, F`	Clé de fa 4e ligne
`french`	Clé de sol 1e ligne
`soprano`	Clé d'ut 1e ligne
`mezzosoprano`	Clé d'ut 2e ligne
`baritone`	Clé d'ut 5e ligne
`varbaritone`	Clé de fa 3e ligne
`subbass`	Clé de fa 5e ligne
`percussion`	Clé de percussion
`tab`	Clé de tablature

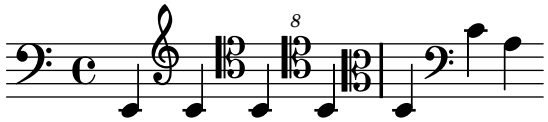
En ajoutant `_8` ou `^8` au nom de la clé, celle-ci est transposée à l'octave respectivement inférieure ou supérieure, et `_15` ou `^15` la transpose de deux octaves. L'argument *clefname* doit être mis entre guillemets lorsqu'il contient un caractère « souligné » ou des chiffres. Par exemple,

```
\clef "G_8" c4
```

Propriétés couramment modifiées

La commande `\clef "treble_8"` équivaut à définir `clefGlyph`, `clefPosition` — qui contrôle la position verticale de la clé — `middleCPosition` et `clefOctavation`. Une clé est imprimée lorsque l'une de ces propriétés est changée. Les exemples suivant font apparaître des possibilités de réglage manuel de ces propriétés.

```
{
  \set Staff.clefGlyph = #"clefs.F"
  \set Staff.clefPosition = #2
  c'4
  \set Staff.clefGlyph = #"clefs.G"
  c'4
  \set Staff.clefGlyph = #"clefs.C"
  c'4
  \set Staff.clefOctavation = #7
  c'4
  \set Staff.clefOctavation = #0
  \set Staff.clefPosition = #0
  c'4
  \clef "bass"
  c'4
  \set Staff.middleCPosition = #4
  c'4
}
```



Voir aussi

Dans ce manuel : Section 1.2.6.1 [Notes d'ornement], page 45.

Référence du programme : Section "Clef" dans *Référence des propriétés internes*.

Armure

L'armure indique la tonalité dans laquelle la pièce doit être jouée. Elle comprend un ensemble d'altérations (dièses ou bémols) à la clé, c'est-à-dire au début de la portée.

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

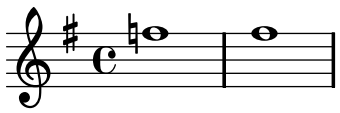
```
\key hauteur type
```

Ici, *type* doit être `\major` ou `\minor` afin d'avoir respectivement *hauteur*-major ou *hauteur*-minor. Vous pouvez aussi avoir recours aux modes anciens que sont `\ionian`, `\locrian`, `\aeolian`, `\mixolydian`, `\lydian`, `\phrygian`, et `\dorian`.

Cette commande fixe la propriété de contexte `Staff.keySignature`. Des armures inhabituelles peuvent être spécifiées en modifiant directement cette propriété.

Les nouveaux utilisateurs s'embrouillent souvent dans les altérations et les armures, car des notes naturelles prennent ou non un bécarre selon l'armure. Pour plus d'informations, voir [Altérations], page 3 ou Section "Accidentals and key signatures" dans *Manuel d'initiation*.

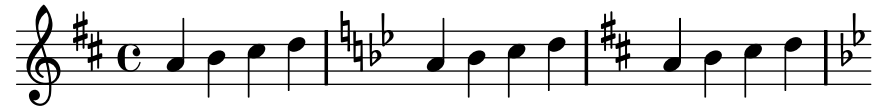
```
\key g \major
f1
fis
```



Propriétés couramment modifiées

Un bécarre est imprimé pour annuler toute altération précédente. Ceci peut être supprimé en réglant la propriété `Staff.printKeyCancellation`.

```
\key d \major
a b cis d
\key g \minor
a bes c d
\set Staff.printKeyCancellation = ##f
\key d \major
a b cis d
\key g \minor
a bes c d
```



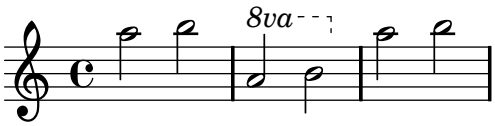
Voir aussi

Référence du programme : Section “KeyCancellation” dans *Référence des propriétés internes*, Section “KeySignature” dans *Référence des propriétés internes*.

Marques d’octavation

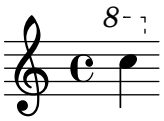
Les marques d’octaviation, *Ottava*, permettent d’introduire une transposition spécifique d’une octave pour la portée en cours. C’est la fonction `ottava` qui s’en charge.

```
\relative c''' {
  a2 b
  \ottava #1
  a b
  \ottava #0
  a b
}
```



La fonction `ottava` peut aussi prendre en argument les valeurs -1 (pour 8va bassa), 2 (pour 15ma — 2 octaves) ou -2 (pour 15ma bassa). En interne, cette fonction détermine les propriétés `ottavation` (p.ex. en `"8va"` ou `"8vb"`) et `centralCPosition`. Vous pouvez modifier le texte d’une marque d’octaviation en définissant `ottavation` après avoir fait appel à `ottava` :

```
{
  \ottava #1
  \set Staff.ottavation = #"8"
  c'''
}
```



Voir aussi

Référence du programme : Section “OttavaBracket” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

`ottava` gère difficilement les changements de clé qui pourraient intervenir alors qu’elle est effective.

Instruments transpositeurs

Vous pouvez spécifier la tonalité d’un instrument transpositeur, ce qui est le cas pour de nombreux instruments à vent comme la clarinette (si bémol, la ou mi bémol), le cor (fa), ou la trompette (si bémol, ut, ré ou mi bémol).

Cette transposition s’indique après le mot-clé `\transposition`.

```
\transposition bes   %% clarinette en si bémol
```

Cette commande détermine la propriété `instrumentTransposition`, dont la valeur sera utilisée pour le fichier MIDI et en cas de citation. Elle n'affecte en rien la manière dont les notes seront imprimées sur la portée. Référez-vous à la section [Transposition], page 6 pour obtenir un autre résultat.

La hauteur donnée en argument à `\transposition` doit correspondre à la note entendue lorsqu'un `do` écrit sur la portée est joué par l'instrument transpositeur. Par exemple, lorsque vous saisissez une partition en notes réelles, toutes les voix devraient être en ut ; pour les instruments transpositeurs, il faudra procéder de cette manière :

```
clarinet = {
  \transposition c'
  ...
}
saxophone = {
  \transposition c'
  ...
}
```

Lorsque vous entrez de la musique à partir d'une partie transposée, utilisez la commande `\transposition`. Si l'on prend l'exemple des parties de cor, leur tonalité change souvent au cours d'un morceau ; en recopiant cette partie, utilisez `\transposition` ainsi :

```
\transposition d'
c'4^"en ré"
...
\transposition g'
c'4^"en sol"
...
```

Altérations accidentelles automatiques

Une fonction a été créée pour regrouper les règles suivant lesquelles s'impriment les altérations. Elle s'invoque de la manière suivante :

```
#(set-accidental-style 'REGLE)
```

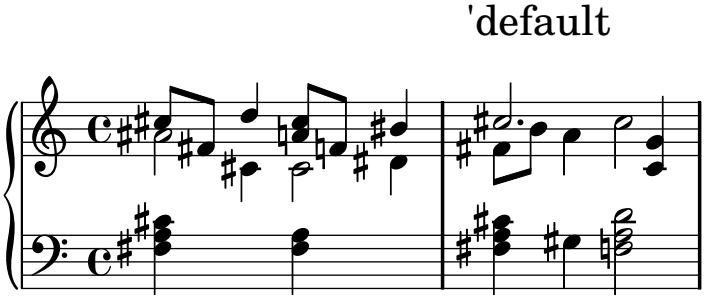
Cette fonction prend pour argument le nom de la règle d'altérations, auquel peut s'ajouter, comme argument facultatif, le contexte devant être affecté :

```
#(set-accidental-style 'REGLE #('CONTEXTE#))
```

Si aucun contexte n'est spécifié, le contexte `Staff` sera affecté ; cependant on peut souhaiter l'appliquer au contexte `Voice` en lieu et place.

Les régles d'altérations suivantes sont possibles :

`default` C'est la règle d'impression par défaut, qui se rapporte à l'usage en vigueur au XVIIIème siècle : les altérations accidentelles sont valables toute une mesure, et uniquement à leur propre octave.

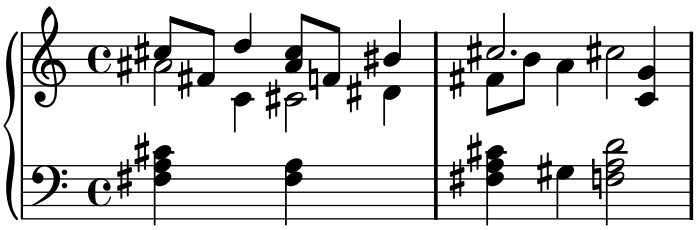


`voice` En principe, LilyPond se souvient de toutes les altérations présentes sur la portée (contexte Staff). Avec cette règle, cependant, les altérations sont indépendantes pour chacune des voix.

```
\new Staff <<
       #(set-accidental-style 'voice)
{ ... }
>>
```

De ce fait, les altérations d'une voix sont ignorées dans les autres voix, ce qui peut donner lieu à un résultat malencontreux. Dans l'exemple suivant, il est difficile de dire si le deuxième '`la`' est dièse ou naturel.

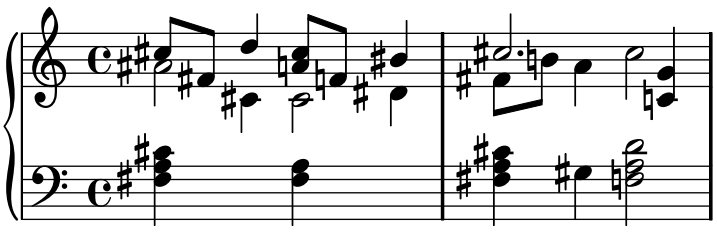
'voice



La règle `voice` n'est à envisager que dans le cas de voix devant être lues par des musiciens différents. S'il s'agit d'un 'conducteur', ou d'une portée destinée à un seul musicien, il vaut mieux utiliser `modern` ou `modern-cautionary`.

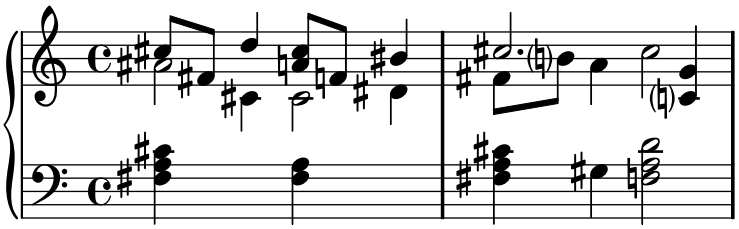
`modern` Cette règle est la plus courante au XXème siècle. Les altérations accidentelles sont imprimées comme avec le style `default`, mais lorsqu'une note non-altérée apparaît à une octave différente, ou bien dans la mesure suivante, des bécarres de précaution sont ajoutés. Dans l'exemple suivant, notez ainsi les deux bécarres dans la deuxième mesure de la main droite.

'modern



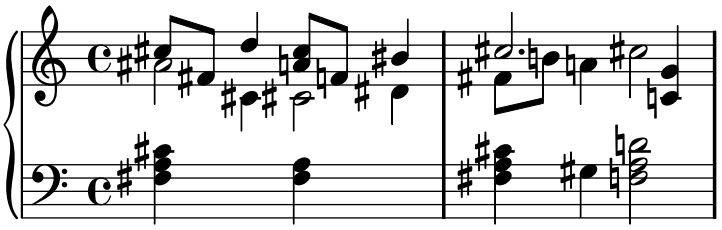
`modern-cautionary`
Cette règle est équivalente à `modern`, mais les bécarres de précaution sont imprimés de façon particulière : soit plus petits, soit (par défaut) entre parenthèses. Il est possible de le définir au moyen de la propriété `cautionary-style` pour l'objet Section "AccidentalSuggestion" dans *Référence des propriétés internes*.

'modern-cautionary



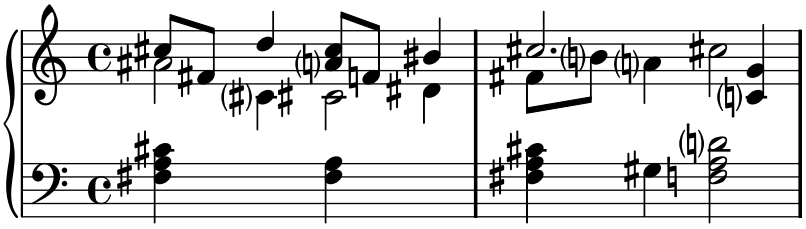
`modern-voice`
: Cette règle sert aux altérations dans de la musique polyphonique destinée autant à des musiciens différents qu'à quelqu'un qui lirait l'ensemble des voix. Les altérations sont imprimées voix par voix, mais les autres voix, dans le même contexte Section "Staff" dans *Référence des propriétés internes*, en *tiennent compte* cette fois.

'modern-voice



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

'modern-voice-cautionary



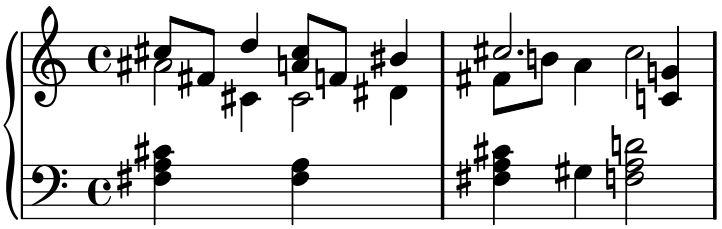
`piano`
: Cette règle est adaptée aux contextes GrandStaff – ce qui n'empêche pas de devoir la spécifier pour chaque portée individuelle au sein du contexte GrandStaff.

```
\new GrandStaff { <<
  \new Staff = "up" { <<
    #(set-accidental-style 'piano)
    { ... }
  >> }
  \new Staff = "down"{ <<
    #(set-accidental-style 'piano)
  { ... }
  >> }
>> }
```

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

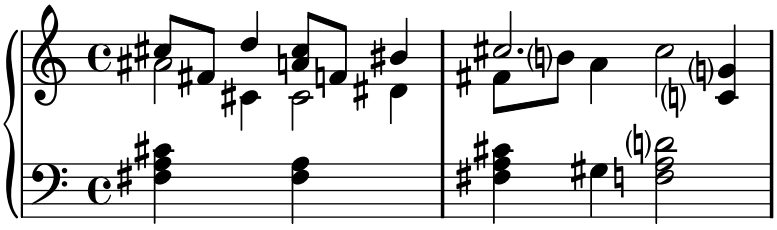
altérations tiennent compte des autre portées du contexte Section “GrandStaff” dans *Référence des propriétés internes* ou Section “PianoStaff” dans *Référence des propriétés internes*.

'piano



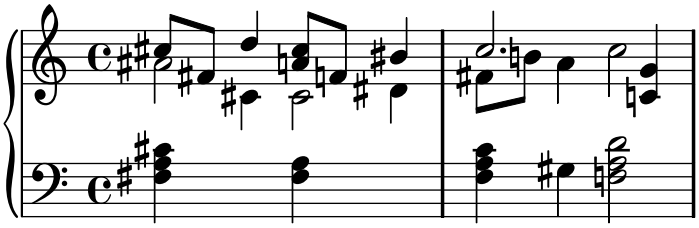
`piano-cautionary`
: Identique à `#(set-accidental-style 'piano)`, mais les altérations de précaution sont imprimées différemment.

'piano-cautionary



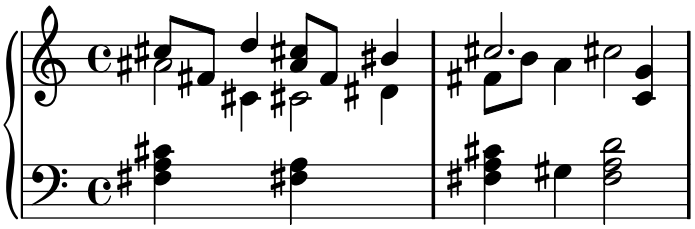
`no-reset` C’est la même règle que `default`, mais l’effet des altérations accidentelles ne cesse jamais, même dans les mesures suivantes.

'no-reset



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

'forget



Voir aussi

Référence du programme : Section “Accidental_engraver” dans *Référence des propriétés internes*, Section “Accidental” dans *Référence des propriétés internes*, Section “AccidentalSuggestion” dans *Référence des propriétés internes* et Section “AccidentalPlacement” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

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

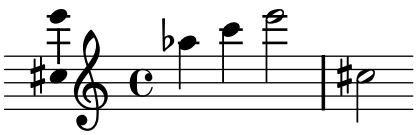
Ambitus

L’*ambitus* est l’amplitude des hauteurs d’une voix donnée dans une partition. Ce terme peut aussi désigner la tessiture qu’un instrument est capable d’atteindre. Souvent, cet ambitus est imprimé au début des partitions vocales, afin que les exécutants puissent voir au premier coup d’oeil s’ils sont en mesure de tenir la partie en question.

Pour exprimer l’ambitus d’une pièce, on indique avant la clé deux notes représentant la hauteur la plus basse et la plus haute. Pour imprimer cet ambitus, il faut ajouter le graveur Section “Ambitus_engraver” dans *Référence des propriétés internes* au contexte Section “Voice” dans *Référence des propriétés internes*. Ainsi,

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

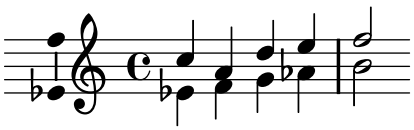
donne pour résultat



Si plusieurs voix se trouvent sur une même portée, on peut attribuer le graveur Section “Ambitus_engraver” dans *Référence des propriétés internes* au contexte Section “Staff” dans *Référence des propriétés internes* plutôt qu’au contexte Section “Voice” dans *Référence des propriétés internes* ; l’ambitus affiché sera alors celui de toutes les voix cumulées, non d’une seule des voix actives.

```
\new Staff \with {
  \consists "Ambitus_engraver"
```

```
}
<<
  \new Voice \with {
    \remove "Ambitus_engraver"
  } \relative c'' {
    \override Ambitus #'X-offset = #-1.0
    \voiceOne
    c4 a d e f2
  }
  \new Voice \with {
    \remove "Ambitus_engraver"
  } \relative c' {
    \voiceTwo
    es4 f g as b2
  }
>>
```



Cet exemple met en œuvre une fonctionnalité avancée :

```
\override Ambitus #'X-offset = #-1.0
```

Ce réglage déplace l'ambitus vers la gauche. Le même résultat aurait pu être obtenu avec `extra-offset`, mais alors le système de mise en forme n'aurait pas attribué d'espace supplémentaire pour l'objet déplacé.

Voir aussi

Référence du programme : Section "Ambitus" dans *Référence des propriétés internes*, Section "AmbitusLine" dans *Référence des propriétés internes*, Section "AmbitusNoteHead" dans *Référence des propriétés internes*, Section "AmbitusAccidental" dans *Référence des propriétés internes*.

Exemples : Section "Pitches" dans *Exemples de code*, Section "Vocal music" dans *Exemples de code*.

Problèmes connus et avertissements

LilyPond ne gère pas les collisions entre plusieurs ambitus présents sur une même portée.

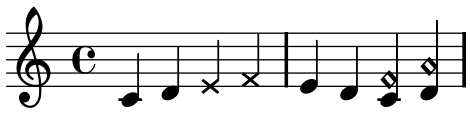
1.1.4 Têtes de note

Têtes de note spécifiques

Certains instruments utilisent des têtes de note différentes à des fins spécifiques — des croix pour le 'parlato' des chanteurs ou les notes étouffées des guitares ; des losanges pour les harmoniques des cordes. Il existe un raccourci (`\harmonic`) pour les notes en losange ; pour les autres styles de tête, il vous faudra jouer avec la propriété `NoteHead`.

```
c4 d
\override NoteHead #'style = #'cross
```

```
e f
\revert NoteHead #'style
e d <c f\harmonic> <d a'\harmonic>
```



Pour une liste exhaustive des styles de tête de note, consultez Section B.5 [Styles de tête de note], page 196.

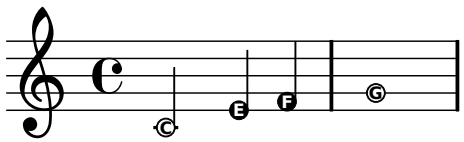
Voir aussi

Référence du programme : Section “NoteHead” dans *Référence des propriétés internes*.

Têtes de note avec nom de note

Les notes ‘easy play’ comportent le nom de la note à l’intérieur de la tête. On l’utilise dans des partitions pour débutants.

```
\easyHeadsOn
c'2 e'4 f' | g'1
```



La commande `\easyHeadsOn` remplace tous les réglages de l’objet Section “NoteHead” dans *Référence des propriétés internes*. L’impression doit être de plus grande taille, afin que les lettres soient lisibles. Voir à ce propos Section 4.2.1 [Définition de la taille de portée], page 171.

Commandes prédéfinies

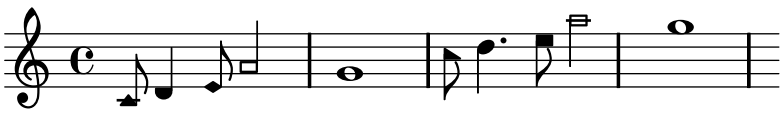
`\easyHeadsOn`

Têtes de note à forme variable

En notation profilée, le profil d’une tête de note correspond à la fonction harmonique de cette note dans la gamme. Ce style de notation était très en vogue dans les recueils de chansons américains du XIXe siècle.

Des notes profilées sont produites après activation de `\aikenHeads` ou `\sacredHarpHeads`, selon le style.

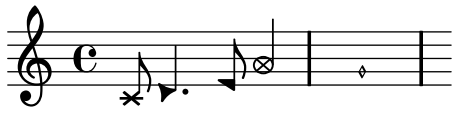
```
\aikenHeads
c8 d4 e8 a2 g1
\sacredHarpHeads
c8 d4. e8 a2 g1
```



Les profils sont déterminés par la hauteur dans la gamme, le premier degré étant défini par la commande `\key`.

Les notes profilées sont mises en œuvre par la propriété `shapeNoteStyles`, dont les valeurs sont constituées d'une liste de symboles. Le n-ième élément indique le style à utiliser pour le n-ième degré de la gamme. Toutes les combinaisons sont possibles :

```
\set shapeNoteStyles  = #'#(cross triangle fa #f mensural xcircle diamond)
c8 d4. e8 a2 g1
```



Improvisation

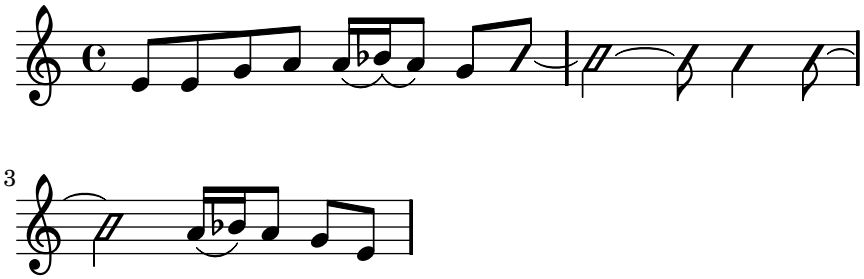
L'improvisation peut quelquefois s'indiquer à l'aide de notes de forme allongée (*slash*). Ces têtes de notes sont créées par l'adjonction du graveur Section "Pitch_squash_engraver" dans *Référence des propriétés internes* au contexte Section "Voice" dans *Référence des propriétés internes*, de telle sorte que la commande

```
\set squashedPosition = #0
\override NoteHead #'style = #'slash
```

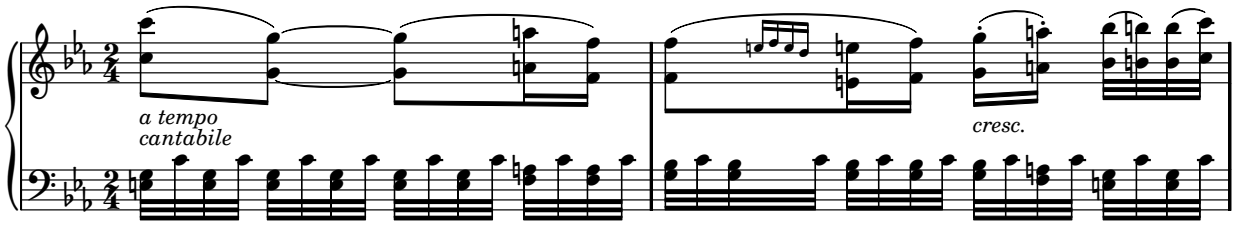
active les notes penchées.

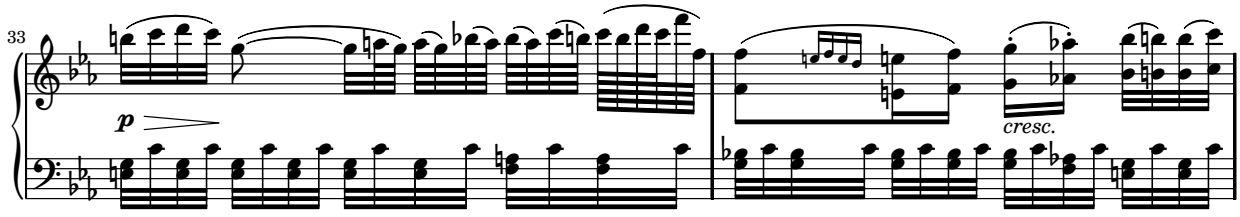
Vous disposez des raccourcis `\improvisationOn` et son corollaire `\improvisationOff` pour cette séquence, comme l'illustre l'exemple ci dessous.

```
\new Voice \with {
  \consists Pitch_squash_engraver
} \transpose c c' {
  e8 e g a a16(bes)(a8) g \improvisationOn
  e8
  ~e2~e8 f4 fis8
  ~fis2 \improvisationOff a16(bes) a8 g e
}
```



1.2 Rythme





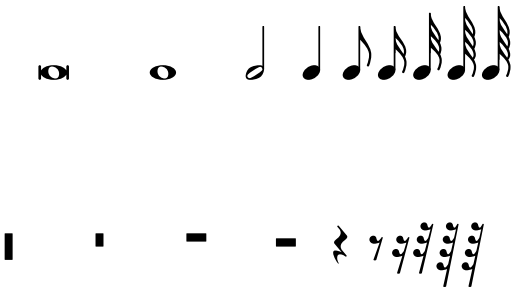
Cette section traite du rythme : durées, silences, barres de ligature et de mesure.

1.2.1 Écriture du rythme

1.2.1.1 Durées

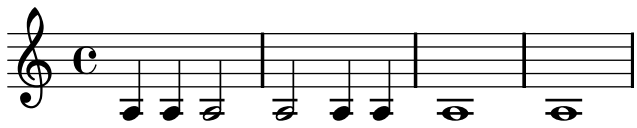
Dans les modes de notes, d'accords et de paroles, les durées sont écrites avec des chiffres et des points : les durées sont indiquées par leur valeur fractionnaire par rapport à la durée d'une ronde. Une noire, par exemple, qui équivaut à un 1/4 de ronde — « quarter note » en anglais — s'écrit `4`, alors qu'une blanche — « half-note », 1/2 ronde — s'écrit `2`. Pour des notes plus longues qu'une ronde, vous devrez utiliser les commandes `\longa` pour une longue, et `\breve` pour une brève, aussi appelée carrée.

```
c'\breve
c'1 c'2 c'4 c'8 c'16 c'32 c'64 c'64
r\longa r\breve
r1 r2 r4 r8 r16 r32 r64 r64
```



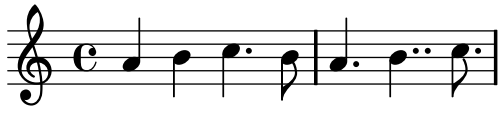
Si la durée n'est pas précisée, elle est alors assimilée à la durée précédente. La valeur par défaut pour la première note est une noire (4).

```
{ a a a2 a a4 a a1 a }
```



Pour obtenir des notes pointées, ajoutez simplement un point (‘`.`’) au chiffre. Les notes doublement pointées sont créées de la même façon.

```
a'4 b' c''4. b'8 a'4. b'4.. c''8.
```



Commandes prédéfinies

Les points sont normalement haussés pour éviter les lignes de portées, sauf dans certaines polyphonies. Les commandes suivantes peuvent être utilisées pour demander manuellement une orientation particulière des points.

`\dotsUp`, `\dotsDown`, `\dotsNeutral`.

Voir aussi

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

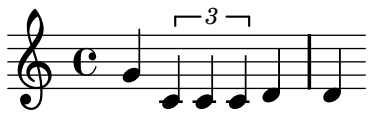
1.2.1.2 Nolets

Les nolets — triolets, quintolets, etc. — sont obtenus en multipliant toutes les durées d’une expression musicale par une fraction.

`\times` *`fraction expr_musique`*

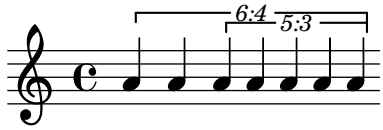
La durée de *expr_musique* sera multipliée par la fraction. Le dénominateur de la fraction sera imprimé au-dessus des notes, parfois avec un crochet. Le nolet le plus courant est le triolet, dans lequel 3 notes ont la durée de 2, et où les notes durent donc 2/3 de leur valeur écrite.

```
g'4 \times 2/3 {c'4 c' c'} d'4 d'4
```



Les nolets peuvent être imbriqués ; par exemple,

```
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 4/6 {
  a4 a
  \times 3/5 { a a a a a }
}
```



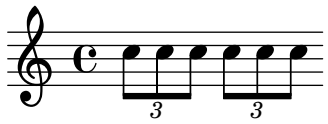
Commandes prédéfinies

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

Propriétés couramment modifiées

La propriété `tupletSpannerDuration` spécifie la longueur voulue de chaque crochet. Avec elle, vous pouvez faire plusieurs nolets en ne tapant `\times` qu’une fois, ce qui évite une longue saisie. Dans le prochain exemple, deux triolets sont imprimés avec une seule fonction `\times`.

```
\set tupletSpannerDuration = #(ly:make-moment 1 4)
\times 2/3 { c8 c c c c c }
```

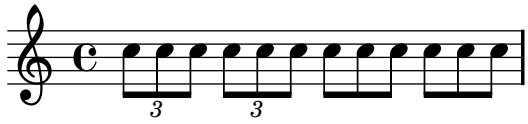


Pour plus d'information sur `make-moment`, voir Section 1.2.6.3 [Gestion du temps], page 49.

L'apparence du chiffre est déterminée par la propriété `text` dans `TupletNumber`. La valeur par défaut imprime seulement le dénominateur, mais si elle est définie par la fonction `tuplet-number::calc-fraction-text`, la fraction entière *num*:*den* sera imprimée à la place.

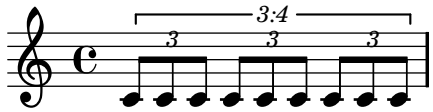
Pour éviter d'imprimer les chiffres des nolets, utilisez

```
\times 2/3 { c8 c c } \times 2/3 { c8 c c }
\override TupletNumber #'transparent = ##t
\times 2/3 { c8 c c } \times 2/3 { c8 c c }
```



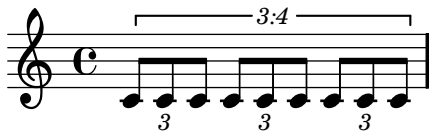
Utilisez la fonction `\tweak` pour définir une priorité dans le cas de nolets imbriqués débutant au même moment. Dans cet exemple, `\tweak` spécifie un texte sous forme de fraction pour le `TupletNumber` externe et de dénominateur pour les trois triolets internes.

```
\new Staff {
  \tweak #'text #tuplet-number::calc-fraction-text
  \times 4/3 {
     \tweak #'text #tuplet-number::calc-denominator-text
     \times 2/3 { c'8[ c'8 c'8] }
     \times 2/3 { c'8[ c'8 c'8] }
     \times 2/3 { c'8[ c'8 c'8] }
  }
}
```

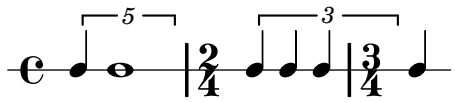


Ici, `\tweak` et `\override` agissent de concert pour spécifier le positionnement du `TupletBracket`. Le premier `\tweak` positionne le `TupletBracket` du nolet externe au dessus de la portée. Le second `\tweak` positionne le `TupletBracket` du premier triolet interne au dessous de la portée. Notez que cette paire de fonctions `\tweak` n'affecte que le triolet extérieur et le premier des trois triolets imbriqués parce qu'ils commencent au même instant musical. Nous utilisons `\override` de manière tout à fait normale, pour forcer le positionnement du `TupletBracket` des deuxième et troisième triolets en dessous de la portée.

```
\new Staff {
  \tweak #'text #tuplet-number::calc-fraction-text
  \tweak #'direction #up
  \times 4/3 {
     \tweak #'direction #down
     \times 2/3 { c'8[ c'8 c'8] }
     \override TupletBracket #'direction = #down
     \times 2/3 { c'8[ c'8 c'8] }
     \times 2/3 { c'8[ c'8 c'8] }
  }
}
```



Les crochets de nolets peuvent aller jusqu'aux prémisses de la mesure suivante, ou à la prochaine note.



Voir aussi

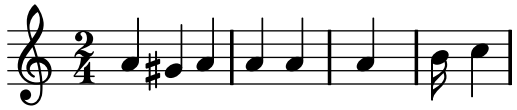
Référence du programme : Section "TupletBracket" dans *Référence des propriétés internes*, Section "TupletNumber" dans *Référence des propriétés internes*, Section "TimeScaledMusic" dans *Référence des propriétés internes*.

1.2.1.3 Changement d'échelle des durées

Vous pouvez altérer la durée des notes en leur joignant une fraction *N/M*, donnant '**N/M*' — ou '**N*' si *M=1*. Ceci n'affectera pas l'apparence des notes ou silences produits.

Dans l'exemple suivant, les trois premières notes prennent exactement deux temps, mais aucun triolet n'est imprimé.

```
\time 2/4
a4*2/3 gis4*2/3 a4*2/3
a4 a4 a4*2
b16*4 c4
```



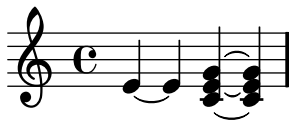
Voir aussi

Dans ce manuel : Section 1.2.1.2 [Nolets], page 20.

1.2.1.4 Liaisons de prolongation

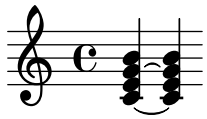
Une liaison de tenue (ou de prolongation) relie deux notes adjacentes de même hauteur. Dans les faits, elle prolonge la durée d'une note, et ne doit donc pas être confondue avec les liaisons d'articulation ou de phrasé. Une liaison de tenue est indiquée au moyen d'un tilde '~'.

```
e' ~ e' <c' e' g'> ~ <c' e' g'>
```

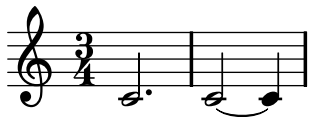


Quand une liaison de tenue se trouve entre deux accords, toutes les notes de même hauteur entre ces deux accords sont reliées. S'il n'y en a aucune, aucune liaison n'est créée. Il est également possible de lier partiellement deux accords, en mettant les liaisons à l'intérieur des accords.

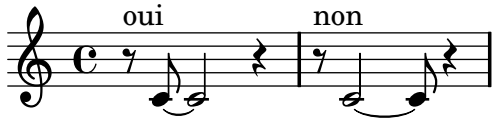
```
<c~ e g~ b> <c e g b>
```



Une liaison de tenue est un moyen parmi d'autres pour prolonger la durée d'une note, tout comme les points. L'exemple suivant montre deux manières de matérialiser exactement la même idée :

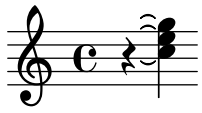


Les liaisons de tenues sont utilisées soit lorsque la note dépasse de la mesure, soit quand les points ne suffisent pas à donner la bonne durée. Lorsque l'on utilise ces liaisons, les valeurs rythmiques les plus longues doivent s'aligner sur les subidivisions de la mesure, comme ici :



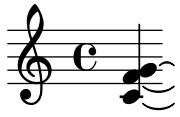
Lorsque l'on doit lier de nombreuses notes sur plusieurs mesures, il devient plus facile d'avoir recours à la division automatique des notes — voir Section 1.2.3.5 [Découpage automatique des notes], page 33. Ce procédé divise automatiquement les notes trop longues, et les lie par-delà les barres de mesure.

Lorsqu'une mesure de seconde fois après une reprise commence sur une note liée, la liaison doit être répétée. C'est à cela que sert la commande `\repeatTie` :



Les liaisons « Laissez vibrer » (L.V.) sont utilisées pour le piano, la harpe, et certains instruments de percussion. Elles indiquent à l'instrumentiste de laisser sonner la note ou l'accord au lieu de l'étouffer. Cet effet s'obtient avec la commande `\laissezVibrer`.

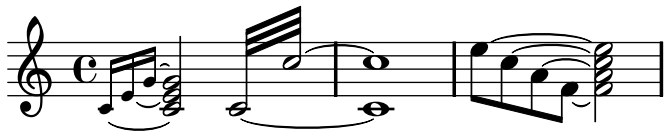
```
<c f g>\laissezVibrer
```



Propriétés couramment modifiées

Les liaisons de tenue servent parfois à rendre un accord arpégé. Dans ce cas, les notes liées ne sont pas toutes consécutives. Il faut alors assigner à la propriété `tieWaitForNote` la valeur *vrai* ('t' pour 'true'). Cette même méthode peut servir, par exemple, à lier un trémolo à un accord.

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



Il est possible de graver manuellement les liaisons de tenue, en modifiant la propriété `tie-configuration`. Pour chaque paire, le premier nombre indique la distance à la portée, en espaces de portée, et le second la direction (1 pour haut, −1 pour bas).

```
<c e g>2~ <c e g> |
\override TieColumn #'tie-configuration =
  #'((0.0 . 1) (-2.0 . 1) (-4.0 . 1))
<c e g>~ <c e g> |
```

Commandes prédéfinies

`\tieUp`, `\tieDown`, `\tieNeutral`, `\tieDotted`, `\tieDashed`, `\tieSolid`.

Voir aussi

Glossaire musical : Section "liaison de prolongation" dans *Glossaire*, Section "laissez vibrer" dans *Glossaire*.

Dans ce manuel : Section 1.2.3.5 [Découpage automatique des notes], page 33.

Référence du programme : Section "Tie" dans *Référence des propriétés internes*, Section "TieColumn" dans *Référence des propriétés internes*, Section "LaissezVibrerTie" dans *Référence des propriétés internes*, Section "LaissezVibrerTieColumn" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

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

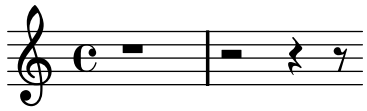
Le changement de clé ou d'octave pendant une liaison de tenue produit un résultat indéfini. Dans ces cas-là, il est préférable d'utiliser un legato.

1.2.2 Écriture des silences

1.2.2.1 Silences

Les silences sont écrits comme des notes avec le nom de note `r`.

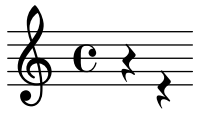
```
r1 r2 r4 r8
```



Les pauses d'une mesure complète, qui sont placées au centre de la mesure, doivent être entrées comme des mesures de silence. Elles peuvent être utilisées pour une seule mesure comme pour plusieurs, et leur utilisation est expliquée dans la section Section 1.2.2.3 [Silences valant une mesure], page 26.

Pour spécifier explicitement la position verticale d'un silence, écrivez une note suivie de `\rest`. Un silence sera placé à la position où serait imprimée la note.

```
a'4\rest d'4\rest
```



Cela rend plus facile la mise en place de la musique polyphonique, puisque le formateur automatique de collision des silences laissera ces silences tranquilles.

Voir aussi

Référence du programme : Section "Rest" dans *Référence des propriétés internes*.

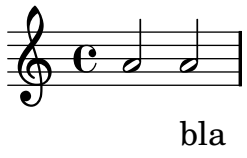
1.2.2.2 Silences invisibles

Un silence invisible — que l'on pourrait appeler un « saut » — peut être entré comme une note avec le nom de note "`s`" ou avec `\skip` *`durée`*

```
a4 a4 s4 a4 \skip 1 a4
```

La syntaxe `s` est seulement disponible pour les modes d'entrée de notes et d'accords. Dans d'autres situations, pour l'entrée de paroles par exemple, vous devrez utiliser la commande `\skip`.

```
<<
  \relative { a'2 a2 }
  \new Lyrics \lyricmode { \skip 2 bla2 }
>>
```



La commande de saut génère simplement une case musicale vide. Elle ne produit rien sur la partition, pas même un symbole transparent. Le code de saut `s` crée tout de même une Section “Staff” dans *Référence des propriétés internes* et une Section “Voice” dans *Référence des propriétés internes* lorsque nécessaire, tout comme les commandes de note et de silence. Ainsi, le code suivant aboutit à une portée vide.

```
{ s4 }
```

Le fragment `{ \skip 4 }` produirait une page vide.

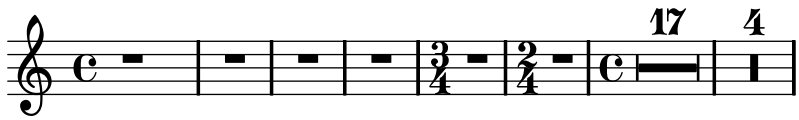
Voir aussi

Référence du programme : Section “SkipMusic” dans *Référence des propriétés internes*.

1.2.2.3 Silences valant une mesure

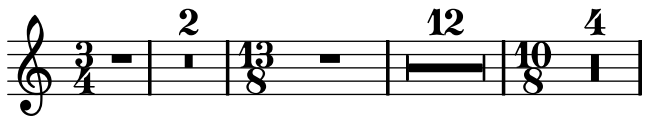
Un silence valant une ou plusieurs mesures entières s’entre avec un ‘`R`’ majuscule. Ceci ne peut être utile que pour une mesure complètement vide, et pour générer des parties séparées : ce silence sera alors répété sur autant de mesures que nécessaire, ou bien imprimé une seule fois. La répétition est contrôlée par la propriété `Score.skipBars`. Au cas où ce commutateur est défini comme vrai (lettre `##t` pour ‘true’), les mesures vides ne seront pas répétées, et le nombre exact de mesures sera ajouté.

```
\time 4/4 r1 | R1 | R1*2 \time 3/4 R2. \time 2/4 R2 \time 4/4
\set Score.skipBars = ##t R1*17 R1*4
```



Le `1` de `R1` est le même que celui utilisé pour la durée des notes. Vous devrez donc, si le morceau n’est pas à 4/4, stipuler un autre durée, qui pourra contenir des points d’augmentation ou être libellé sous forme de fraction :

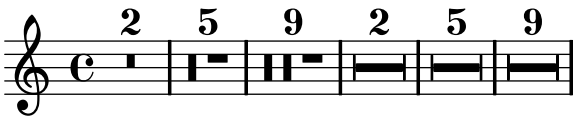
```
\set Score.skipBars = ##t
\time 3/4
R2. | R2.*2
\time 13/8
R1*13/8
R1*13/8*12 |
\time 10/8 R4*5*4 |
```



Un `R` qui s’étend sur une seule mesure s’imprime tantôt comme une pause, tantôt comme une brève, et sera centré sur la mesure quelle que soit la métrique.

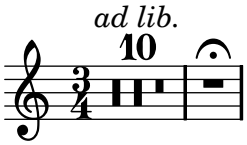
Dans le cas où ce silence ne dure que quelques mesures, LilyPond imprime sur la portée des « silences d’église », simple suite de rectangles. La propriété `MultiMeasureRest.expand-limit` permet d’obtenir un silence unique.

```
\set Score.skipBars = ##t
R1*2 | R1*5 | R1*9
\override MultiMeasureRest #'expand-limit = 1
R1*2 | R1*5 | R1*9
```



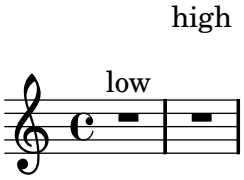
Vous pouvez aussi ajouter du texte à un silence multi-mesures en utilisant la syntaxe *note-markup* (cf. Section 1.8.2 [Mise en forme du texte], page 99). La variable `\fermataMarkup` permet d'ajouter un point d'orgue.

```
\set Score.skipBars = ##t
\time 3/4
R2.*10^\markup { \italic "ad lib." }
R2.^\fermataMarkup
```



Attention ! c'est `MultiMeasureRestText` qui créera le texte, et non `TextScript`.

```
\override TextScript #'padding = #5
R1^"low"
\override MultiMeasureRestText #'padding = #5
R1^"high"
```



Pour aligner votre texte sur le début de la mesure, rattachez-le à un silence invisible de longueur zéro comme ceci :

```
s1*0^"Allegro"
R1*4
```

Voir aussi

Référence du programme : Section "MultiMeasureRestMusic" dans *Référence des propriétés internes*, Section "MultiMeasureRest" dans *Référence des propriétés internes*.

L'objet de rendu Section "MultiMeasureRestNumber" dans *Référence des propriétés internes* traite les nombres, et Section "MultiMeasureRestText" dans *Référence des propriétés internes* le texte ajouté par l'utilisateur.

Problèmes connus et avertissements

Vous ne pouvez pas utiliser de doigtés (p.ex. `R1-4`) pour positionner des nombres au dessus d'un silence multi-mesures, ni modifier la hauteur.

Condenser plusieurs silences en un unique silence multi-mesures ne peut être automatisé. Les silences multi-mesures peuvent générer des collisions avec d'autres silences.

Pensez à indiquer explicitement la durée de la note qui suit un silence multi-mesures, car elle sera par défaut égale à la durée totale des mesures à compter. Ainsi, dans l'exemple ci-après, les deux do dièses vaudront chacun quatre mesures à 4/4.

```
R1*4 cis cis
```

Lorsque `skipBars` est activé, le résultat semblera correct, mais la numérotation des mesures sera suspendue.

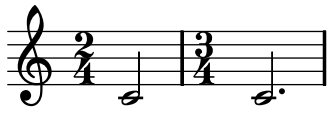
1.2.3 Gravure du rythme

1.2.3.1 Métrique

Le chiffre de mesure indique le mètre d'une pièce : une alternance régulière de temps forts et de temps faibles. Il est indiqué par une fraction au début de la portée.

Le chiffre de mesure est réglé par la commande `\time`.

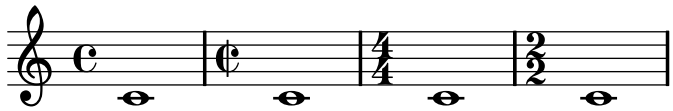
```
\time 2/4 c'2 \time 3/4 c'2.
```



Propriétés couramment modifiées

Le symbole imprimé peut être modifié avec la propriété `style`. En la réglant sur `#'()`, une fraction sera utilisée pour les chiffres de mesure 4/4 et 2/2.

```
\time 4/4 c'1
\time 2/2 c'1
\override Staff.TimeSignature #'style = #'()
\time 4/4 c'1
\time 2/2 c'1
```



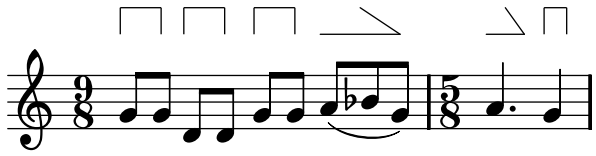
Il y a bien d'autres options pour sa mise en place. Voir Section 2.8.2.6 [Métriques anciennes], page 153 pour plus d'exemples.

`\time` définit les propriétés `timeSignatureFraction`, `beatLength` et `measureLength` dans le contexte `Timing`, qui en principe est assimilé à Section "Score" dans *Référence des propriétés internes*. La propriété `measureLength` détermine où des barres de mesure doivent être insérées, et comment les groupements de notes doivent être gérés. La modification de la valeur de `timeSignatureFraction` donne également lieu à l'impression d'un symbole.

Plus d'options sont accessibles au moyen de la fonction Scheme `set-time-signature`. De concert avec le Section "Measure_grouping_engraver" dans *Référence des propriétés internes*, elle crée les signes de Section "MeasureGrouping" dans *Référence des propriétés internes*, qui

facilitent la lecture de musiques modernes, complexes rythmiquement. Dans l'exemple suivant, les mesures à 9/8 sont subdivisées en `(2 2 2 3)`, ce qui est donné comme argument à la commande `set-time-signature`, en troisième position.

```
\score {
  \relative c'' {
    #(set-time-signature 9 8 '(2 2 2 3))
    g8[ g] d[ d] g[ g] a8[( bes g]) |
    #(set-time-signature 5 8 '(3 2))
    a4. g4
  }
  \layout {
    \context {
      \Staff
      \consists "Measure_grouping_engraver"
    }
  }
}
```



Voir aussi

Référence du programme : Section "TimeSignature" dans *Référence des propriétés internes* et Section "Timing_translator" dans *Référence des propriétés internes*.

Exemples : Section "Rhythms" dans *Exemples de code*.

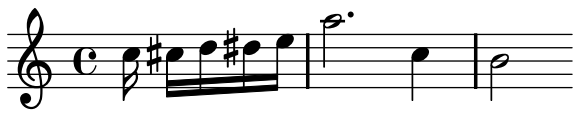
Problèmes connus et avertissements

Le groupement automatique des ligatures n'utilise pas les groupements spécifiés par `set-time-signature`.

1.2.3.2 Levées

Les mesures incomplètes, telles que les anacrouses ou levées, doivent être entrées avec la commande

```
\partial 16*5 c16 cis d dis e | a2. c,4 | b2
```



La syntaxe de cette commande est

`\partial` *`durée`*

`durée` étant la valeur rythmique devant être ajoutée avant la mesure suivante.

Le programme traduit cette commande en

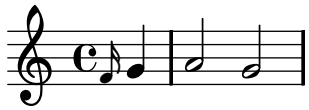
`\set Timing.measurePosition = -durée`

La propriété `measurePosition` contient un nombre rationnel qui indique, à ce point précis, où l'on en est de la mesure. Notez qu'il peut s'agir d'un nombre négatif ; `\partial 4` signifie, pour le programme : « Dans cette mesure, il reste juste une noire ».

Problèmes connus et avertissements

Cette commande ne prend pas en compte les notes d'ornement ou appoggiatures au début de la musique. Lorsqu'un morceau commence par une levée et avec des petites notes, la commande `\partial` devrait être placée après celles-ci.

```
\grace f16
\partial 4
g4
a2 g2
```

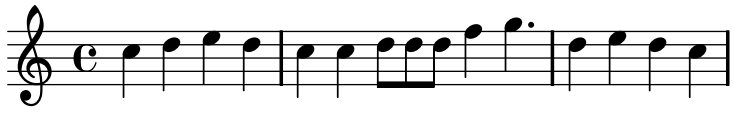


`\partial` n'est destiné à être utilisé qu'en début de pièce. Si on l'utilise ailleurs qu'au début, des messages d'erreurs peuvent s'afficher.

1.2.3.3 Musique sans métrique

Les barres de mesure et les numéros de mesure sont calculés automatiquement, ce qui n'est pas souhaitable dans le cas d'une musique non mesurée — les cadences, par exemple. Les commandes `\cadenzaOn` et `\cadenzaOff` permettent de désactiver et de rétablir la métrique automatique.

```
c4 d e d
\cadenzaOn
c4 c d8 d d f4 g4.
\cadenzaOff
\bar "|"
d4 e d c
```



Problèmes connus et avertissements

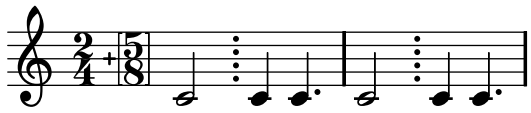
LilyPond ne change de ligne ou de page qu'au niveau des barres de mesure. Si votre musique non mesurée dure plus d'une ligne, il vous faudra insérer des barres de mesure invisibles, pour indiquer à quels endroit un saut de ligne peut intervenir.

```
\bar ""
```

1.2.3.4 Notation polymétrique

LilyPond ne gère pas les métriques composites de manière explicite, mais on peut contourner ce problème. Dans l'exemple suivant, l'indicateur de métrique est obtenu grâce à une étiquette textuelle. Cette étiquette vient s'insérer dans l'objet graphique (*grob*) Section "TimeSignature" dans *Référence des propriétés internes*.

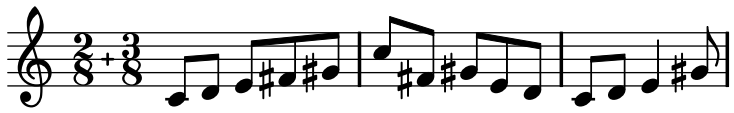
```
% Mise en évidence de la décomposition de 9/8 en 2/4 + 5/8
tsEtiquette = \markup {
  \override #'(baseline-skip . 2) \number {
    \column { "2" "4" }
    \vcenter "+"
    \bracket \column { "5" "8" }
  }
}

{
  \override Staff.TimeSignature #'stencil =
    #ly:text-interface::print
  \override Staff.TimeSignature #'text = #tsEtiquette
  \time 9/8
  c'2 \bar ":" c'4 c'4.
  c'2 \bar ":" c'4 c'4.
}
```



Compound time signatures

Odd 20th century time signatures (such as "5/8") can often be played as compound time signatures (e.g. "3/8 + 2/8"), which combine two or more inequal metrics. LilyPond can make such music quite easy to read and play, by explicitly printing the compound time signatures and adapting the automatic beaming behavior. (Graphic measure grouping indications can also be added; see the appropriate snippet in this database.)

```
#(define (compound-time one two num)
  (markup #:override '(baseline-skip . 0) #:number
   (#:line ((#:column (one num)) #:vcenter "+" (#:column (two num))))
  ))

\relative {
  \override Staff.TimeSignature #'stencil = #ly:text-interface::print
  \override Staff.TimeSignature #'text = #(compound-time "2" "3" "8")
  \time 5/8
  #(override-auto-beam-setting '(end 1 8 5 8) 1 4)
  c8 d e fis gis
  c8 fis, gis e d
  c8 d e4 gis8
}
```

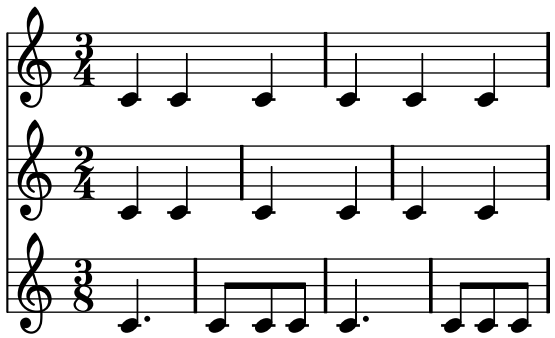


Il arrive aussi que chaque portée ait sa propre métrique. Vous y parviendrez en déplaçant le Section “Timing_translator” dans *Référence des propriétés internes* dans le contexte Section “Staff” dans *Référence des propriétés internes*.

```
\layout {
  \context { \Score
     \remove "Timing_translator"
     \remove "Default_bar_line_engraver"
  }
  \context {
    \Staff
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  }

}
```

Maintenant, chacune des portées dispose de sa propre métrique.

```
<<
  \new Staff {
    \time 3/4
    c4 c c | c c c |
  }
  \new Staff {
    \time 2/4
    c4 c | c c | c c
  }
  \new Staff {
    \time 3/8
    c4. c8 c c c4. c8 c c
  }
>>
```



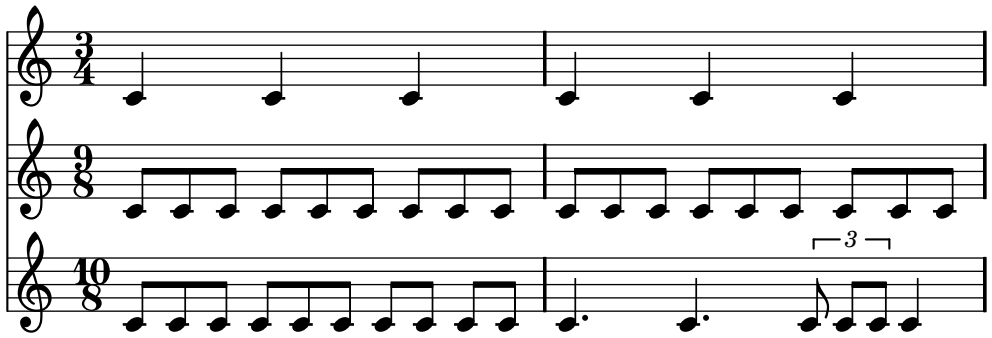
Une autre forme de notation polymétrique consiste dans le fait que des notes aient une durée relative différente selon la portée.

Vous pouvez créer une telle notation en définissant une métrique commune à toutes les portées, que vous proratiserez manuellement selon le cas en utilisant `timeSignatureFraction` pour obtenir la division adéquate pour chaque portée. Les durées, dans chacune des portées, seront alors échelonnées par rapport à la métrique commune. L'échelle de représentation se règle avec `\scaleDurations` — qui fonctionne comme `\times`, sans toutefois créer de crochet. La syntaxe appropriée est :

`\scaleDurations #'(`*`numérateur . dénominateur`*`) `*`exprmusicale`*

L'exemple suivant utilise parallèlement des mesures à 3/4, 9/8 et 10/8. Pour la deuxième portée, les durées sont multipliées par 2/3, de telle sorte que 2/3 * 9/8 = 3/4 ; pour la troisième, elles sont multipliées par 3/5, de telle sorte que 3/5 * 10/8 = 3/4.

```
\relative c' { <<
  \new Staff {
    \time 3/4
    c4 c c | c c c |
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = #'(9 . 8)
    \scaleDurations #'(2 . 3)
      \repeat unfold 6 { c8[ c c] }
  }
  \new Staff {
    \time 3/4
    \set Staff.timeSignatureFraction = #'(10 . 8)
    \scaleDurations #'(3 . 5) {
      \repeat unfold 2 { c8[ c c] }
      \repeat unfold 2 { c8[ c] }
      | c4. c4. \times 2/3 { c8 c c } c4
    }
  }
>> }
```



Voir aussi

Exemples : Section “Rhythms” dans *Exemples de code*.

Problèmes connus et avertissements

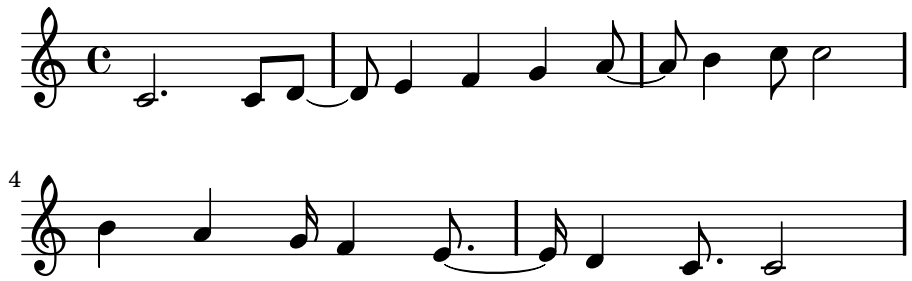
L’utilisation de métriques différentes en parallèle entraine un alignement vertical. De ce fait, les barres de mesure ont tendance à fausser l’espacement régulier.

1.2.3.5 Découpage automatique des notes

On peut convertir automatiquement les notes longues en notes liées. Il faut pour cela remplacer le graveur Section “Note_heads_engraver” dans *Référence des propriétés internes* par le graveur Section “Completion_heads_engraver” dans *Référence des propriétés internes*. Dans les exemples suivants, les notes dépassant de la mesure sont divisées et liées.

```
\new Voice \with {
  \remove "Note_heads_engraver"
  \consists "Completion_heads_engraver"
```

```
} {
  c2. c8 d4 e f g a b c8 c2 b4 a g16 f4 e d c8. c2
}
```



Ce graveur divise toutes les notes qui sortent de la mesure, et insère des liaisons de prolongation. Une utilisation posible consiste à déboguer des partitions complexes : si les mesures ne sont pas entièrement remplies, alors les liaisons de prolongation montrent exactement la durée des décalages de mesure.

Si vous voulez permettre un saut de ligne aux barres de mesure où Section “Completion_heads_engraver” dans *Référence des propriétés internes* divise les notes, vous devez aussi enlever Section “Forbid_line_break_engraver” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Bien que toutes les durées — particulièrement celles contenant des nolets — ne puissent pas être représentées exactement avec des notes normales et des points, le graveur n’insèrera pas de nolets.

`Completion_heads_engraver` affecte seulement les notes, il ne divise pas les silences.

Voir aussi

Référence du programme : Section “Completion_heads_engraver” dans *Référence des propriétés internes*.

1.2.3.6 Gravure de lignes rythmiques

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

```
\new RhythmicStaff {
  \time 4/4
  c4 e8 f g2 | r4 g r2 | g1:32 | r1 |
}
```

Voir aussi

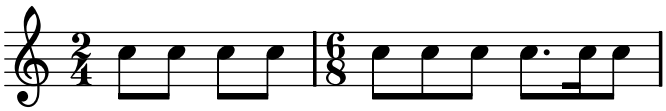
Référence du programme : Section “RhythmicStaff” dans *Référence des propriétés internes*.

1.2.4 Barres de ligature

1.2.4.1 Barres de ligature automatiques

LilyPond décide automatiquement de la manière de grouper les notes et d'imprimer les ligatures.

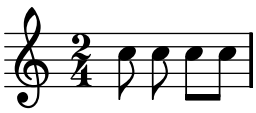
```
\time 2/4 c8 c c c \time 6/8 c c c c8. c16 c8
```



Lorsque ce comportement automatisé n'est pas satisfaisant, on peut définir des groupements manuellement — voir section suivante — ou personnaliser les groupements automatiques — voir Section 1.2.4.2 [Définition des règles de ligature automatique], page 35.

La commande `\noBeam` peut servir à empêcher des notes individuelles d'être rattachées aux autres.

```
\time 2/4 c8 c\noBeam c c
```



Voir aussi

Référence du programme : Section "Beam" dans *Référence des propriétés internes*.

1.2.4.2 Définition des règles de ligature automatique

Dans les métriques courantes, les ligatures automatiques peuvent commencer sur n'importe quelle note, mais ne peuvent se terminer qu'à certains points précis dans la mesure : sur une pulsation, ou après toute durée spécifiée par les propriétés nommées `autoBeamSettings`. Ces propriétés consistent en une liste de règles relatives au début ou à la fin des ligatures. Par défaut, elles sont définies dans le fichier '`scm/auto-beam.scm`'.

On peut ajouter à cette liste une nouvelle règle, au moyen de la commande

```
#(override-auto-beam-setting '(extrémité p q n m) a b [contexte])
```

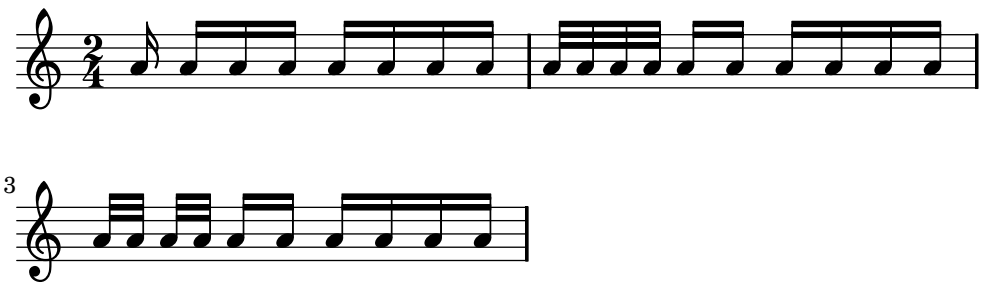
- `extrémité` désigne le début (`begin`) ou la fin (`end`) de la ligature.
- `p/q` désigne la valeur rythmique de la note que l'on veut affecter, ou de la plus brève des notes concernées. Si cette règle doit s'appliquer à toutes les ligatures, remplacez `p` et `q` par des étoiles « `*` ».
- `n/m` est le chiffre de mesure dans lequel la règle doit s'appliquer. Si celle-ci doit s'appliquer dans toutes les métriques, remplacez `n` et `m` par des étoiles « `*` ».
- `a/b` est l'emplacement, dans la mesure, auquel les ligatures doivent débuter ou s'achever (suivant la valeur 'extrémité' que nous venons de voir).
- `contexte` est un argument facultatif, qui détermine le contexte dans lequel la règle doit s'appliquer. Par défaut, il s'agit de `'Voice`. `#(score-override-auto-beam-setting '(A B C D) E F)` équivaut à `#(override-auto-beam-setting '(A B C D) E F 'Score)`.

Par exemple, si l'on veut que les ligatures s'achèvent toujours après la première noire :

```
#(override-auto-beam-setting '(end * * * *) 1 4)
```

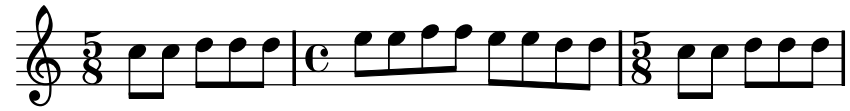
On peut obliger une règle de ligatures à ne s'appliquer qu'à des groupes dont la note la plus brève est d'une valeur précise :

```
\time 2/4
#(override-auto-beam-setting '(end 1 16 * *) 1 16)
a16 a a a a a a |
a32 a a a a16 a a a a |
#(override-auto-beam-setting '(end 1 32 * *) 1 16)
a32 a a a a16 a a a a |
```



On peut obliger une règle de ligatures à ne s'appliquer que pour un chiffre de mesure précis :

```
\time 5/8
#(override-auto-beam-setting '(end * * 5 8) 2 8)
c8 c d d d
\time 4/4
e8 e f f e e d d
\time 5/8
c8 c d d d
```

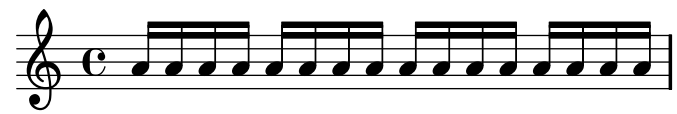


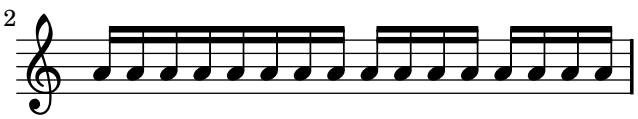
Enfin, on peut désactiver une règle de ligatures au moyen du réglage suivant :

```
#(revert-auto-beam-setting '(extrémité p q n m) a b [contexte])
```

`extrémité`, `p`, `q`, `n`, `m`, `a`, `b` et `contexte` étant les mêmes que plus haut. Il est même possible de désactiver des règles que l'on n'a pas explicitement créées : les règles par défaut, qui se trouvent dans le fichier '`scm/auto-beam.scm`'.

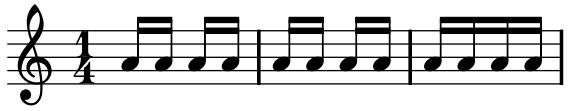
```
\time 4/4
a16 a a a a a a a a a a a a a a a a
#(revert-auto-beam-setting '(end 1 16 4 4) 1 4)
a16 a a a a a a a a a a a a a a a a
```





La commande `revert-auto-beam-setting` requiert exactement les mêmes arguments que la règle d'origine. En d'autres termes, les étoiles ne seront pas prises en compte ici.

```
\time 1/4
#(override-auto-beam-setting '(end 1 16 1 4) 1 8)
a16 a a a
#(revert-auto-beam-setting '(end 1 16 * *) 1 8) % ceci ne désactive pas la règle !
a a a a
#(revert-auto-beam-setting '(end 1 16 1 4) 1 8) % ceci marche
a a a a
```



Si, dans une mesure à 5/4, l'on veut que les ligatures soient regroupées temps par temps, il est nécessaire d'indiquer toutes les terminaisons de ligatures.

```
#(override-auto-beam-setting '(end * * * *) 1 4 'Staff)
#(override-auto-beam-setting '(end * * * *) 1 2 'Staff)
#(override-auto-beam-setting '(end * * * *) 3 4 'Staff)
#(override-auto-beam-setting '(end * * * *) 5 4 'Staff)
...
```

La même syntaxe peut servir à définir les points de départ des ligatures. Dans l'exemple suivant, les ligatures automatiques ne peuvent se terminer que sur une noire pointée.

```
#(override-auto-beam-setting '(end * * * *) 3 8)
#(override-auto-beam-setting '(end * * * *) 1 2)
#(override-auto-beam-setting '(end * * * *) 7 8)
```

Dans une mesure à 4/4, cela implique que les ligatures ne peuvent se terminer que sur la troisième croche, ou sur le quatrième temps (après la valeur de deux fois trois croches).

Si une ligature se fait de manière inattendue, pensez à vérifer les règles automatiques dans le fichier '`scm/auto-beam.scm`' pour rechercher d'éventuels conflits, dans la mesure ou les règles par défaut s'ajoutent à vos propres règles. Il est alors nécessaire de désactiver toute règle par défaut conduisant à des ligatures indésirables.

Ainsi, pour obtenir des ligatures en groupes de `(3 4 3 2)` croches, dans une mesure à 12/8, il faudra préalablement utiliser :

```
%%% annulons les réglages par défaut relatifs à 12/8, dans scm/auto-beam.scm
#(revert-auto-beam-setting '(end * * 12 8) 3 8)
#(revert-auto-beam-setting '(end * * 12 8) 3 4)
#(revert-auto-beam-setting '(end * * 12 8) 9 8)

%%% puis ajoutons nos propres règles
#(override-auto-beam-setting '(end 1 8 12 8) 3 8)
#(override-auto-beam-setting '(end 1 8 12 8) 7 8)
#(override-auto-beam-setting '(end 1 8 12 8) 10 8)
```

Si des ligatures sont utilisées dans les paroles d'une chanson (pour indiquer des mélismes), les ligatures automatiques doivent être désactivées, avec `\autoBeamOff`.

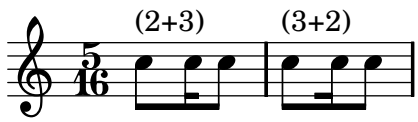
Commandes prédéfinies

`\autoBeamOff`, `\autoBeamOn`.

Propriétés couramment modifiées

Les groupes de notes reliées par les ligatures peuvent être spécifiés au moyen de la propriété `beatGrouping`.

```
\time 5/16
\set beatGrouping = #'(2 3)
c8[^"(2+3)" c16 c8]
\set beatGrouping = #'(3 2)
c8[^"(3+2)" c16 c8]
```



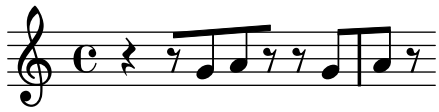
Problèmes connus et avertissements

Si une partition se termine alors qu'une ligature automatique est restée inachevée, cette dernière ligature ne sera pas imprimée du tout. C'est également valable dans le cas d'une musique polyphonique, saisie avec la syntaxe `<< ... \\ ... >>`, où une voix se terminerait sans que la dernière ligature soit achevée.

1.2.4.3 Barres de ligature manuelles

Dans certaines situations, il peut s'avérer nécessaire de supplanter l'algorithme de groupement automatique des notes, par exemple pour prolonger une ligature par-dessus un silence ou une barre de mesure. Le début et la fin de la ligature sont alors indiqués par `[` et `]`.

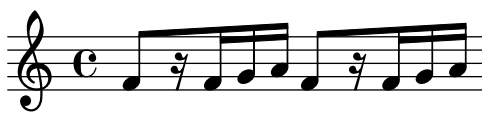
```
{
  r4 r8[ g' a r8] r8 g[ | a] r8
}
```



Propriétés couramment modifiées

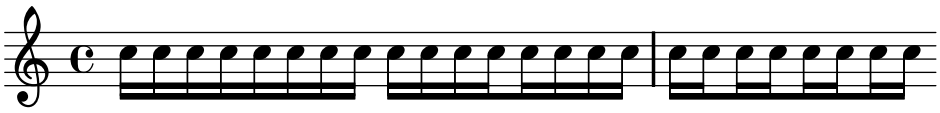
LilyPond peut déterminer automatiquement les sous-groupes à l'intérieur d'un groupement de notes, bien que le résultat ne soit pas toujours optimal. Les propriétés `stemLeftBeamCount` et `stemRightBeamCount` permettent alors d'ajuster ce comportement. Lorsque l'une ou l'autre de ces propriétés est définie, elle ne s'applique qu'une seule fois, après quoi sa définition est effacée.

```
{
   f8[ r16
      f g a]
   f8[ r16
   \set stemLeftBeamCount = #1
      f g a]
}
```



La propriété `subdivideBeams` sert à grouper les double-croches ou les valeurs plus brèves pulsation par pulsation, la pulsation étant définie par la propriété `beatLength`.

```
c16[ c c c c c c]
\set subdivideBeams = ##t
c16[ c c c c c c]
\set Score.beatLength = #(ly:make-moment 1 8)
c16[ c c c c c c]
```



Pour plus d'information sur `make-moment`, voir Section 1.2.6.3 [Gestion du temps], page 49.

Lorsqu'une ligature franchit une barre de mesure, le saut de ligne est en principe interdit à cet endroit. Ce comportement peut être outrepassé en définissant `breakable`.

LilyPond insère automatiquement des ligatures coudées — certaines notes vers le haut, d'autres vers le bas — lorsqu'il détecte un espace important entre des têtes de notes. Ce comportement peut être changé par l'intermédiaire de l'objet `auto-knee-gap`

Problèmes connus et avertissements

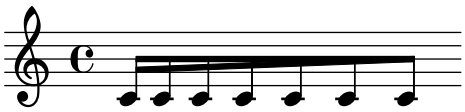
Les ligatures coudées à cheval sur deux portées ne peuvent être employées en même temps que des portées invisibles. Voir [Masquage de portées], page 77.

Les ligatures peuvent entrer en collision avec des symboles entourant les notes, contrairement aux textes ou aux altérations.

1.2.4.4 Liens de croches en soufflet

Les ligatures en soufflet s'obtiennent en définissant la propriété `grow-direction` d'une barre de ligature. La fonction `\featherDurations` sert à ajuster la durée des notes.

```
\override Beam #'grow-direction = #LEFT
\featherDurations #(ly:make-moment 5 4)
{
  c16[ c c c c c]
}
```



Problèmes connus et avertissements

La commande `\featherDurations` ne permet de traiter que de très courts extraits.

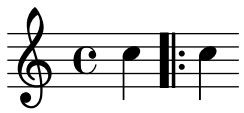
1.2.5 Barres de mesure

1.2.5.1 Barres de mesure

Les barres de mesures délimitent les mesures, mais peuvent aussi indiquer une reprise. En principe, elles sont insérées automatiquement, et les sauts de ligne ne peuvent avoir lieu qu'au niveau de ces barres.

Il est possible de forcer l'impression d'une barre de mesure spéciale, avec la commande `\bar` :

```
c4 \bar "|:" c4
```



Les styles de barres de mesure disponibles sont

En plus de cela, on peut demander `"||:"`, qui équivaut à `"|:"`, mais qui donnera, en cas de saut de ligne, une double barre en fin de ligne, et une barre de reprise au début de la ligne suivante.

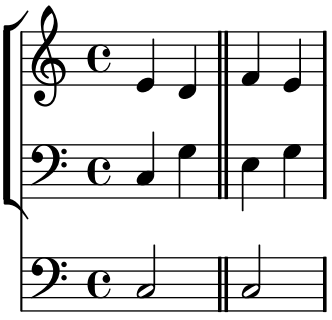
Il est possible d'autoriser un saut de ligne même s'il n'y a pas de barre de mesure visible, en utilisant :

```
\bar ""
```

Ceci insérera une barre de mesure invisible, et permettra de sauter de ligne à cet endroit, sans incrémenter le numéro de mesure.

Dans une partition comprenant plusieurs portées, la commande `\bar` placée sur une portée s'applique automatiquement à toutes les portées. Les barres de mesure que l'on obtient alors sont d'un seul tenant sur les portées d'un `StaffGroup`, d'un `PianoStaff` ou d'un `GrandStaff`.

```
<<
  \new StaffGroup <<
    \new Staff {
      e'4 d'
      \bar "||"
      f' e'
    }
    \new Staff { \clef bass c4 g e g }
  >>
  \new Staff { \clef bass c2 c2 }
>>
```



Propriétés couramment modifiées

La commande `\bar` *bartype* sert de raccourci pour `\set Timing.whichBar =` *bartype*. Dès que l'on définit `whichBar`, une barre de mesure est créée selon le style défini.

Dès que la propriété `whichBar` est définie, une barre de mesure est créée. À chaque début de mesure, elle prend la valeur de `Timing.defaultBarType`. La valeur de `repeatCommands` sert à remplacer les barres de mesure par défaut.

Nous vous invitons à utiliser `\repeat` pour indiquer les reprises. Voyez à ce sujet Section 1.4 [Répétitions et reprises], page 60.

Voir aussi

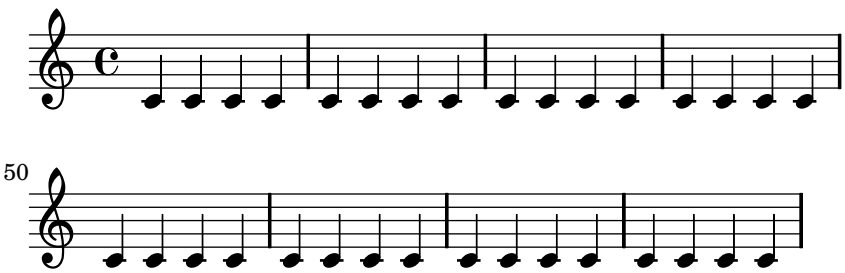
Dans ce manuel : Section 1.4 [Répétitions et reprises], page 60, [Regroupement de portées], page 74.

Référence du programme : Section "BarLine" dans *Référence des propriétés internes* (faisant partie du contexte Section "Staff" dans *Référence des propriétés internes*), Section "SpanBar" dans *Référence des propriétés internes* (sur plusieurs portées).

1.2.5.2 Numéros de mesure

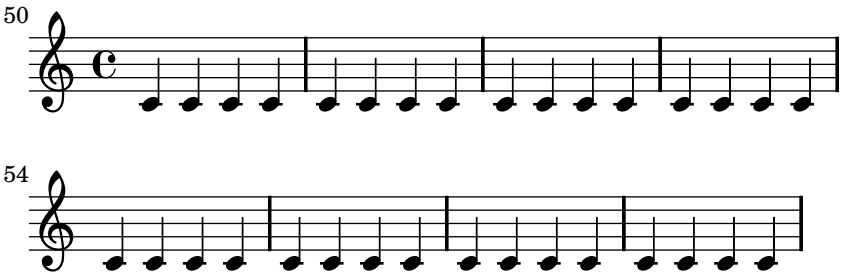
Les numéros de mesure sont imprimés par défaut à chaque début de ligne. Ce nombre est stocké par la propriété `currentBarNumber` qui sera mis à jour à chaque mesure.

```
\repeat unfold 4 {c4 c c c} \break
\set Score.currentBarNumber = #50
\repeat unfold 4 {c4 c c c}
```



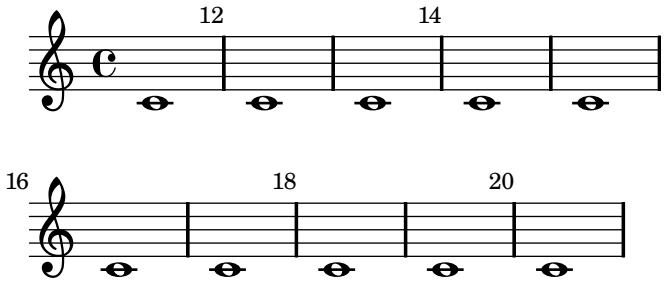
L'impression d'un numéro de mesure ne peut intervenir que s'il y a une barre. Aussi, pour pouvoir le faire au début d'un morceau, devrez-vous ajouter une barre vide :

```
\set Score.currentBarNumber = #50
\bar ""
\repeat unfold 4 {c4 c c c} \break
\repeat unfold 4 {c4 c c c}
```



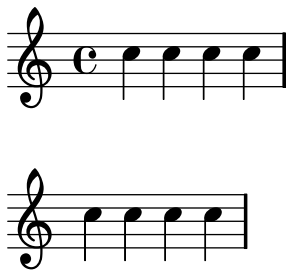
Vous pouvez imprimer un numéro de mesure à intervalles réguliers plutôt qu'en tête de chaque ligne. C'est ce qu'illustre l'exemple suivant.

```
\override Score.BarNumber #'break-visibility = #'#(#f #t #t)
\set Score.currentBarNumber = #11
\bar ""  % Le numéro de la première mesure sera affiché
% Affichage du numéro toutes les deux mesures
\set Score.barNumberVisibility = #(every-nth-bar-number-visible 2)
c1 c c c c
\break
c c c c c
```



Désactiver le graveur concerné — `Bar_number_engraver` — donnera une partition sans numéros de mesure.

```
\layout {
  \context {
    \Score
    \remove "Bar_number_engraver"
  }
}
\relative c''{
c4 c c c \break
c4 c c c
}
```



Voir aussi

Référence du programme : Section “BarNumber” dans *Référence des propriétés internes*.

Exemples : Section “Staff notation” dans *Exemples de code*.

Problèmes connus et avertissements

Les numéros de mesure peuvent entrer en collision avec les crochets de Section “StaffGroup” dans *Référence des propriétés internes*. La propriété `padding` — décalage — de l’objet Section “BarNumber” dans *Référence des propriétés internes* permet alors d’ajuster leur positionnement.

1.2.5.3 Vérification des limites et numéros de mesure

Les tests de limites de mesure (ou tests de mesure) aident à détecter les erreurs dans les durées. Un test de mesure s'écrit avec une barre verticale, '`|`'. Lors du traitement, elle doit correspondre à une barre de mesure. Sinon, un avertissement est émis. Dans l'exemple suivant, le deuxième test de mesure signale une erreur.

```
\time 3/4 c2 e4 | g2 |
```

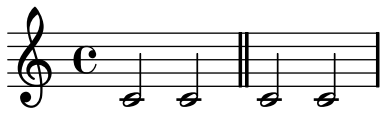
Le test de mesure peut être aussi utilisé dans les paroles, par exemple :

```
\lyricmode {
 \time 2/4
 Twin -- kle | Twin -- kle
}
```

Des durées incorrectes font échouer les tests de mesure, et peuvent souvent mettre la partition sens dessus dessous, particulièrement s'il s'agit de musique polyphonique. Vérifier les tests de mesure qui ont échoué et les durées incorrectes est un bon moyen de commencer à corriger sa partition.

Il est aussi possible d'attribuer une autre valeur au symbole `|`, en assignant une expression musicale à `pipeSymbol`,

```
pipeSymbole = \bar "||"

{ c'2 c' | c'2 c' }
```



Lorsque l'on recopie de longues pièces, il peut être utile de vérifier que les numéros de mesures de LilyPond correspondent à l'original que l'on recopie. Cela se fait avec `\barNumberCheck`. Par exemple,

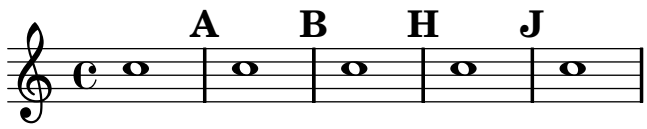
```
\barNumberCheck #123
```

affiche un avertissement lors du traitement si le numéro de mesure à ce point (variable `currentBarNumber`) n'est pas 123.

1.2.5.4 Indications de repère

Indiquer un repère s'obtient grâce à la commande `\mark`.

```
c1 \mark \default
c1 \mark \default
c1 \mark #8
c1 \mark \default
c1 \mark \default
```



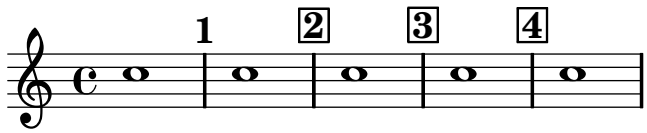
La lettre 'I' n'est pas utilisée, conformément aux usages de la gravure. Cependant, vous pourrez intégrer la lettre 'I' en utilisant

```
\set Score.markFormatter = #format-mark-alphabet
```

Lorsque vous utilisez `\mark \default`, le repère s'incrémente automatiquement ; toutefois donner un nombre en argument permet de spécifier manuellement le repère en question. La valeur à utiliser est enregistrée dans la propriété `rehearsalMark`.

Le style du repère est déterminé par la propriété `markFormatter`. Il s'agit d'une fonction qui prend en arguments le repère en cours (un entier) ainsi que le contexte en cours, et retournera un objet de type étiquette. Dans l'exemple qui suit, `markFormatter` est réglé pour une procédure type. Quelques mesure plus loin, son comportement est modifié pour imprimer un repère encadré.

```
\set Score.markFormatter = #format-mark-numbers
c1 \mark \default
c1 \mark \default
\set Score.markFormatter = #format-mark-box-numbers
c1 \mark \default
c1 \mark \default
c1
```



Le fichier '`scm/translation-functions.scm`' comporte les définitions de `format-mark-numbers` (comportement par défaut), `format-mark-box-numbers`, `format-mark-letters` et `format-mark-box-letters`. Vous pouvez vous en inspirer pour d'autres fonctions de formatage.

`format-mark-barnumbers`, `format-mark-box-barnumbers` et `format-mark-circle-barnumbers` permettent d'imprimer le numéro de mesure au lieu des compteurs alphabétique ou numérique.

On peut aussi spécifier manuellement une marque de repère :

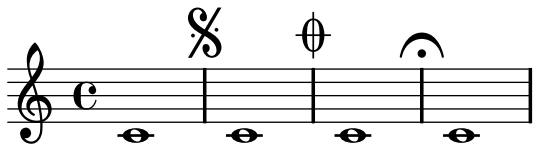
```
\mark "A1"
```

`Score.markFormatter` sera sans effet sur des repères ainsi définis. Un `\markup` peut néanmoins s'utiliser en argument.

```
\mark \markup{ \box A1 }
```

Un `\mark` peut contenir un glype musical tel que le signe *segno*.

```
c1 \mark \markup { \musicglyph #"scripts.segno" }
c1 \mark \markup { \musicglyph #"scripts.coda" }
c1 \mark \markup { \musicglyph #"scripts.ufermata" }
c1
```



Pour connaître les différents symboles accessibles par `\musicglyph`, consultez Section B.4 [La fonte Feta], page 196.

Pour affiner le positionnement des repères, veuillez vous référer à [Indications textuelles], page 96.

Voir aussi

Dans ce manuel : [Indications textuelles], page 96.

Référence du programme : Section “RehearsalMark” dans *Référence des propriétés internes*.

Fichiers d’initialisation : ‘`scm/translation-functions.scm`’ contient les définitions de `format-mark-numbers` et `format-mark-letters`. Elles seront source d’inspiration pour d’autres fonctions de formatage.

Exemples : Section “Rhythms” dans *Exemples de code*, Section “Expressive marks” dans *Exemples de code*.

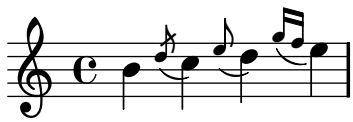
1.2.6 Fonctionnalités rythmiques particulières

1.2.6.1 Notes d’ornement

Les petites notes sont des ornements entièrement écrits. Les plus courantes sont les accacciatures, qui doivent se jouer très vite, et qui s’écrivent sous forme d’une petite note barrée (sur la hampe) et liée. L’appoggiature est une petite note non barrée, qui vole une fraction à la durée de la note réelle qui la suit.

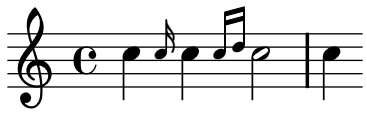
Ces petites notes sont entrées avec les commandes `\acciaccatura` et `\appoggiatura`, comme le montre l’exemple suivant :

```
b4 \acciaccatura d8 c4 \appoggiatura e8 d4
\acciaccatura { g16[ f] } e4
```



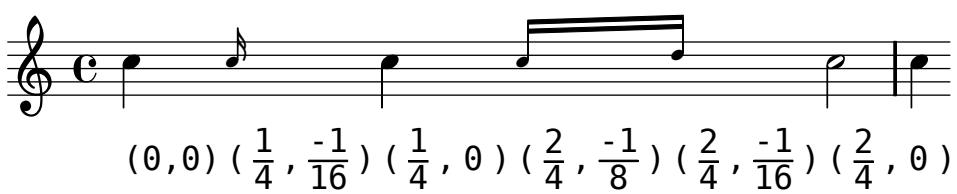
Ce sont là deux formes spéciales de la commande `\grace`, qui prend en charge toutes les petites notes. Si on la fait suivre d’une expression musicale, un groupe de petites notes sera créé, sans impact sur la métrique.

```
c4 \grace c16 c4
\grace { c16[ d16] } c2 c4
```



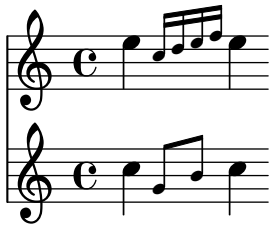
Contrairement à `\acciaccatura` ou `\appoggiatura`, la commande `\grace` ne provoque pas de liaison.

La durée des petites notes est interprétée par le programme en fonction d’un deuxième compteur de temps, le chronomètre `grace`. Chaque instant est défini par deux nombres rationnels : le premier compte les durées réelles, le second compte la durée des petites notes. Reprenons l’exemple ci-dessus en y ajoutant ces couples de nombres :



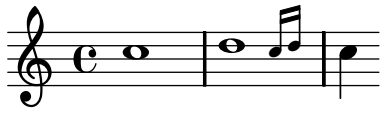
Les petites notes se placent de façon synchrone entre les différentes portées. Dans l’exemple suivant, il y a deux petites double-croches pour chaque petite croche.

```
<< \new Staff { e4 \grace { c16[ d e f] } e4 }
   \new Staff { c4 \grace { g8[ b] } c4 } >>
```



La commande `\afterGrace` sert à placer une petite note après une note réelle — et non *avant* comme d'ordinaire. Cette commande requiert deux arguments : la note réelle, et la ou les petites notes qui suivent.

```
c1 \afterGrace d1 { c16[ d] } c4
```



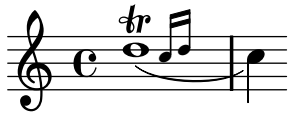
Les petites notes se placent alors aux 3/4 de la durée de la note réelle. Cette fraction peut être changée en définissant `afterGraceFraction` ; ainsi,

```
#(define afterGraceFraction (cons 7 8))
```

placera la petite note à 7/8 de la note réelle.

On peut obtenir le même effet manuellement, de la façon suivante :

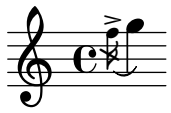
```
\new Voice {
  << { d1^\trill_( }
     { s2 \grace { c16[ d] } } >>
  c4)
}
```



Le silence invisible peut être plus ou moins long — ici c'est une demi-pause — afin d'ajuster l'espace entre la note réelle et les petites notes.

Les expressions `\grace` obéissent à des règles typographiques particulières, notamment pour régler l'orientation et la taille des objets. De ce fait, toute subtilité de mise en forme devra être indiquée *à l'intérieur* de l'expression introduite par `\grace` :

```
\new Voice {
  \acciaccatura {
    \stemDown
    f16->
    \stemNeutral
  }
  g4
}
```



Tous les réglages ajoutés doivent également être désactivés dans cette même expression.

Il est possible de changer globalement la mise en forme des petites notes dans un morceau, au moyen de la fonction `add-grace-property`. Ici, par exemple, on ôte la définition de l'orientation des objets `Stem` pour toutes les petites notes, afin que les hampes ne soient pas toujours orientées vers le haut.

```
\new Staff {
 #(add-grace-property 'Voice 'Stem 'direction '())
 ...
}
```

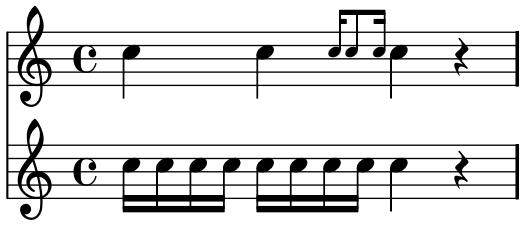
Il est par ailleurs possible de modifier les variables `startGraceMusic`, `stopGraceMusic`, `startAcciaccaturaMusic`, `stopAcciaccaturaMusic`, `startAppoggiaturaMusic`, et `stopAppoggiaturaMusic`. Pour plus de détails, voir le fichier '`ly/grace-init.ly`'.

Le trait que l'on trouve sur les hampes des accacciatures peut être appliqué dans d'autres situations en définissant
`\override Stem #'stroke-style = #"grace"`.

Propriétés couramment modifiées

Il est possible de forcer l'élasticité de l'espacement des notes d'agrément.

```
<<
  \override Score.SpacingSpanner #'strict-grace-spacing = ##t
  \new Staff {
    c4
    \afterGrace c4 { c16[ c8 c16] }
    c4 r
  }
  \new Staff {
    c16 c c c c c c c c4 r
  }
>>
```



Voir aussi

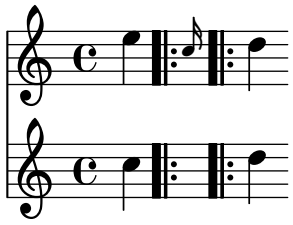
Référence du programme : Section "GraceMusic" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Une partition commençant par une expression `\grace` doit faire intervenir la commande `\new Voice`, sans quoi la note réelle et la petite note se retrouveront sur des portées différentes.

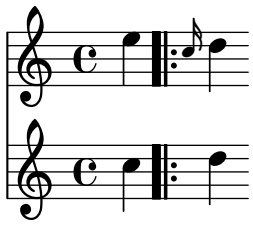
La synchronisation des petites notes se fait de façon parfois surprenante, car les autres objets de la portée — barres de mesure, armures, etc. — sont eux aussi synchrones. Pensez-y lorsque vous mélez des portées comprenant des petites notes et d'autres sans :

```
<< \new Staff { e4 \bar "|:" \grace c16 d4 }
   \new Staff { c4 \bar "|:" d4 } >>
```



Il est possible de remédier à cela en insérant sur les autres portées des silences invisibles dans une expression précédée de `\grace`, correspondant à la durée des petites notes.

```
<< \new Staff { e4 \bar "|:" \grace c16 d4 }
   \new Staff { c4 \bar "|:" \grace s16 d4 } >>
```



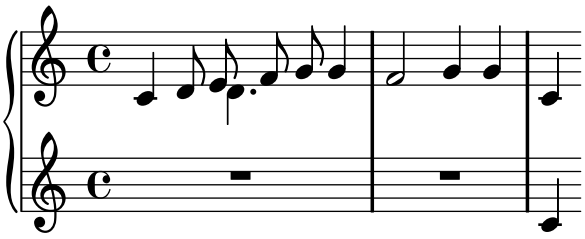
Seules des expressions musicales séquentielles peuvent être utilisées pour des petites notes ; il n'est pas possible d'imbriquer ni de juxtaposer des sections de petites notes, faute de quoi le traitement du code peut échouer ou produire des erreurs.

1.2.6.2 Alignement et cadences

Dans un contexte orchestral, une cadence constitue un problème spécifique. Lors du montage d'une partition contenant une cadence, tous les autres instruments doivent sauter autant de notes que ce qu'en comporte la cadence, faute de quoi il démarreraient trop tôt ou trop tard.

Les fonctions `mmrest-of-length` ou `skip-of-length` permettent de résoudre ce problème. Ces fonctions Scheme prennent en argument un fragment de musique, et génèrent un `\skip` ou un silence multi-mesures d'une durée correspondant à ce fragment. L'exemple qui suit illustre l'utilisation de `mmrest-of-length`.

```
cadence = \relative c' {
  c4 d8 << { e f g } \\ { d4. } >>
  g4 f2 g4 g
}

\new GrandStaff <<
  \new Staff { \cadence c'4 }
  \new Staff {
    #(ly:export (mmrest-of-length cadence))
    c'4
  }
>>
```



1.2.6.3 Gestion du temps

Le temps est administré par le Section “Time_signature_engraver” dans *Référence des propriétés internes*, qui réside en principe dans le contexte Section “Score” dans *Référence des propriétés internes*. Sa gestion traite les variables suivantes :

`currentBarNumber`
: Le numéro de mesure.

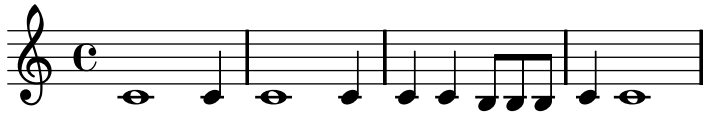
`measureLength`
: La longueur de la mesure, dans la métrique en cours. Pour une mesure à 4/4, elle est de 1, et de 3/4 pour une mesure à 6/8.

`measurePosition`
: Le moment où l’on en est dans la mesure en cours. Cette quantité est remise à 0 dès lors qu’on dépasse `measureLength` ; la variable `currentBarNumber` est alors incrémentée.

`timing`
: Lorsqu’on lui assigne la valeur *vrai*, les valeurs ci-dessus mentionnées sont mises à jour à chaque pas. Fixée à *faux*, le graveur restera indéfiniment dans la mesure en cours.

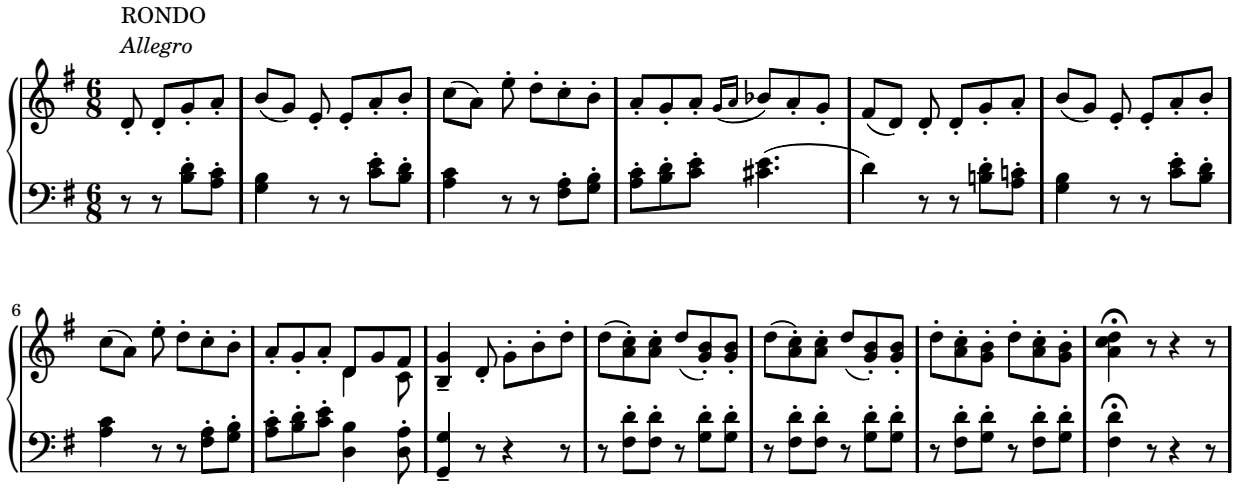
Le calage peut être modifié en réglant explicitement l’une de ces variables. Dans l’exemple qui suit, nous réglons la métrique à 4/4, tout en fixant `measureLength` à 5/4. Un peu plus loin, nous raccourcissons la mesure de 1/8, en assignant 7/8 à `measurePosition`, alors que nous en sommes à 2/4 dans la mesure ; la barre de mesure tombera donc à 2/4 + 3/8. Les 3/8 résultent du fait que 5/4 équivaut à 10/8, mais nous nous sommes recalés à 7/8 de la mesure ; donc $10/8 - 7/8 = 3/8$.

```
\set Score.measureLength = #(ly:make-moment 5 4)
c1 c4
c1 c4
c4 c4
\set Score.measurePosition = #(ly:make-moment 7 8)
b8 b b
c4 c1
```



Comme le montre cet exemple, `ly:make-moment n m` construit une durée de n/m fois une ronde. Par conséquent, `ly:make-moment 1 8` correspond à une croche, et `ly:make-moment 7 16` à la durée de sept doubles croches.

1.3 Signes d'interprétation



1.3.1 Indications attachées à des notes

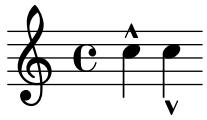
Articulations et ornements

Différents symboles peuvent être ajoutés au-dessus ou au-dessous des notes pour indiquer des ponctuations ou des modes de jeu différents. On les ajoute à chaque note au moyen d'un tiret suivi du caractère correspondant à l'articulation désirée. En voici une démonstration :

Il est possible de changer la signification de ces raccourcis : voir des exemples dans '`ly/script-init.ly`'.

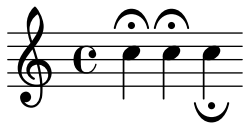
Même si LilyPond place automatiquement ces symboles, il est possible de l'obliger à les placer au-dessus ou en-dessous de la note, tout comme d'autres objets, en utilisant respectivement `^` et `_`.

```
c''4^^ c''4_^
```

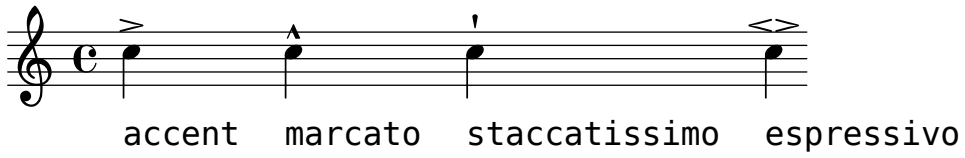


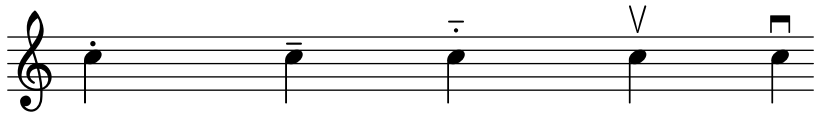
On peut ajouter d'autres symboles, avec la syntaxe *note*`\`*symbole*. Ici encore, on peut forcer leur orientation avec `^` and `_`.

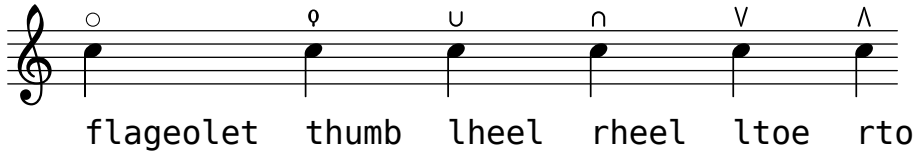
```
c\fermata c^\fermata c_\fermata
```

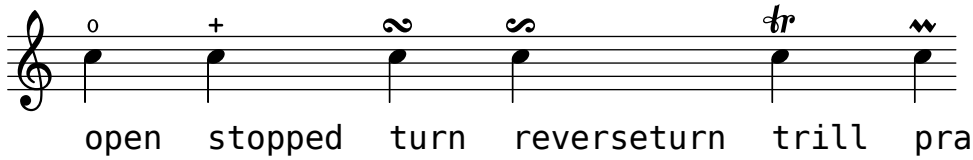


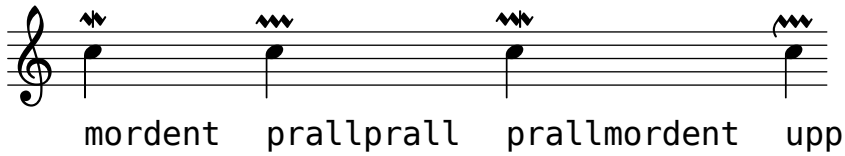
Voici la liste exhaustive des symboles :

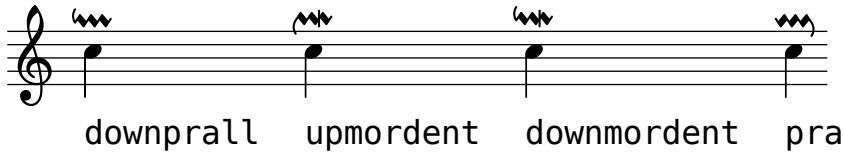


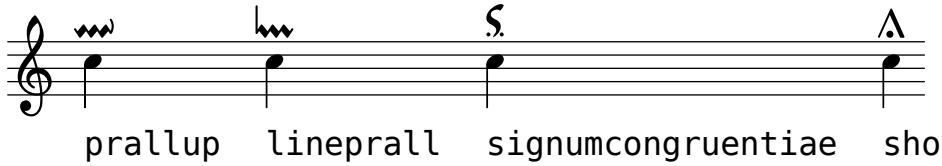

staccato tenuto portato upbow downbow

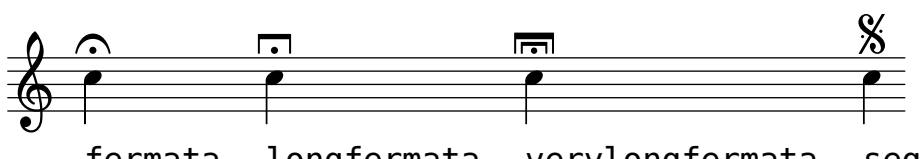

flageolet thumb lheel rheel ltoe rtoe

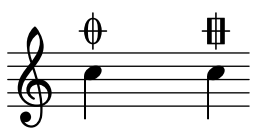

open stopped turn reverseturn trill prall


mordent prallprall prallmordent upprall


downprall upmordent downmordent pralldown


prallup lineprall signumcongruentiae shortfermata

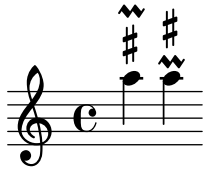

fermata longfermata verylongfermata segno


coda varcoda

Propriétés couramment modifiées

Les symboles s'ordonnent verticalement suivant la propriété `script-priority`. Plus sa valeur numérique est faible, plus le symbole sera proche de la note. Dans l'exemple suivant, l'objet Section "TextScript" dans *Référence des propriétés internes* — le dièse — a d'abord la propriété la plus basse, et il est donc placé plus près de la note ; ensuite c'est l'objet Section "Script" dans *Référence des propriétés internes* qui a la propriété la plus basse, et il se place alors sous le dièse. Lorsque deux objets ont la même priorité, c'est l'ordre dans lequel ils sont indiqués qui détermine lequel sera placé en premier.

```
\once \override TextScript #'script-priority = #-100
a4^\prall^\markup { \sharp }

\once \override Script #'script-priority = #-100
a4^\prall^\markup { \sharp }
```



Voir aussi

Référence du programme : Section "Script" dans *Référence des propriétés internes*.

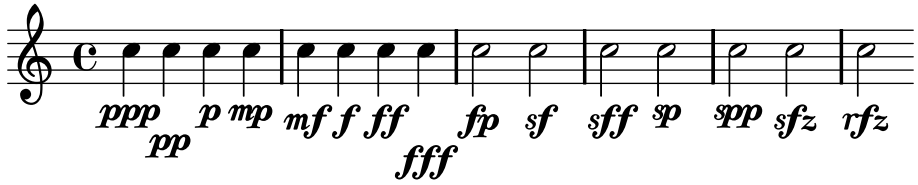
Problèmes connus et avertissements

Ces symboles sont présents sur la partition imprimée, mais n'ont pas d'effet sur le rendu de la musique en MIDI.

Nuances

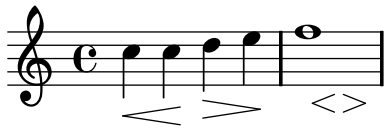
À chaque nuance absolue correspond une commande, qui peut être indiquée après une note : `c4\ff` par exemple. Les commandes de nuances disponibles sont `\ppppp`, `\pppp`, `\ppp`, `\pp`, `\p`, `\mp`, `\mf`, `\f`, `\ff`, `\fff`, `\ffff`, `\fp`, `\sf`, `\sff`, `\sp`, `\spp`, `\sfz`, et `\rfz`.

```
c\ppp c\pp c \p c\mp c\mf c\f c\ff c\fff
c2\fp c\sf c\sff c\sp c\spp c\sfz c\rfz
```



Un crescendo est délimité par `\<` et `\!`, ou peut se terminer par une commande de nuance. Au lieu de `\<` ou `\>`, on peut aussi utiliser `\cr` et `\decr`. Ces indications sont attachées aux notes ; aussi si l'on veut faire se succéder plusieurs nuances pendant une note tenue, il faudra avoir recours à des silences invisibles :

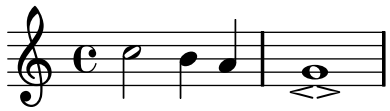
```
c\< c\! d\> e\!
<< f1 { s4 s4\< s4\! \> s4\! } >>
```



En principe, un soufflet — (de)crescendo imprimé sous forme graphique – commence au bord gauche de la note de départ, et se termine au bord droit de la note d'arrivée. Cependant, si la note d'arrivée est sur un premier temps, le soufflet s'arrêtera au niveau de la barre de mesure qui la précède. Ce comportement peut être annulé en assignant *faux* (lettre 'f') à la propriété `hairpinToBarline` :

On peut avoir recours à l'indication `\espressivo` pour indiquer un crescendo puis un decrescendo sur une seule note.

```
c2 b4 a g1\espressivo
```

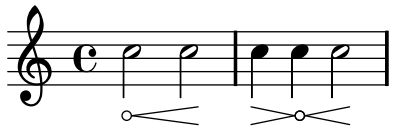


Cependant, ces deux indications graphiques peuvent se trouver très comprimées horizontalement. Pour les rallonger, on peut modifier la propriété `minimum-length` de `Voice.Hairpin` — Section "Voice" dans *Référence des propriétés internes* étant le contexte, et Section "Hairpin" dans *Référence des propriétés internes* l'objet affecté. Par exemple :

```
\override Voice.Hairpin #'minimum-length = #5
```

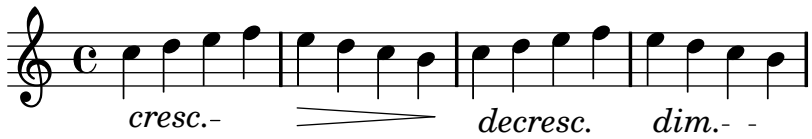
Des crescendos ou decrescendos *al niente* peuvent être indiqués de manière graphique, en assignant *vrai* (lettre 't') à la propriété `circled-tip`, ce qui affiche un cercle à leur extrémité.

```
\override Hairpin #'circled-tip = ##t
c2\< c\!
c4\> c\< c2\!
```



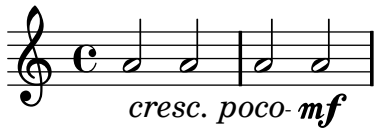
Au lieu d'une notation graphique des crescendos, on peut utiliser une notation textuelle.

```
\crescTextCresc
c\< d e f\!
\crescHairpin
e\> d c b\!
\dimTextDecresc
c\> d e f\!
\dimTextDim
e\> d c b\!
```



On peut même définir ses propres indications textuelles :

```
\set crescendoText = \markup { \italic "cresc. poco" }
\set crescendoSpanner = #'text
a'2\< a a a\!\mf
```



Pour créer des indications de nuances qui restent alignées avec les nuances habituelles, voir [Personnalisation des indications de nuance], page 54.

Le positionnement vertical des nuances est géré par le Section “DynamicLineSpanner” dans *Référence des propriétés internes*.

Propriétés couramment modifiées

Des nuances différentes situées — ou commençant — sur une même note seront alignées verticalement. Pour aligner des nuances qui ne se situeraient pas sur une même note, il est possible d'augmenter la propriété `staff-padding`.

```
\override DynamicLineSpanner #'staff-padding = #4
```

Cette propriété peut aussi servir à régler des problèmes de collision entre des nuances et d'autres objets.

Les crescendos ou decrescendos qui aboutissent sur la première note d'une nouvelle ligne ne sont imprimés que jusqu'à la fin de la ligne précédente. Ce comportement peut être outrepassé en définissant :

```
\override Score.Hairpin #'after-line-breaking = ##t
```

Les crescendos et decrescendos indiqués textuellement — tels que *cresc.* ou *dim.* — sont suivis de pointillés qui montrent leur étendue. On peut empêcher l'impression de ces pointillés avec :

```
\override DynamicTextSpanner #'dash-period = #-1.0
```

Commandes prédéfinies

`\dynamicUp`, `\dynamicDown`, `\dynamicNeutral`.

Voir aussi

Référence du programme : Section “DynamicText” dans *Référence des propriétés internes*, Section “Hairpin” dans *Référence des propriétés internes*.

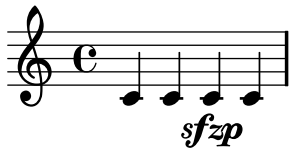
Le placement vertical de ces éléments graphiques est géré par le Section “DynamicLineSpanner” dans *Référence des propriétés internes*.

Personnalisation des indications de nuance

Grâce à la commande `make-dynamic-script`, vous pouvez créer de nouvelles marques textuelles de nuances que vous combinerez éventuellement avec les signes de nuances. Notez bien que la police des nuances en contient que les caractères `f`,`m`,`p`,`r`,`s` and `z`.

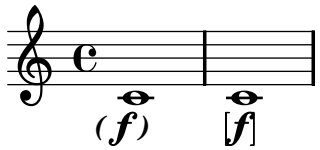
Certains composants, tels que les marques de nuances, possèdent des propriétés particulières et prédéfinies quant à leur police. Lorsque vous créez du texte en pareille situation, nous vous recommandons d'utiliser `normal-text` pour annuler ces propriétés. Voir Section B.6 [Text markup commands], page 196 pour plus de détails.

```
sfzp = #(make-dynamic-script "sfzp")
\relative c' {
  c4 c c\sfzp c
}
```



Vous pouvez aussi encadrer les nuances entre parenthèses ou entre crochets. Ceci est souvent utilisé pour ajouter des nuances propres à une édition donnée.

```
rndf = \markup{ \center-align {\line { \bold{\italic (}
  \dynamic f \bold{\italic )} }} }
boxf = \markup{ \bracket { \dynamic f } }
{ c'1_\rndf c'1_\boxf }
```

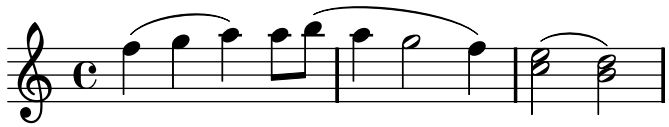


1.3.2 Courbes

Liaisons d'articulation

Une liaison d'articulation indique que les notes doivent être jouées liées, ou *legato*. Ces liaisons s'indiquent au moyen de parenthèses.

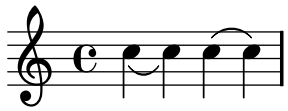
```
f( g a) a8 b( a4 g2 f4)
<c e>2( <b d>2)
```



On peut indiquer l'orientation des liaisons suivantes avec `\slur`*`DIR`*, *`DIR`* pouvant être `Up` pour une liaison vers le haut, `Down` pour une liaison vers le bas, ou `Neutral` pour laisser LilyPond décider.

Il existe également un raccourci pratique pour forcer l'orientation d'une seule liaison. Il suffit pour cela d'ajouter `_` ou `^` avant d'ouvrir une parenthèse.

```
c4_( c) c^( c)
```

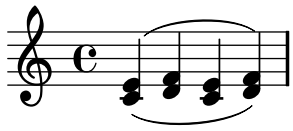


Une seule liaison d'articulation peut être imprimée à la fois. S'il est nécessaire d'imprimer une liaison plus longue, englobant des liaisons plus courtes, utilisez des [Liaisons de phrasé], page 56.

Propriétés couramment modifiées

Certains auteurs utilisent deux liaisons lorsqu'ils veulent lier des accords. Dans LilyPond, il faut pour cela assigner *vrai* ('true') la propriété `doubleSlurs` :

```
\set doubleSlurs = ##t
<c e>4 ( <d f> <c e> <d f> )
```



Commandes prédéfinies

`\slurUp`, `\slurDown`, `\slurNeutral`, `\slurDashed`, `\slurDotted`, `\slurSolid`.

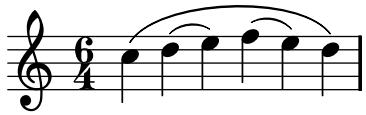
Voir aussi

Référence du programme : Section “Slur” dans *Référence des propriétés internes*.

Liaisons de phrasé

Une liaison de phrasé relie plusieurs notes en délimitant une phrase musicale. On indique les points de départ et d’arrivée avec `\(` et `\)` respectivement.

```
\time 6/4 c'\( d( e) f( e) d\)
```



D’un point de vue typographique, rien ne distingue une liaison de phrasé d’une liaison d’articulation. Cependant, LilyPond les considère comme des objets différents. Une commande `\slurUp` n’affectera donc pas une liaison de phrasé : il faut plutôt utiliser `\phrasingSlurUp`, `\phrasingSlurDown` ou `\phrasingSlurNeutral`.

Il n’est pas possible d’avoir plusieurs liaisons de phrasé en même temps.

Commandes prédéfinies

`\phrasingSlurUp`, `\phrasingSlurDown`, `\phrasingSlurNeutral`.

Voir aussi

Référence du programme : Section “PhrasingSlur” dans *Référence des propriétés internes*.

Signes de respiration

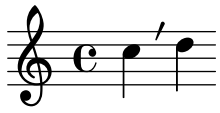
Les indications de respiration sont indiquées par la commande `\breathe`.

```
c'4 \breathe d4
```

Propriétés couramment modifiées

On peut choisir le glyphe imprimé par cette commande, en modifiant la propriété `text` de l'objet `BreathingSign`, pour lui affecter n'importe quelle indication textuelle. Par exemple :

```
c'4
\override BreathingSign #'text
  = #(make-musicglyph-markup "scripts.rvarcomma")
\breathe
d4
```



Voir aussi

Référence du programme : Section "BreathingSign" dans *Référence des propriétés internes*.

Exemples : Section "Winds" dans *Exemples de code*.

Chutes et sauts

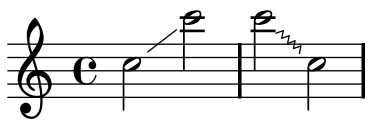
Des indications de désinence peuvent être obtenues au moyen de la commande `\bendAfter` :

1.3.3 Lignes

Glissando

Un glissando relie une hauteur à une autre en passant par chaque hauteur intermédiaire. Il est indiqué graphiquement, par une ligne ou des vaguelettes entre ces deux notes. On l'obtient en accolant la commande `\glissando` à la première note.

```
c2\glissando c'
\override Glissando #'style = #'zigzag
c2\glissando c,
```



Propriétés couramment modifiées

```
I = \once \override NoteColumn #'ignore-collision = ##t

\relative <<
  { \oneVoice \stemDown f2 \glissando \stemNeutral a } \\
  { \oneVoice \I c2 \glissando \I d, }
>>
```

Voir aussi

Référence du programme : Section “Glissando” dans *Référence des propriétés internes*.

Exemples : Section “Expressive marks” dans *Exemples de code*.

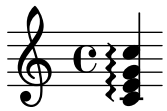
Problèmes connus et avertissements

Il n’est pas possible d’imprimer un texte (tel que *gliss.*) le long de la ligne de glissando.

Arpèges

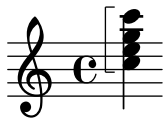
On peut indiquer qu’un accord doit être arpégé en lui accolant la commande `\arpeggio` :

```
<c e g c>\arpeggio
```



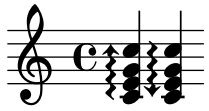
Pour spécifier qu’un autre accord doit être plaqué et non arpégé, on peut remplacer ce signe par un crochet :

```
\arpeggioBracket
<c' e g c>\arpeggio
```



Le sens de l’arpège est parfois indiqué par une pointe de flèche au bout de la vaguelette :

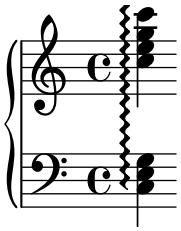
```
\new Voice {
  \arpeggioArrowUp
  <c e g c>\arpeggio
  \arpeggioArrowDown
  <c e g c>\arpeggio
}
```



Propriétés couramment modifiées

Quand un arpège couvre plusieurs portées, il est possible d’indiquer l’arpège sur chacune des portées, puis de relier la ligne de vaguelettes en assignant *vrai* à la propriété `connectArpeggios`, par exemple dans le contexte Section “PianoStaff” dans *Référence des propriétés internes* :

```
\new PianoStaff <<
  \set PianoStaff.connectArpeggios = ##t
  \new Staff { <c' e g c>\arpeggio }
  \new Staff { \clef bass <c,, e g>\arpeggio }
>>
```



Commandes prédéfinies

`\arpeggio`, `\arpeggioArrowUp`, `\arpeggioArrowDown`, `\arpeggioNormal`, `\arpeggioBracket`.

Voir aussi

Dans ce même manuel : Section 1.2.1.4 [Liaisons de prolongation], page 22, pour noter explicitement des arpèges.

Référence du programme : Section “Arpeggio” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

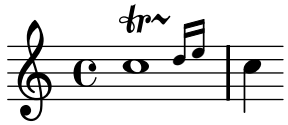
Il est impossible de mêler au même instant, dans un contexte Section “PianoStaff” dans *Référence des propriétés internes*, des lignes d’arpèges connectées et d’autres non connectées.

Trilles

Les trilles brefs s’indiquent comme n’importe quelle ponctuation : voir [Articulations et ornements], page 50.

Les trilles plus longs sont délimités par `\startTrillSpan` et `\stopTrillSpan` :

```
\new Voice {
  << { c1 \startTrillSpan }
     { s2. \grace { d16[\stopTrillSpan e] } } >>
  c4 }
```



Les trilles qui font intervenir une hauteur précise peuvent être indiqués par la commande `pitchedTrill`.

```
\pitchedTrill c4\startTrillSpan fis
f\stopTrillSpan
```

Le premier argument est la note réelle ; le second est une hauteur qui sera imprimée comme une tête de note noire entre parenthèses.

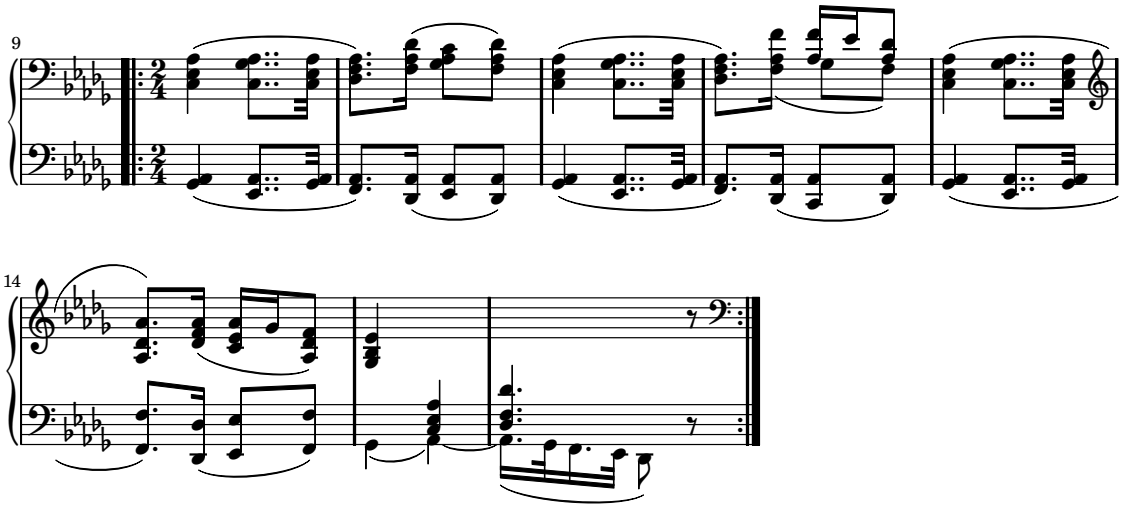
Commandes prédéfinies

`\startTrillSpan`, `\stopTrillSpan`.

Voir aussi

Référence du programme : Section “TrillSpanner” dans *Référence des propriétés internes*.

1.4 Répétitions et reprises



La répétition est une notion essentielle en musique, et il existe de nombreuses façons de mettre en œuvre et noter ce concept.

1.4.1 Écriture de répétitions

Types de répétitions

On peut indiquer des répétitions des façons suivantes :

`unfold` La musique qui doit être répétée sera entièrement imprimée (et jouée). Ceci est particulièrement utile dans de la musique répétitive. Ce type de reprise est le seul pris en compte dans le rendu MIDI.

`volta` Le passage répété ne sera pas écrit explicitement, mais il sera encadré sur la partition par des barres de reprises, et peut se terminer par plusieurs fins alternatives, imprimées de gauche à droite sous des crochets. Il s’agit là de la notation courante des reprises avec des fins alternatives. Ces dernières, par défaut, ne sont pas jouées dans le rendu MIDI.

`tremolo` Pour réaliser des trémolos. Ceux-ci, par défaut, ne sont pas joués dans le rendu MIDI.

`percent` Pour répéter des temps ou des mesures, imprimés sous la forme de signes de pourcentage. Ceux-ci, par défaut, ne sont pas joués dans le rendu MIDI. Les répétitions indiquées par ces symboles doivent être déclarées dans un contexte `Voice`.

Syntaxe des répétitions

Tous les différents types de reprise se spécifient dans LilyPond avec une même construction syntaxique, qui est :

```
\repeat type_de_la_reprise nombre_de_répétitions expression_à_répéter
```

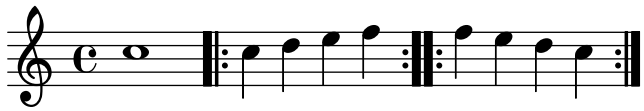
On peut ajouter, pour indiquer une fin alternative :

```
\alternative {
 alternative1
 alternative2
 alternative3
 ...
}
```

chaque *alternative* étant une expression musicale. Si l'on donne trop peu d'alternatives en regard du nombre de fois où le passage doit être rejoué, la première alternative sera jouée plusieurs fois.

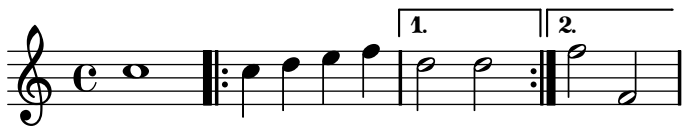
Les reprises courantes s'indiquent comme ceci :

```
c1
\repeat volta 2 { c4 d e f }
\repeat volta 2 { f e d c }
```



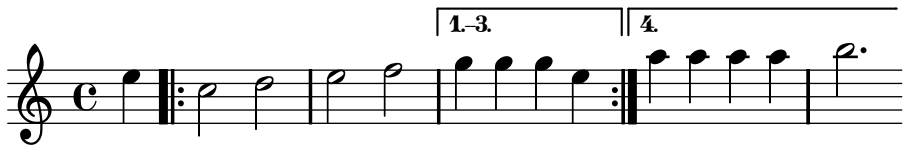
Et avec des fins alternatives :

```
c1
\repeat volta 2 {c4 d e f}
\alternative { {d2 d} {f f,} }
```



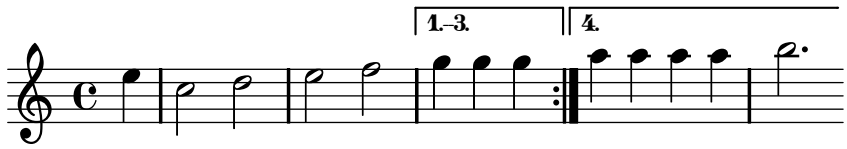
Il est possible de créer des répétitions avec une levée.

```
\new Staff {
  \partial 4 e |
  \repeat volta 4 { c2 d2 | e2 f2 | }
  \alternative { { g4 g g e } { a a a a | b2. } }
}
```



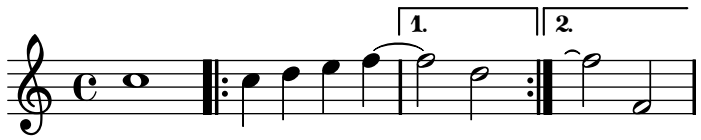
ou

```
\new Staff {
  \partial 4
  \repeat volta 4 { e | c2 d2 | e2 f2 | }
  \alternative { { \partial 4*3 g4 g g } { a a a a | b2. } }
}
```



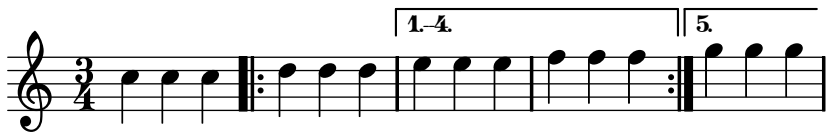
Des liaisons de tenue peuvent être ajoutées à toute alternative :

```
c1
\repeat volta 2 {c4 d e f ~ }
\alternative { {f2 d} {f\repeatTie f,} }
```



On peut raccourcir les crochets indiquant les fins alternatives, en définissant la propriété `voltaSpannerDuration`. Dans l'exemple suivant, le crochet ne se prolonge que sur une mesure à 3/4.

```
\relative c''{
  \time 3/4
  c c c
  \set Staff.voltaSpannerDuration = #(ly:make-moment 3 4)
  \repeat volta 5 { d d d }
  \alternative { { e e e f f f }
  { g g g } }
}
```



Si l'on souhaite à la fois terminer une ligne par une double barre et débuter la ligne suivante avec une section reprise, on peut utiliser

```
... \bar "||:" \break
\repeat volta 2 { ...
```

Consultez Section 1.2.5.1 [Barres de mesure], page 40 pour plus d'informations.

Voir aussi

Program reference: Section "VoltaBracket" dans *Référence des propriétés internes*, Section "RepeatedMusic" dans *Référence des propriétés internes*, Section "VoltaRepeatedMusic" dans *Référence des propriétés internes*, et Section "UnfoldedRepeatedMusic" dans *Référence des propriétés internes*.

Exemples

Les crochets précédant une reprise s'impriment d'ordinaire seulement au-dessus de la portée du haut. On peut ajuster cela en déplaçant le graveur `Volta_engraver` vers les contextes de portée (`Staff`) qui doivent comporter ces crochets ; voir Section 5.1.3 [Modification des greffons de contexte], page 177, et Section "Repeats" dans *Exemples de code*.

Problèmes connus et avertissements

Des reprises imbriquées telles que

```
\repeat ...
\repeat ...
\alternative
```

présentent une ambiguïté, dans la mesure où l'on ne sait à quelle section `\repeat` attribuer la section `\alternative`. Pour résoudre cette ambiguïté, il convient de toujours insérer la commande `\alternative` à l'intérieur de la section `\repeat`. Il est préférable, dans une telle situation, d'utiliser des accolades pour plus de clarté.

Lorsqu'une alternative commence, les informations de métrique sont perdues, il faut donc les rappeler après une reprise, par exemple en définissant `Score.measurePosition` ou en invoquant la commande `\partial`. De même, aucune liaison (de tenue ou autre) n'est répétée.

Les crochets qui indiquent les alternatives ne sont pas alignés verticalement.

Commandes de reprise manuelles

La propriété `repeatCommands` sert à contrôler la mise en forme des reprises. On la définit par une suite de commandes de reprise Scheme.

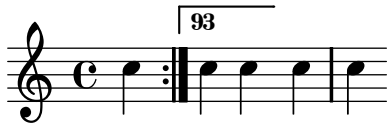
`start-repeat`
: Pour imprimer une barre de reprise `|:`

`end-repeat`
: Pour imprimer une barre de reprise `:|`

`(volta` *texte*`)`
: Pour imprimer un crochet indiquant une alternative. L'argument *texte* mentionné dans le crochet peut être n'importe quelle chaîne de caractères ou indication textuelle — voir Section 1.8.2 [Mise en forme du texte], page 99. Attention cependant à changer la police, car la police employée par défaut pour les chiffres ne contient aucun caractère alphabétique.

`(volta #f)`
: Pour terminer un crochet indiquant une alternative.

```
c4
  \set Score.repeatCommands = #'((volta "93") end-repeat)
c4 c4
  \set Score.repeatCommands = #'((volta #f))
c4 c4
```



Voir aussi

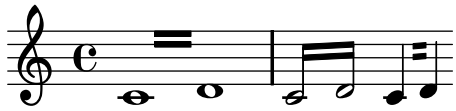
Référence du programme : Section "VoltaBracket" dans *Référence des propriétés internes*, Section "RepeatedMusic" dans *Référence des propriétés internes*, Section "VoltaRepeatedMusic" dans *Référence des propriétés internes*, et Section "UnfoldedRepeatedMusic" dans *Référence des propriétés internes*.

1.4.2 Autres types de répétition

Répétition en trémolo

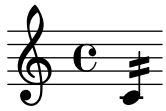
On peut placer une notation de trémolo entre deux notes, avec la commande `\repeat` suivie du style trémolo :

```
\new Voice \relative c' {
  \repeat tremolo 8 { c16 d16 }
  \repeat tremolo 4 { c16 d16 }
  \repeat tremolo 2 { c16 d16 }
}
```



On peut aussi indiquer un trémolo sur une seule note, qu'il faudra alors laisser sans accolades.

```
\repeat tremolo 4 c'16
```



La subdivision des trémolos aboutit à un résultat semblable : voir [Subdivision de trémolos], page 64.

Voir aussi

Dans ce manuel : [Subdivision de trémolos], page 64, Section 1.4 [Répétitions et reprises], page 60.

Référence du programme : Section "Beam" dans *Référence des propriétés internes*, Section "StemTremolo" dans *Référence des propriétés internes*.

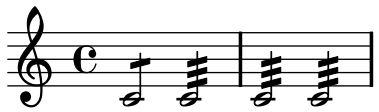
Subdivision de trémolos

Un trémolo peut être indiqué sur une seule note, en la faisant suivre de deux points et d'un nombre :

`note:`[*nombre*].

Le nombre en question correspond à la valeur de la subdivision ; il doit être au moins de 8, auquel cas la hampe sera barrée par un seul trait de ligature. Si ce nombre est omis, la dernière valeur — telle que mémorisée dans `tremoloFlags` — sera utilisée.

```
c'2:8 c':32 | c': c': |
```



Problèmes connus et avertissements

Les trémolos entrés de cette manière ne sont pas rendus dans le fichier MIDI.

Voir aussi

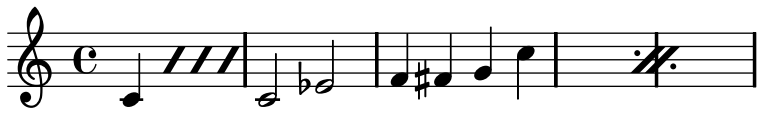
Dans ce manuel : [Répétition en trémolo], page 64.

Référence du programme : Section “StemTremolo” dans *Référence des propriétés internes*.

Répétitions de mesure

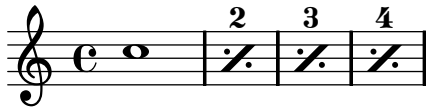
Le style de « reprise en pourcent » sert à répéter une séquence de notes. Elle sera imprimée une fois, puis remplacée par un symbole spécial. Les séquences d’une ou deux mesures sont remplacées par un symbole qui ressemble au symbole de pourcentage, tandis que les séquences inférieures à une mesure sont remplacées par une barre oblique. Toutes ces répétitions doivent être déclarées dans un contexte `Voice`.

```
\new Voice \relative c' {
  \repeat percent 4 { c4 }
  \repeat percent 2 { c2 es2 f4 fis4 g4 c4 }
}
```



Les répétitions de plus de 2 mesures sont surmontées d’un compteur, si l’on assigne *vrai* (lettre ‘t’) à la propriété `countPercentRepeats`.

```
\new Voice {
\set countPercentRepeats = ##t
  \repeat percent 4 { c1 }
}
```



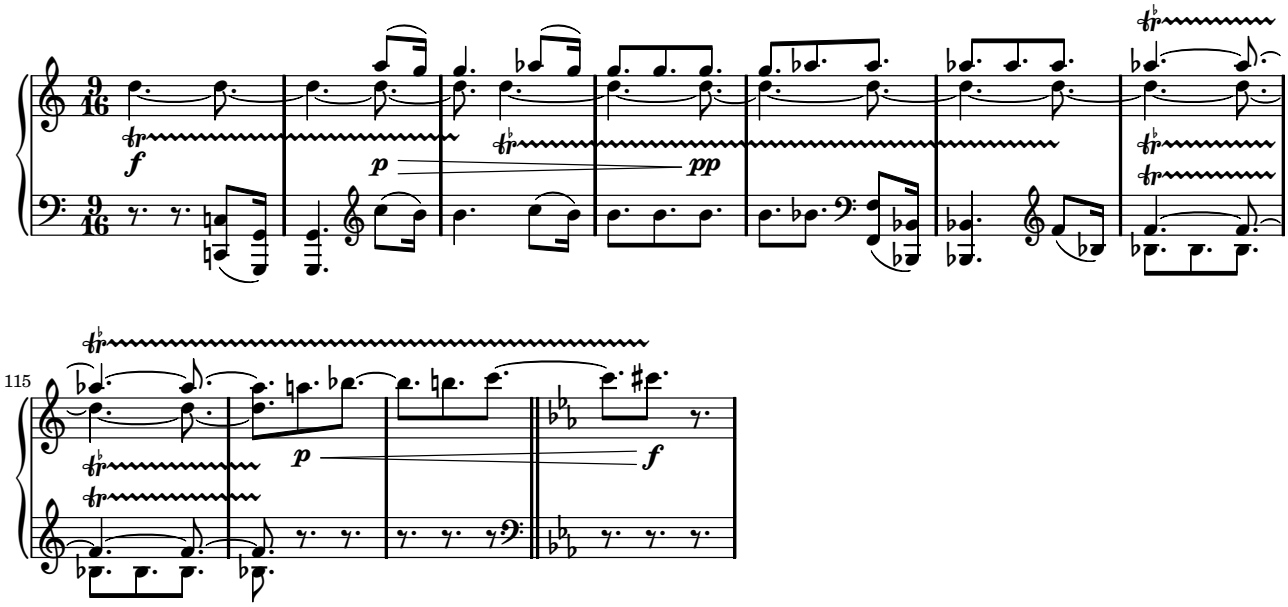
Des symboles de pourcentage isolés peuvent aussi être obtenus, au moyen d’un silence multi-mesures dont on modifie l’aspect :

```
\override MultiMeasureRest #'stencil
  = #ly:multi-measure-rest::percent
R1
```

Voir aussi

Référence du programme : Section “RepeatSlash” dans *Référence des propriétés internes*, Section “PercentRepeat” dans *Référence des propriétés internes*, Section “DoublePercentRepeat” dans *Référence des propriétés internes*, Section “DoublePercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatCounter” dans *Référence des propriétés internes*, Section “PercentRepeatedMusic” dans *Référence des propriétés internes*.

1.5 Notes simultanées



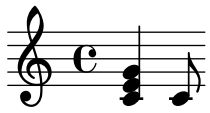
La notion musicale de polyphonie fait référence au fait d'avoir plus d'une voix simultanément dans une pièce. Dans LilyPond, la notion de polyphonie fait référence au fait d'avoir plus d'une voix sur la même portée.

1.5.1 Monophonie

Notes en accords

Un accord est formé en mettant une série de hauteurs entre `<` et `>`. Un accord peut être suivi d'une durée et d'indications d'articulation, comme une simple note.

```
<c e g>4 <c>8
```



Pour plus d'information à propos des accords, voir Section 2.7 [Notation des accords], page 137.

Clusters

Un cluster indique un agrégat de sons. On peut le représenter par une plage limitée par un ambitus (notes extrêmes). On obtient une telle notation en appliquant la fonction `makeClusters` à une séquence d'accords, comme

```
\makeClusters { <c e > <b f'> }
```

Des notes ordinaires et des clusters peuvent cohabiter sur une même portée, y compris simultanément — en pareil cas, rien ne sera fait pour tenter d'empêcher les chevauchements entre notes et clusters.

Voir aussi

Référence du programme : Section “ClusterSpanner” dans *Référence des propriétés internes*, Section “ClusterSpannerBeacon” dans *Référence des propriétés internes*, Section “Cluster_spanner_engraver” dans *Référence des propriétés internes*.

Exemples : Section “Simultaneous notes” dans *Exemples de code*.

Problèmes connus et avertissements

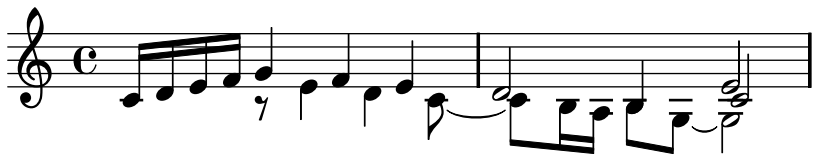
Les expressions musicales du type `<< { g8 e8 } a4 >>` ne seront pas imprimées de façon acceptable. utilisez plutôt `<g a>8 <e a>8` .

1.5.2 Plusieurs voix

Polyphonie basique

La manière la plus facile d’entrer des fragments avec plus d’une voix sur une portée est d’entrer chaque voix comme une suite de notes — entre accolades : `{...}` — puis de les combiner simultanément, en séparant les voix avec `\\`

```
\new Staff \relative c' {
  c16 d e f
  <<
    { g4 f e | d2 e2 } \\
    { r8 e4 d c8 ~ | c b16 a b8 g ~ g2 } \\
    { s2. | s4 b4 c2 }
  >>
}
```



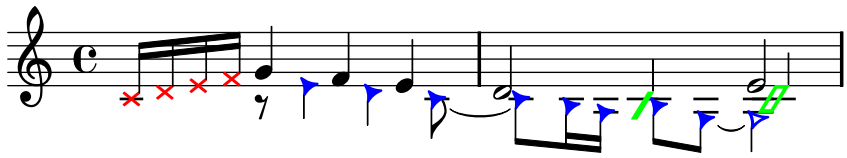
Le séparateur permet aux contextes Section “Voice” dans *Référence des propriétés internes*[1] d’être identifiés. Les contextes de voix portent les noms `"1"`, `"2"`, etc. Dans chacun de ces contextes, la direction verticale des liaisons, hampes, etc. est réglée de manière appropriée.

Ces voix sont toutes distinctes de la voix qui contient les notes en dehors de la construction `<< \\ >>`. On doit le prendre en compte lorsqu’on fait des changements au niveau des voix. Cela veut aussi dire que les liaisons de prolongation et d’articulation ne peuvent ni entrer ni sortir de la construction `<< \\ >>`. À l’inverse, des voix parallèles venant de constructions `<< \\ >>` séparées sur la même portée sont dans les mêmes voix. Voici le même exemple, avec des couleurs et têtes de note différentes pour chaque voix. Notez que le changement de style de tête de note n’affecte pas l’intérieur des constructions `<< \\ >>`. Aussi, le changement à la deuxième voix dans la première construction `<< \\ >>` l’affecte aussi dans la deuxième construction `<< \\ >>`, et la voix est liée entre les deux constructions.

```
\new Staff \relative c' {
  \override NoteHead #'style = #'cross
  \override NoteHead #'color = #red
  c16 d e f
  <<
```

[1] Ces voix polyphoniques sont parfois appelées « couches » ou « calques » dans d’autres logiciels de notation.

```
    { g4 f e } \\
    { \override NoteHead #'style = #'triangle
      \override NoteHead #'color = #blue
    r8 e4 d c8 ~ }
  >> |
  <<
    { d2 e2 } \\
    { c8 b16 a b8 g ~ g2 } \\
    { \override NoteHead #'style = #'slash
      \override NoteHead #'color = #green
      s4 b4 c2 }
  >>
}
```



La polyphonie ne change pas la relation des notes dans un bloc `\relative { }`. Chaque note est calculée en fonction de la note qui la précède immédiatement, sans tenir compte des différentes voix.

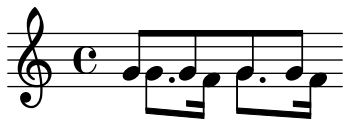
```
\relative { noteA << noteB \\ noteC >> noteD }
```

`noteC` est calculé relativement à `noteB`, non pas à `noteA` ; `noteD` est calculé relativement à `noteC`, non pas à `noteB` ou `noteA`.

Résolution des collisions

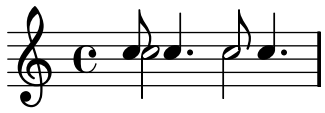
D'ordinaire, les têtes de notes pointées et non-pointées ne sont pas fusionnées, mais lorsque la propriété `merge-differently-dotted` de l'objet Section "NoteCollision" dans *Référence des propriétés internes* est définie, elles se trouvent fusionnées :

```
\new Voice << {
  g8 g8
  \override Staff.NoteCollision
    #'merge-differently-dotted = ##t
  g8 g8
} \\ { g8.[ f16] g8.[ f16] } >>
```



De même, vous pouvez fusionner une tête de blanche avec une tête de croche, en définissant `merge-differently-headed`

```
\new Voice << {
  c8 c4.
  \override Staff.NoteCollision
    #'merge-differently-headed = ##t
c8 c4. } \\ { c2 c2 } >>
```

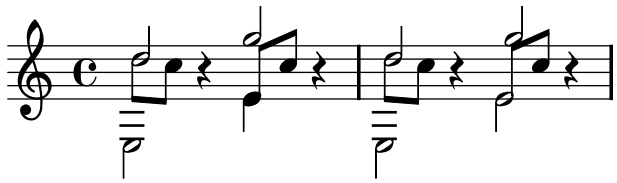


LilyPond décale aussi verticalement les silences à l'opposé des hampes, par exemple

```
\new Voice << c''4 \\ r4 >>
```

Lorsque trois notes ou plus s'agglutinent dans un même empilement, `merge-differently-headed` ne peut mener à bien la fusion des deux notes qui devraient l'être. Pour obtenir une fusion optimale, appliquez un décalage (`\shift`) à la note qui ne devrait pas s'empiler. Dans la première mesure de l'exemple suivant, `merge-differently-headed` ne fonctionne pas — la tête de la blanche est noire. Dans la seconde mesure, `\shiftOn` s'applique pour sortir le sol (`g`) de l'alignement, et `merge-differently-headed` fonctionne correctement.

```
\override Staff.NoteCollision #'merge-differently-headed = ##t
<<
  { d=''2 g2 } \\
  { \oneVoice d=''8 c8 r4 e,8 c'8 r4 } \\
  { \voiceFour e,,2 e'2}
>>
<<
  { d'=''2 \shiftOn g2 } \\
  { \oneVoice d=''8 c8 r4 e,8 c'8 r4 } \\
  { \voiceFour e,,2 e'2}
>>
```



Commandes prédéfinies

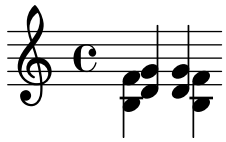
`\oneVoice`, `\voiceOne`, `\voiceTwo`, `\voiceThree`, `\voiceFour`.

`\shiftOn`, `\shiftOnn`, `\shiftOnnn`, `\shiftOff` : toutes ces commandes précisent le degré de décalage des notes de la voix courante. Les voix externes — habituellement, les voix une et deux — ont `\shiftOff`, alors que les voix internes — trois et quatre — ont `\shiftOn`. `\shiftOnn` et `\shiftOnnn` sont des niveaux supplémentaires de décalage.

Quand LilyPond est dépassé, la propriété `force-hshift` de l'objet Section "NoteColumn" dans *Référence des propriétés internes*, et des silences à hauteur déterminée, peuvent s'avérer utiles pour dicter au programme les choix de placement.

```
\relative <<
{
  <d g>
  <d g>
} \\ {
  <b f'>
  \once \override NoteColumn #'force-hshift = #1.7
```

```
  <b f'>
} >>
```



Voir aussi

Référence du programme : les objets appropriés pour résoudre les collisions sont Section "NoteCollision" dans *Référence des propriétés internes* et Section "RestCollision" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Quand `merge-differently-headed` est utilisé avec une croche ou une valeur plus courte à hampe vers le haut, et une blanche hampe vers le bas, la croche se retrouve à la mauvaise place.

Il n'y a aucune prise en charge des agrégats dans lesquels une même note apparaît avec différentes altérations. Il est conseillé d'avoir recours aux enharmoniques, ou d'utiliser la notation spécifique de cluster — voir [Clusters], page 66.

Regroupement automatique de parties

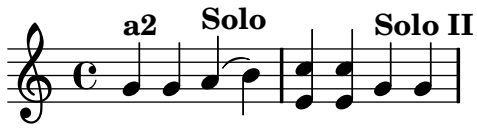
Le regroupement automatique de parties vous permet de fusionner deux pupitres sur une seule portée, ceci dans le but de créer des partitions d'orchestre. Lorsque les deux parties sont identiques sur une certaine durée, une seule s'affiche. Lorsqu'elles diffèrent, deux voix séparées apparaissent, avec des hampes dont la direction est gérée automatiquement. Vous pouvez aussi identifier et faire ressortir les solos et parties *a due*.

Voici la syntaxe qui permet de combiner des parties :

`\partcombine` *`musicexpr1 musicexpr2`*

L'exemple suivant illustre les fonctionnalités élémentaires du combinateur de parties : positionner les parties sur une portée, gérer la direction des hampes et de la polyphonie.

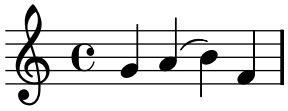
```
\new Staff \partcombine
  \relative g' { g g a( b) c c r r }
  \relative g' { g g r4 r e e g g }
```



Le premier `sol` n'apparaît qu'une seule fois, alors qu'il a été spécifié deux fois (une fois dans chacune des parties). La direction des hampes et des liaisons de tenue ou de phrasé est gérée automatiquement, selon qu'il s'agisse d'un solo ou d'un unisson. La première partie, dont le contexte s'appellera `one`, aura toujours ses hampes dirigées vers le haut et sera notée 'Solo', alors que la deuxième, appelée `two`, aura des hampes vers le bas et sera notée 'Solo II'.

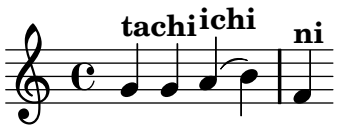
Si votre intention n'est que de fusionner les parties, sans ajouter de texte, assignez faux à la propriété `printPartCombineTexts`.

```
\new Staff <<
  \set Staff.printPartCombineTexts = ##f
  \partcombine
    \relative g' { g a( b) r }
    \relative g' { g r4 r f }
>>
```



Le texte imprimé pour les sections solo ou à l'unisson se règle par les propriétés `soloText`, `soloIIText`, et `aDueText`.

```
\new Staff <<
  \set Score.soloText = #"ichi"
  \set Score.soloIIText = #"ni"
  \set Score.aDueText = #"tachi"
  \partcombine
    \relative g' { g4 g a( b) r }
    \relative g' { g4 g r r f }
>>
```



LilyPond interprète dans un contexte Section "Voice" dans *Référence des propriétés internes* les arguments fournis à `\partcombine`. Si vous travaillez avec des octaves relatives, spécifiez `\relative` dans chacune des expressions musicales, comme ceci :

```
\partcombine
  \relative ... musicexpr1
  \relative ... musicexpr2
```

Une section `\relative` en dehors de `\partcombine` sera sans effet sur les hauteurs de *musicexpr1* et *musicexpr2*.

Voir aussi

Référence du programme : Section "PartCombineMusic" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Lorsque `printPartCombineTexts` est actif et que les deux voix jouent souvent les mêmes notes, le combinateur peut afficher `a2` plus d'une fois par mesure.

`\partcombine` ne peut s'inscrire dans un bloc `\times`.

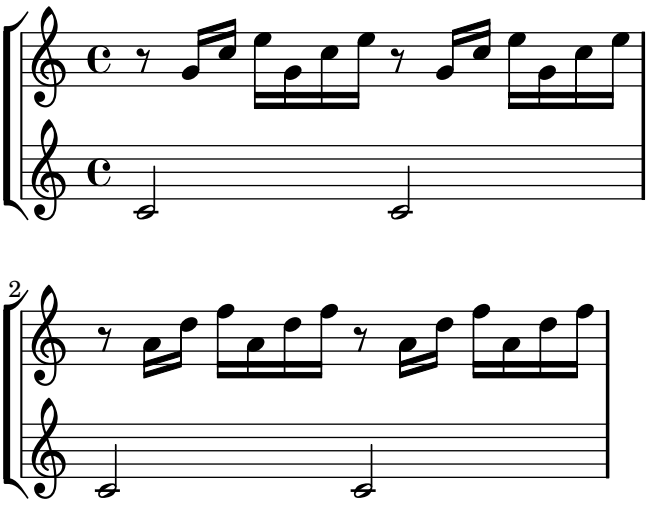
`\partcombine` ne peut s'inscrire dans un bloc `\relative`.

En interne, `\partcombine` interprète les deux arguments en tant que `Voice`s, dénommées `one` et `two`, puis décide de quand les parties seront fusionnées. Par conséquent, si les arguments changent pour d'autres noms de contexte Section "Voice" dans *Référence des propriétés internes*, les événements qu'ils contiendraient seront ignorés.

Saisie la musique en parallèle

On peut écrire plusieurs voix de façon entremêlée :

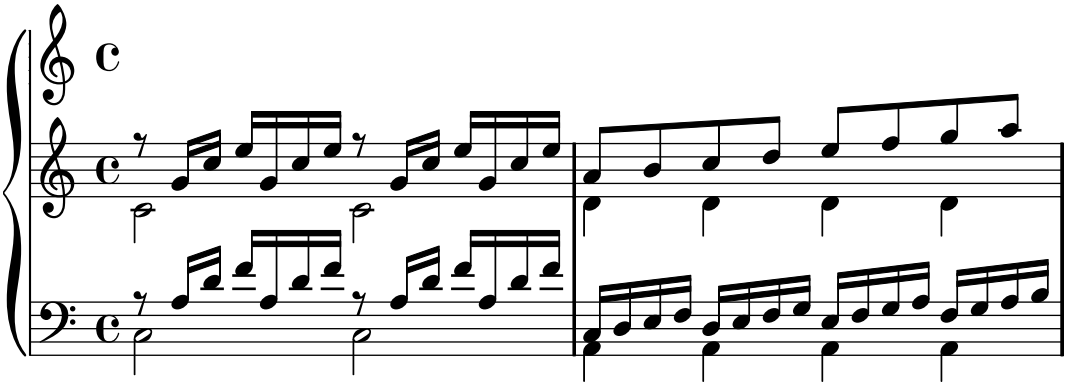
```
\parallelMusic #'(voiceA voiceB) {
  r8     g'16[ c''] e''[ g' c'' e''] r8     g'16[ c''] e''[ g' c'' e''] |
  c'2                                c'2                                |
  r8     a'16[ d''] f''[ a' d'' f''] r8     a'16[ d''] f''[ a' d'' f''] |
  c'2                                c'2                                |
}
\new StaffGroup <<
  \new Staff \new Voice \voiceA
  \new Staff \new Voice \voiceB
>>
```



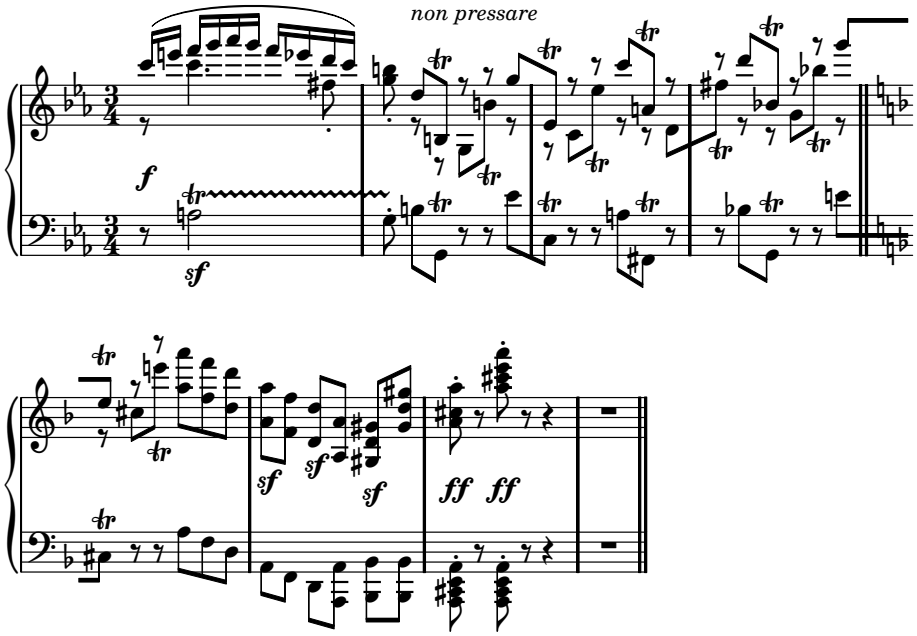
Ceci fonctionne bien avec la musique pour piano :

```
musique = {
  \key c \major
  \time 4/4
  \parallelMusic #'(voiceA voiceB voiceC voiceD) {
    % Mesure 1
    r8  g'16[ c''] e''[ g' c'' e''] r8  g'16[ c''] e''[ g' c''
e''] |
    c'2                                 c'2 |
    r8  a16[ d'] f'[ a d' f']       r8  a16[ d'] f'[ a d' f'] |
    c2                                  c2 |

    % Mesure 2
    a'8 b'      c'' d''    e'' f''    g'' a'' |
    d'4         d'         d'         d' |
    c16 d e f   d e f g    e f g a    f g a b |
    a,4         a,4        a,4        a,4 |

    % Mesure 3...
  }
}

\score {
```

```
  \new PianoStaff <<
    \musique
    \new Staff <<
      \voiceA \\
      \voiceB
    >>
    \new Staff {
      \clef bass
      <<
        \voiceC \\
        \voiceD
      >>
    }
  >>
}
```



1.6 Notation sur la portée



Cette section aborde les détails de gravure de la portée, la réalisation de partitions avec plusieurs portées et l'ajout d'indications globales d'exécution, présentes sur certaines portées seulement.

1.6.1 Gravure des portées

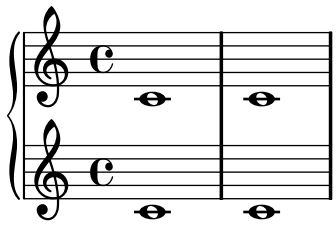
Initialisation de nouvelles portées

Regroupement de portées

De nombreuses partitions sont écrites sur plusieurs portées. Ces portées peuvent être regroupées de quatre manières différentes.

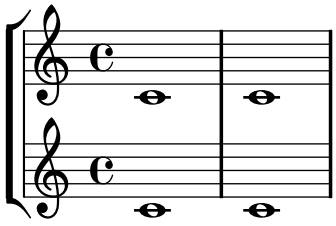
- Le groupe de portées est attaché par une accolade sur la gauche, et les barres de mesure sont d'un seul tenant. Il s'agit du contexte Section "GrandStaff" dans *Référence des propriétés internes*.

```
\new GrandStaff
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



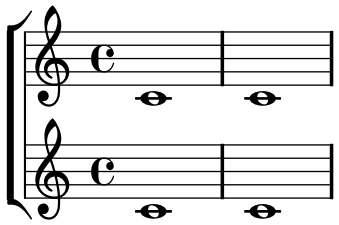
- Le groupe de portées est attaché par un crochet, et les barres de mesure sont d'un seul tenant. Il s'agit du contexte Section "StaffGroup" dans *Référence des propriétés internes*.

```
\new StaffGroup
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



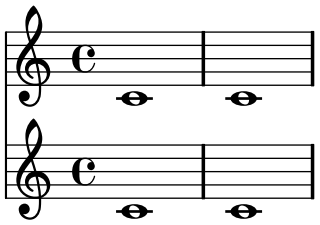
- Le groupe de portées est attaché par un crochet, mais les barres de mesure sont séparées d'une portée à l'autre. Il s'agit du contexte Section "ChoirStaff" dans *Référence des propriétés internes*.

```
\new ChoirStaff
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



- Les portées du groupe ne sont pas attachées (hormis par une simple ligne verticale). Les barres de mesure sont détachées. Il s'agit de l'assemblage par défaut.

```
\relative <<
  \new Staff { c1 c }
  \new Staff { c c }
>>
```



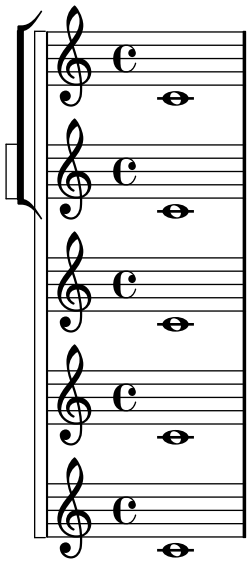
Voir aussi

Les barres de mesure au début de chaque système prennent l'un des styles Section "SystemStartBar" dans *Référence des propriétés internes*, Section "SystemStartBrace" dans *Référence des propriétés internes*, Section "SystemStartBracket" dans *Référence des propriétés internes*. Dans chaque contexte, seul l'un de ces styles est utilisé, et c'est la propriété `systemStartDelimiter` qui détermine lequel.

Propriétés couramment modifiées

Les accolades et crochets délimitant les systèmes peuvent être imbriqués en profondeur,

```
\new StaffGroup
\relative <<
  \set StaffGroup.systemStartDelimiterHierarchy
    = #'(SystemStartSquare (SystemStartBracket a (SystemStartSquare b)) d)
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
  \new Staff { c1 }
>>
```



Regroupements imbriqués de portées

1.6.2 Modification de portées individuelles

Cette section explique le réglage de la gravure de chaque portée, comme la taille de portée ou le nombre de lignes ; sont aussi décrits la suspension et la reprise de portées et les portées d'*ossia*.

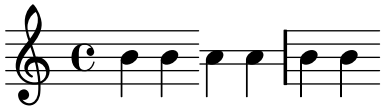
Symbole de la portée

Les notes, nuances, etc. sont regroupés dans un assemblage de lignes horizontales, que l'on nomme la portée (en anglais « staff », et « staves » au pluriel). Dans LilyPond, ces lignes sont dessinées au moyen d'un objet de mise en forme à part entière, nommé `staff symbol` — symbole de portée.

L'aspect du symbole de portée peut être modifié selon différentes propriétés, telles que le nombre de lignes, leur épaisseur, ou leur éloignement.

De plus, la portée peut commencer et s'arrêter où l'on veut, grâce aux commandes `\startStaff` et `\stopStaff`.

```
b4 b
\override Staff.StaffSymbol #'line-count = 2
\stopStaff \startStaff
b b
\revert Staff.StaffSymbol #'line-count
\stopStaff \startStaff
b b
```



Cette manière de procéder peut être utilisée pour introduire des « ossias », ou dans des partitions à nombre de portées variable, comme sur l'exemple suivant :

Voir aussi

Référence du programme : Section "StaffSymbol" dans *Référence des propriétés internes*.

Exemples : Section "Staff notation" dans *Exemples de code*.

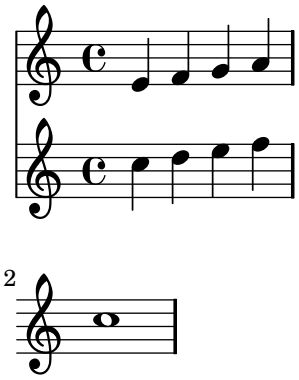
Portées d'ossia

Masquage de portées

Dans les partitions d'orchestre, les portées qui n'ont que des silences sont habituellement masquées afin de gagner de la place. Ce style d'édition s'appelle en anglais « French Score ». Cette fonctionnalité est activée par défaut dans les contextes Section "Lyrics" dans *Référence des propriétés internes*, Section "ChordNames" dans *Référence des propriétés internes* et Section "FiguredBass" dans *Référence des propriétés internes*. Lorsque des lignes appartenant à ces contextes se retrouvent vides après placement des sauts de ligne, elles sont effacées.

En ce qui concerne les portées normales, il existe un contexte Section "Staff" dans *Référence des propriétés internes* spécifique qui permet d'arriver à ce résultat : les portées ne contenant rien ou uniquement des silences multi-mesures seront retirées. La définition de ce contexte est enregistrée dans la variable `\RemoveEmptyStaffContext`. Voyez comment la deuxième portée disparaît du deuxième système :

```
\layout {
  \context { \RemoveEmptyStaffContext }
}

{
  \relative c' <<
    \new Staff { e4 f g a \break c1 }
    \new Staff { c4 d e f \break R1 }
  >>
}
```



Le premier système comportera absolument toutes les portées. Si vous voulez masquer les portées vides y compris pour le premier système, vous devrez assigner vrai à la propriété `remove-first` dans Section "VerticalAxisGroup" dans *Référence des propriétés internes*.

```
\override Score.VerticalAxisGroup #'remove-first = ##t
```

Pour masquer d'autres types de contextes, vous pouvez utiliser `\AncientRemoveEmptyStaffContext` ou `\RemoveEmptyRhythmicStaffContext`.

Une application particulière de cette fonctionnalité est la création d'une *ossia* — variante d'une partie de la mélodie — affichée à l'aide d'une portée supplémentaire.

1.6.3 Écriture de parties séparées

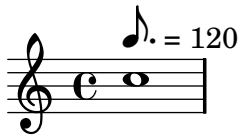
Indications métronomiques

Le métronome se règle de la manière suivante,

```
\tempo durée = par minute
```

Les indications métronomiques seront interprétées, dans le fichier MIDI, comme des changements de tempo. Ils seront imprimés sur la partition comme ici :

```
\tempo 8.=120 c''1
```



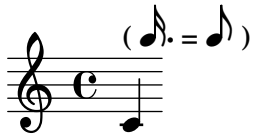
Propriétés couramment modifiées

Vous pouvez indiquer un changement de tempo pour le fichier MIDI sans pour autant l'imprimer. Il suffit alors de le rendre invisible pour l'impression :

```
\once \override Score.MetronomeMark #'transparent = ##t
```

Vous pouvez imprimer d'autres indications métronomiques, telles que des équivalences, en utilisant ce type d'étiquette textuelle :

```
c4^\markup {
  (
  \smaller \general-align #Y #DOWN \note #"16." #1
  =
  \smaller \general-align #Y #DOWN \note #"8" #1
  ) }
```



Pour plus de détails, voir Section 1.8.2 [Mise en forme du texte], page 99.

Voir aussi

Référence du programme : Section "MetronomeMark" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les risques de collision ne sont pas vérifiés. Dans le cas où il y aurait des notes au dessus de la portée ou d'autres objets (articulations, liaisons, texte, etc), l'indication métronomique peut venir en surimpression. Augmentez alors le décalage de cette indication par rapport à la portée :

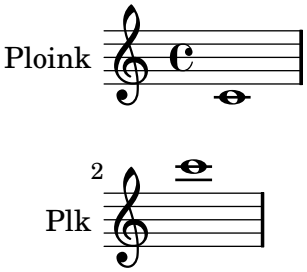
```
\override Score.MetronomeMark #'padding = #2.5
```

Noms d'instrument

Dans un conducteur, les noms d'instrument sont portés en regard de chacune des portées.

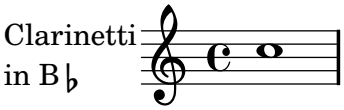
Ce résultat s'obtient en spécifiant Section "Staff" dans *Référence des propriétés internes*.`instrumentName` et Section "Staff" dans *Référence des propriétés internes*.`shortInstrumentName`, ou Section "PianoStaff" dans *Référence des propriétés internes*.`instrumentName` et Section "PianoStaff" dans *Référence des propriétés internes*.`shortInstrumentName`. L'argument textuel apparaîtra avant le début de la portée. La première ligne affichera `instrumentName`, et les suivantes `shortInstrumentName`.

```
\set Staff.instrumentName = "Ploink "
\set Staff.shortInstrumentName = "Plk "
c1
\break
c''
```



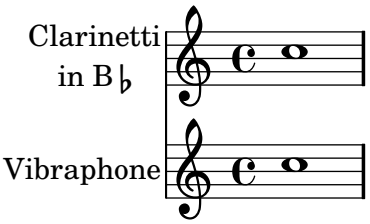
Le recours à la commande `\markup` permet de construire des noms d'instruments particuliers, tels que

```
\set Staff.instrumentName = \markup {
  \column { "Clarinetti"
            \line { "in B" \smaller \flat } } }
c''1
```



Si vous centrez le nom d'un instrument, il faudra le faire pour tous

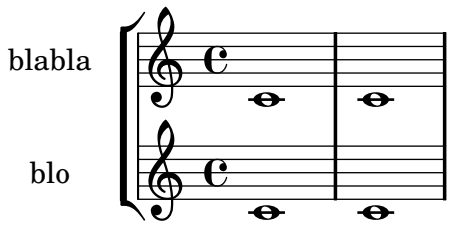
```
{ <<
\new Staff {
  \set Staff.instrumentName = \markup {
    \center-column { "Clarinetti"
      \line { "in B" \smaller \flat } } }
  c''1
}
\new Staff {
  \set Staff.instrumentName = \markup{ \center-align { Vibraphone }}
  c''1
}
>>
}
```



Losque le nom d'un instrument est relativement long, il est judicieux d'augmenter le retrait — `indent` — au sein du bloc `\layout`.

Procédez comme suit pour centrer des noms d'instruments tout en préservant un décalage par rapport à la portée :

```
\new StaffGroup \relative
<<
  \new Staff {
    \set Staff.instrumentName = \markup { \hcenter-in #10 "blabla" }
    c1 c1
  }
  \new Staff {
    \set Staff.instrumentName = \markup { \hcenter-in #10 "blo" }
    c1 c1
  }
>>
```



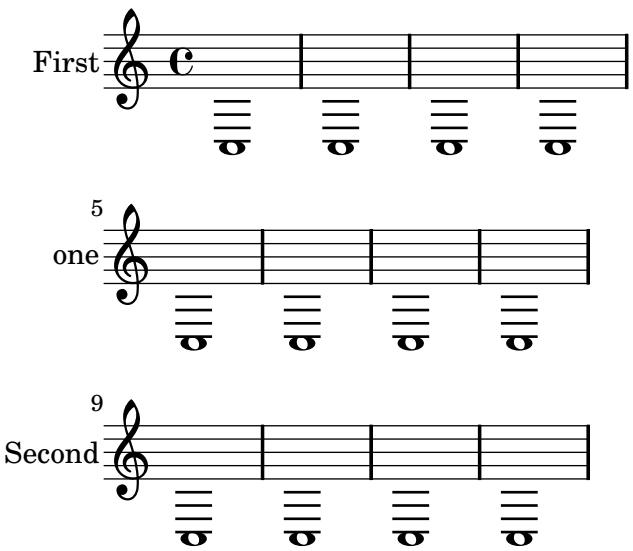
Des noms d'instruments peuvent s'utiliser dans d'autres contextes, tels que `GrandStaff`, `ChoirStaff`, ou `StaffGroup`, à condition de leur affecter le graveur approprié :

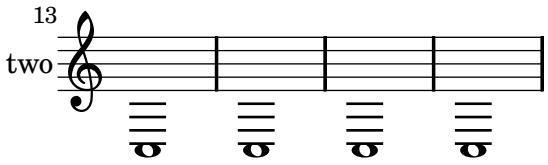
```
\layout{
  \context {\GrandStaff \consists "Instrument_name_engraver"}
}
```

Pour de plus amples informations sur la manière d'activer ou désactiver un graveur, voir Section 5.1.3 [Modification des greffons de contexte], page 177.

Vous pouvez changer d'instrument en cours de morceau :

```
\set Staff.instrumentName = "First"
\set Staff.shortInstrumentName = "one"
c1 c c c \break
c1 c c c \break
\set Staff.instrumentName = "Second"
\set Staff.shortInstrumentName = "two"
c1 c c c \break
c1 c c c \break
```





Voir aussi

Référence du programme : Section “InstrumentName” dans *Référence des propriétés internes*.

Citation d’autres voix

Grâce aux citations de répliques, vous pouvez insérer directement dans une partie des fragments d’une autre partie. Avant qu’une partie ne puisse être mentionnée ailleurs, elle doit être considérée comme reproductible. C’est le but de la commande `\addQuote`.

`\addQuote` *`nom musique`*

Ici, *nom* représente une chaîne d’identification, et *musique* n’importe quelle musique. Voici un exemple de `\addQuote` :

```
\addQuote clarinet \relative c' {
  f4 fis g gis
}
```

Vous devez placer cette commande au niveau le plus haut, c’est à dire en dehors de tout bloc de musique.

Après avoir fait appel à `\addquote`, la citation interviendra en utilisant `\quoteDuring` ou `\cueDuring` :

`\quoteDuring` *`#nom musique`*

Au cours d’une partie, des extraits de répliques peuvent être cités en utilisant la commande `\quoteDuring`.

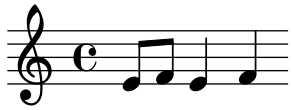
```
\quoteDuring #"clarinet" { s2. }
```

Cela citera trois noires (la durée de `s2.`) appartenant à la voix `clarinette` précédemment générée.

Plus précisément, on s’arrête à cet instant de la partie en cours d’impression, et l’on extrait les notes à ce même instant dans la voix citée — celle qui contient `\addQuote`. C’est la raison pour laquelle l’argument de `\addQuote` doit englober toute la voix en question, y compris les éventuels silences à son début.

Les citations tiennent compte des transpositions, aussi bien celle de l’instrument d’origine que celle de la partie où elle intervient, dans la mesure où elles sont spécifiées par la commande `\transposition`.

```
\addQuote clarinet \relative c' {
  \transposition bes
  f4 fis g gis
}

{
  e'8 f'8 \quoteDuring #"clarinet" { s2 }
}
```



Le type d'événements pris en charge pour la citation peut se régler avec la propriété `quotedEventTypes`. Par défaut, sa valeur est fixée à `(note-event rest-event)`, ce qui signifie que seuls les notes et silences seront mentionnés par `\quoteDuring`. Définir

```
\set Staff.quotedEventTypes =
       #'(note-event articulation-event dynamic-event)
```

reproduira les notes (mais pas les silences), ainsi que les scripts et nuances.

Problèmes connus et avertissements

Seul le contenu de la première Section "Voice" dans *Référence des propriétés internes* rencontrée dans la partie marquée d'une commande `\addQuote` pourra être retenu. Par voie de conséquence, `music` ne saurait comprendre de `\new` ou une instance `context Voice` qui la ferait passer à une autre voix.

Citer des notes d'ornement ne fonctionne pas, et peut même entraîner un blocage de LilyPond.

Citer des triolets imbriqués peut entraîner un résultat de médiocre qualité.

Voir aussi

Dans ce manuel : [Instruments transpositeurs], page 10.

Exemples : Section "Staff notation" dans *Exemples de code*.

Référence du programme : Section "QuoteMusic" dans *Référence des propriétés internes*.

Mise en forme d'une citation

La section précédente indiquait comment insérer des notes d'une autre voix. Nous allons maintenant voir une fonction musicale avancée, `\cueDuring`, qui facilite le formatage des petites notes.

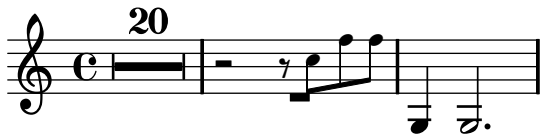
Sa syntaxe est :

```
\cueDuring #nom #updown musique
```

Des notes issues de la partie *nom* s'insèreront dans une Section "Voice" dans *Référence des propriétés internes* nommée `cue`, simultanément avec *musique* — habituellement un silence. L'apparition des petites notes initialise une polyphonie temporaire pour la portée concernée. L'argument *updown* détermine si ces petites notes seront attachées à la première ou à la seconde voix.

```
smaller = {
  \set fontSize = #-2
  \override Stem #'length-fraction = #0.8
  \override Beam #'thickness = #0.384
  \override Beam #'length-fraction = #0.8
}

\addQuote clarinet \relative {
  R1*20
  r2 r8 c' f f
}

\new Staff \relative  <<

  % setup a context for cue notes.
```

```
  \new Voice = "cue" { \smaller \skip 1*21 }

  \set Score.skipBars = ##t

  \new Voice {
    R1*20
    \cueDuring #"clarinet" #UP {
      R1
    }
    g4 g2.
  }
>>
```



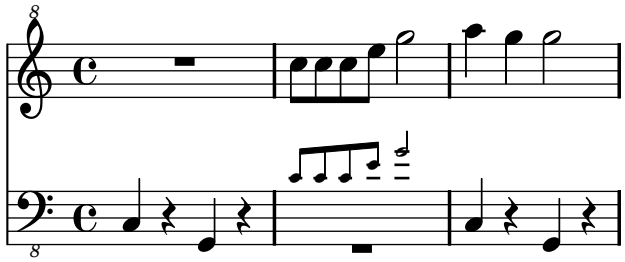
Quelques indications pour une citation efficace :

- Les notes de la citation sont dans une police plus petite.
- La citation comporte une étiquette indiquant l'instrument qui joue.
- Lorsque la partie originale reprend sa place, rappeler l'instrument.

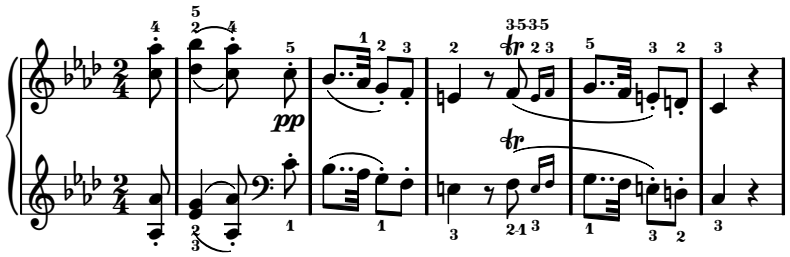
  Tout autre modification introduite par la citation devrait être annulée. Par exemple, si l'instrument cité utilise une autre clé, il faudra revenir à la clé habituelle.

La macro `\transposedCueDuring` est particulièrement adaptée pour des instrument ayant une tessiture éloignée, comme dans le cas d'un piccolo cité dans une partie de contrebasson.

```
picc = \relative c''' {
  \clef "treble^8"
  R1 |
  c8 c c e g2 |
  a4 g g2 |
}
\addQuote "picc" { \picc }

cbsn = \relative c, {
  \clef "bass_8"
  c4 r g r
  \transposedCueDuring #"picc" #UP c,, { R1 } |
  c4 r g r |
}

<<
  \context Staff = "picc" \picc
  \context Staff = "cbsn" \cbsn
>>
```



1.7 Notation éditoriale



1.7.1 Dans la portée

Indication de la taille de fonte musicale

Le plus sûr moyen de régler la taille de la police, quelque soit le contexte, consiste à définir la propriété `fontSize`.

```
c8
\set fontSize = #-4
c f
\set fontSize = #3
g
```

Ceci ne modifiera en rien la taille des différents symboles tels que ligatures ou liaisons.

En interne, la propriété `fontSize` d'un contexte aura pour effet de définir la propriété `font-size` pour tous les objets de rendu. La valeur de `font-size` est un entier représentant la taille proportionnellement à la hauteur standard de la portée en cours. Chaque incrément correspond à une augmentation d'environ 12 % de la taille de la police. Un pas de six aboutit exactement au doublement de la taille. La fonction Scheme `magstep` convertit le nombre affecté à `font-size` en facteur d'échelle. Vous pouvez aussi définir directement la propriété `font-size` de manière à n'affecter seulement que certains objets de rendu.

```
c8
\override NoteHead #'font-size = #-4
c f
\override NoteHead #'font-size = #3
g
```

Pour changer la taille des symboles musicaux (police Feta), LilyPond met à l'échelle la fonte dont la taille est la plus proche de la taille voulue — cf. Section 4.2.1 [Définition de la taille de portée], page 171. La taille standard, pour laquelle `font-size` vaut 0, dépend de la hauteur de la portée. À une portée de 20 points correspond une police de 10 points.

La propriété `font-size` ne peut intervenir que pour les objets qui utilisent des polices, autrement dit ceux qui disposent de l'interface de rendu Section "font-interface" dans *Référence des propriétés internes*.

Commandes prédéfinies

Les commandes suivantes définissent `fontSize` pour la voix en cours :

`\tiny`, `\small`, `\normalsize`.

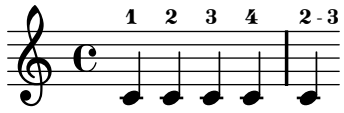
Doigtés

Les doigtés peuvent être indiqués comme suit :

`note-chiffre_du_doigt`

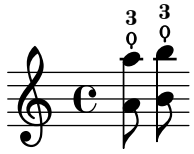
Pour les substitutions de doigts, on a recours à une indication textuelle (commande `\markup`) de doigté (commande `\finger`).

```
c4-1 c-2 c-3 c-4
c^\markup { \finger "2 - 3" }
```



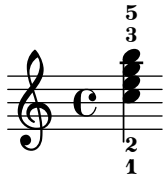
La commande `\thumb` peut être utilisée pour indiquer, par exemple dans une partition de violoncelle, si une note doit être jouée avec le pouce ('thumb' en anglais).

```
<a_\thumb a'-3>8 <b_\thumb b'-3>
```



Les doigtés des accords peuvent être saisis note par note, en les indiquant après chaque hauteur de note.

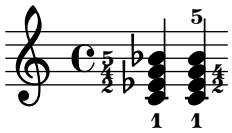
```
< c-1 e-2 g-3 b-5 >4
```



Propriétés couramment modifiées

On peut contrôler précisément les doigtés des accords en réglant la propriété `fingeringOrientations`.

```
\set fingeringOrientations = #'(left down)
<c-1 es-2 g-4 bes-5 > 4
\set fingeringOrientations = #'(up right down)
<c-1 es-2 g-4 bes-5 > 4
```



Cette propriété permet également, dans de la musique monophonique, d'indiquer des doigtés très proches des têtes de notes.

```
\set fingeringOrientations = #'(right)
<es'-2>4
```

Voir aussi

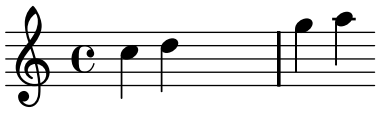
Référence du programme : Section "Fingering" dans *Référence des propriétés internes*.

Exemples : Section "Editorial annotations" dans *Exemples de code*.

Dictée à trous

Les notes masquées — ou invisibles ou encore transparentes — sont utiles dans le cadre d'exercices de théorie ou de composition.

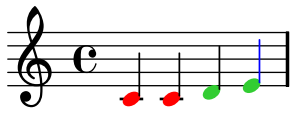
```
c4 d4
\hideNotes
e4 f4
\unHideNotes
g4 a
```



Coloration d'objets

Des objets peuvent être colorisés individuellement. Une listes des noms des couleurs disponibles se trouvent à l'annexe Section B.3 [Liste des couleurs], page 196.

```
\override NoteHead #'color = #red
c4 c
\override NoteHead #'color = #(x11-color 'LimeGreen)
d
\override Stem #'color = #blue
e
```



L'intégralité de la palette des couleurs définies pour X11 est accessible par la fonction Scheme x11-color. Cette fonction prend en argument une expression symbolique

```
\override Beam #'color = #(x11-color 'MediumTurquoise)
```

ou une chaîne de caractères

```
\override Beam #'color = #(x11-color "MediumTurquoise")
```

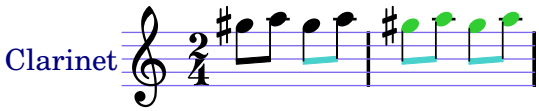
La première formulation est à la fois plus rapide à écrire et aussi plus efficace. Néanmoins, la deuxième forme permet d'accéder aux noms composés des couleurs de X11.

```
\override Beam #'color = #(x11-color "medium turquoise")
```

Lorsque la fonction x11-color ne trouve pas le paramètre fourni, elle revient à la couleur par défaut, le noir. Le problème ressort de façon évidente au vu de la partition finale.

L'exemple suivant illustre l'utilisation de la fonction `x11-color`. Notez que, après avoir été définie à (x11-color 'Boggle), la couleur des hampes repasse au noir.

```
{
  \override Staff.StaffSymbol #'color = #(x11-color 'SlateBlue2)
  \set Staff.instrumentName = \markup {
    \with-color #(x11-color 'navy) "Clarinet"
  }
  \time 2/4
  gis''8 a''
  \override Beam #'color = #(x11-color "medium turquoise")
  gis'' a''
  \override NoteHead #'color = #(x11-color "LimeGreen")
  gis'' a''
  \override Stem #'color = #(x11-color 'Boggle)
  gis'' a''
}
```



Voir aussi

Annexes : Section B.3 [Liste des couleurs], page 196.

Problèmes connus et avertissements

Les couleurs de X11 ne sont pas toutes perceptibles dans un navigateur internet. Aussi nous vous recommandons, dans le cadre d'une présentation multimedia, d'utiliser des couleurs de base.

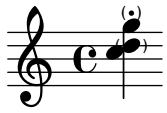
Une couleur x11 n'aura pas forcément le même rendu qu'une couleur normale ayant un nom similaire.

Vous ne pouvez pas coloriser des notes à l'intérieur d'un accord avec `\override`. si besoin est, utilisez `\tweak`. Pour plus de détails, consultez Section 5.2.5 [La commande tweak], page 187.

Parenthèses

Des objets peuvent être mis entre parenthèses en saisissant `\parenthesize` juste avant l'événement musical.

```
<
  c
  \parenthesize d
  g
>4-\parenthesize -.
```



Ceci n'est opérationnel que dans le cadre d'un accord, qui peut ne comprendre qu'une seule note.

```
< \parenthesize NOTE>
```

Hampes

Dès qu'une note est rencontrée, un objet Section "Stem" dans *Référence des propriétés internes* se crée automatiquement. Pour les rondes et les silences, ils sont aussi créés, mais en mode invisible.

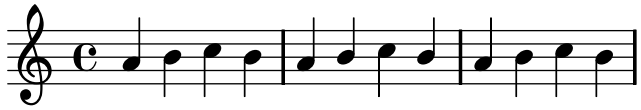
Commandes prédéfinies

`\stemUp`, `\stemDown`, `\stemNeutral`.

Propriétés couramment modifiées

Pour changer la direction des hampes au milieu de la portée, utilisez

```
a4 b c b
\override Stem #'neutral-direction = #up
a4 b c b
\override Stem #'neutral-direction = #down
a4 b c b
```



1.7.2 Hors de la portée

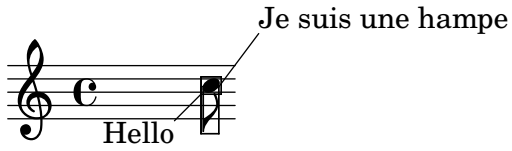
Info-bulle

Vous pouvez marquer et nommer des éléments de notation à l'aide de bulles. L'objectif premier de cette fonctionnalité est d'expliquer la notation.

En voici un exemple :

```
\new Voice \with { \consists "Balloon_engraver" }
{
  \balloonGrobText #'Stem #'(3 . 4) \markup { "Je suis une hampe" }
  <c-\balloonText #'(-2 . -2) \markup { Hello }  >8
```

```
}
```



Vous disposez de deux fonctions musicales, `balloonText` et `balloonGrobText`. `balloonGrobText` prend en argument l'objet graphique à agrémenter, alors que `balloonText` s'utilise comme une simple articulation. Les autres arguments sont le décalage et le texte de la bulle.

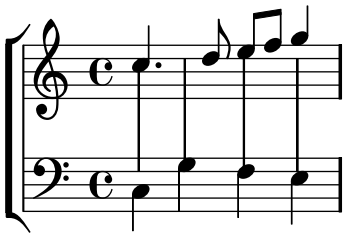
Voir aussi

Référence du programme : Section "balloon-interface" dans *Référence des propriétés internes*.

Quadrillage temporel

Vous pouvez tracer des lignes entre les portées, synchronisées avec les notes.

```
\layout {
  \context {
    \Staff
    \consists "Grid_point_engraver" %% active les guides
    gridInterval = #(ly:make-moment 1 4)
  }
}

\new Score \with {
  \consists "Grid_line_span_engraver"
  %% centre les lignes guides horizontalement sous les notes
  \override NoteColumn #'X-offset = #-0.5
}

\new ChoirStaff <<
  \new Staff {
    \stemUp
    \relative {
      c'4. d8 e8 f g4
    }
  }
  \new Staff {
    %% centre les lignes guides verticalement
    \override Score.GridLine #'extra-offset = #'( 0.0 . 1.0 )
    \stemDown
    \clef bass
    \relative c {
      c4  g'  f  e
    }
  }
>>
```

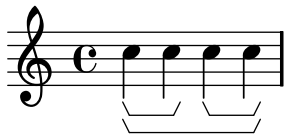


Exemples : Section “Editorial annotations” dans *Exemples de code*.

Crochets d’analyse

On utilise des crochets en analyse musicale, pour indiquer la structure d’une pièce. LilyPond permet d’utiliser une forme simplifiée de crochets horizontaux imbriqués, dans la mesure où le contexte Section “Staff” dans *Référence des propriétés internes* comporte le graveur Section “Horizontal_bracket_engraver” dans *Référence des propriétés internes*. Un crochet s’ouvre avec `\startGroup`, et se ferme avec `\stopGroup`.

```
\score {
  \relative c'' {
    c4\startGroup\startGroup
    c4\stopGroup
    c4\startGroup
    c4\stopGroup\stopGroup
  }
  \layout {
    \context {
      \Staff \consists "Horizontal_bracket_engraver"
}}}
```



Voir aussi

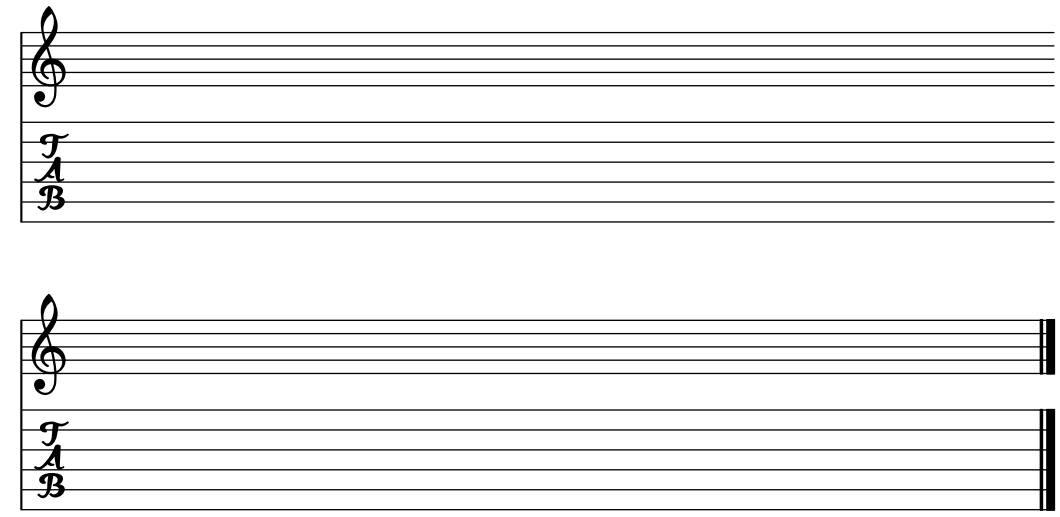
Référence du programme : Section “HorizontalBracket” dans *Référence des propriétés internes*.

Papier à musique

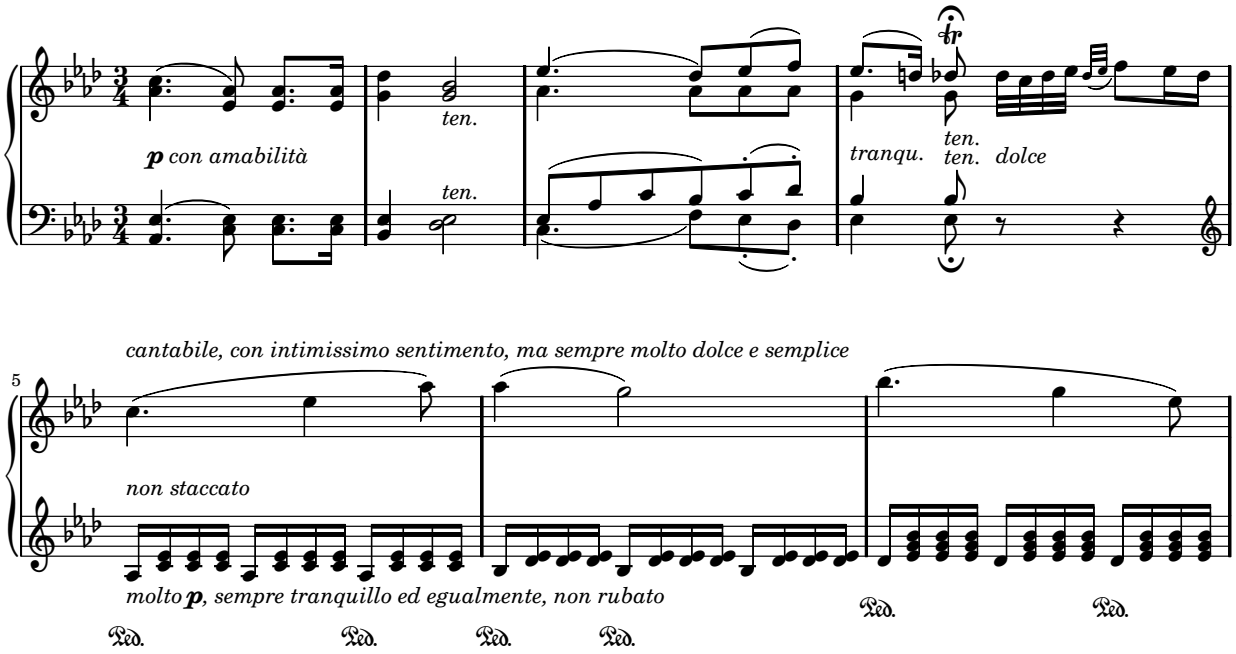
Une feuille de papier musique s’obtient en utilisant des notes invisibles, et en invalidant le `Bar_number_engraver`.

```
\layout{ indent = #0 }
emptymusic = {
  \repeat unfold 2 % À modifier pour plus de lignes.
  { s1\break }
  \bar "|."
}
\new Score \with {
  \override TimeSignature #'transparent = ##t
% décommenter la  ligne suivante selon besoin
%  \override Clef #'transparent = ##t
  defaultBarType = #""
  \remove Bar_number_engraver
} <<
```

```
% à adapter selon les portées désirées
  \new Staff \emptymusic
  \new TabStaff \emptymusic
>>
```



1.8 Texte



Nous allons voir ici comment insérer dans une partition du texte, avec différentes possibilité de formatage.

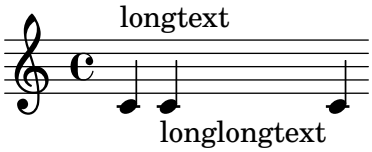
Pour écrire des accents et autres caractères spéciaux, il suffit de les insérer directement dans votre fichier LilyPond. Ce fichier devra être sauvegardé avec l'encodage UTF-8. Pour plus d'informations, voir Section 3.3.3 [Codage du texte], page 168.

1.8.1 Ajout de texte

Commentaires textuels

Vous pouvez placer arbitrairement des chaînes de caractères, ou Section 1.8.2 [Mise en forme du texte], page 99 en langage LilyPond, au dessus ou au dessous des notes en employant la syntaxe `c^"text"`. Par défaut, ces indications n'affecteront en rien l'espacement des notes, sauf à utiliser la commande `\textLengthOn`.

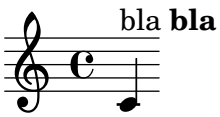
```
c4^"longtext" \textLengthOn c4_"longlongtext" c4
```



Pour revenir à l'espacement par défaut, utilisez `\textLengthOff`.

Des constructions plus élaborées d'étiquette peuvent être obtenues en ayant recours à la commande *markup* :

```
c'4^\markup { bla \bold bla }
```



La commande `\markup` est décrite plus en détails dans la section Section 1.8.2 [Mise en forme du texte], page 99.

Commandes prédéfinies

`\textLengthOn`, `\textLengthOff`.

Propriétés couramment modifiées

S'assurer que tous les éléments textuels et les paroles respectent les marges du document requiert des calculs relativement lourds ; c'est la raison pour laquelle LilyPond, par défaut, ne s'en préoccupe pas. Vous pouvez cependant l'y forcer en définissant

```
\override Score.PaperColumn #'keep-inside-line = ##t
```

Voir aussi

Dans ce manuel : Section 1.8.2 [Mise en forme du texte], page 99.

Référence du programme : Section "TextScript" dans *Référence des propriétés internes*.

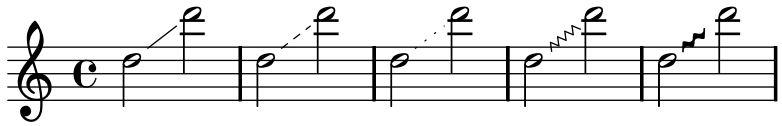
Indications textuelles et lignes d'extension

Certaines indications d'interprétation comme *rallentando*, *accelerando* ou *trilles*, s'incrivent textuellement et se prolongent sur plusieurs mesures à l'aide d'une ligne pleine, pointillée ou ondulée.

Les routines chargées de matérialiser un *glissando* sont tout à fait adaptées à une gestion précise, à la fois du placement du texte, et du calcul de l'envergure de sa ligne d'extension. La routine `ly:line-interface::print` est plus particulièrement en charge de déterminer les points d'ancrage de la ligne et de la dessiner selon le style requis.

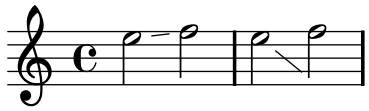
Voici un exemple qui illustre les différents styles de ligne disponibles, ainsi que la manière de les personnaliser.

```
d2 \glissando d'2
\once \override Glissando #'style = #'dashed-line
d,2 \glissando d'2
\override Glissando #'style = #'dotted-line
d,2 \glissando d'2
\override Glissando #'style = #'zigzag
d,2 \glissando d'2
\override Glissando #'style = #'trill
d,2 \glissando d'2
```



L'information qui va déterminer les extrémités est calculée à la volée pour chaque objet graphique. Il est tout à fait possible de les régler vous-même :

```
e2 \glissando f
\once \override Glissando #'bound-details #'right #'Y = #-2
e2 \glissando f
```

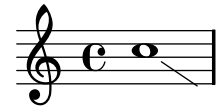


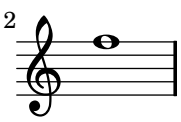
L'objet `Glissando`, comme tous ceux qui utilisent la routine `ly:line-interface::print`, comporte une liste d'associations. Dans le code ci-dessus, la valeur de `Y` est fixée, dans la liste d'associations, à `-2` pour l'ancrage à droite. Vous pouvez naturellement ajuster l'extrémité gauche en remplaçant `right` (pour `droite`) par `left`.

Si `Y` n'est pas fixé, sa valeur sera calculée en fonction de la hauteur du point de référence droite de la ligne.

Lorsque survient un saut de ligne, la liste des ancrages est augmentée d'une liste complémentaire contenant `left-broken` (brisure à gauche) et `right-broken` (brisure à droite), comme dans l'exemple suivant :

```
\override Glissando #'breakable = ##T
\override Glissando #'bound-details #'right-broken #'Y = #-3
c1 \glissando \break
f1
```





Vous disposez des propriétés suivantes :

`Y` Fixe l'ordonnée (coordonnée-Y) de l'extrémité, mesurée en taille de portée. Il s'agit par défaut du centre de l'objet de rattachement ; pour un glissando, ce sera le milieu de la tête de note.

Pour des marques horizontales, telles du texte ou le trait d'un trille, cette valeur est figée à 0.

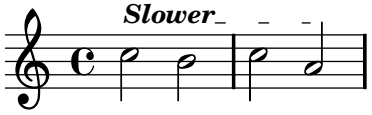
`attach-dir`
Détermine l'endroit où la ligne commence et finit, relativement à l'objet de rattachement. Autrement dit, une valeur de `-1` (ou `LEFT` pour gauche) fera commencer ou finir la ligne du côté gauche de la tête de note de référence.

`X` Coordonnée absolue du point final. Dans la mesure où elle est calculée à la volée, il n'y a pas vraiment de raison de l'outrepasser.

`stencil` Sous-propriété contenant les éventuels symboles présents avant ou après la ligne. Destinée à un usage interne, nous vous recommendons d'utiliser plutôt `text`.

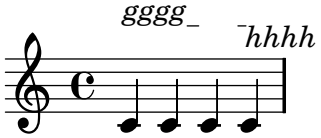
`text` Marqueur qui sera analysé pour alimenter `stencil`. On y trouve habituellement les *cresc.* ou *tr* des extenseurs horizontaux.

```
\override TextSpanner #'bound-details #'left #'text
   = \markup { \small \bold Slower }
c2\startTextSpan b c a\stopTextSpan
```



`stencil-align-dir-y`
`stencil-offset`
Lorsqu'ils ne sont pas définis, le tracé est tout simplement positionné conformément aux sous-propriétés `X` et `Y`. En fixant soit `stencil-align-dir-y`, soit `stencil-offset`, vous pouvez décaler le coin du marqueur par rapport à l'extrémité de la ligne.

```
\override TextSpanner #'bound-details #'left #'stencil-align-dir-y = #DOWN
\override TextSpanner #'bound-details #'right #'stencil-align-dir-y = #UP

\override TextSpanner #'bound-details #'left #'text = #"gggg"
\override TextSpanner #'bound-details #'right #'text = #"hhhh"
c4^\startTextSpan c c c \stopTextSpan
```



`arrow` Assigner à cette sous-propriété la valeur `vrai` (`#t`) produira une terminaison en pointe de flèche.

`padding` Cette sous-propriété contrôle l'espace entre les extrémités de la ligne, telles que définies, et la réalité. Sans ce léger décalage, le début et la fin d'un glissando seraient en plein milieu des têtes de note.

Voir aussi

Référence du programme : Section “TextSpanner” dans *Référence des propriétés internes*, Section “Glissando” dans *Référence des propriétés internes*, Section “VoiceFollower” dans *Référence des propriétés internes*, Section “TrillSpanner” dans *Référence des propriétés internes*, Section “line-spanner-interface” dans *Référence des propriétés internes*.

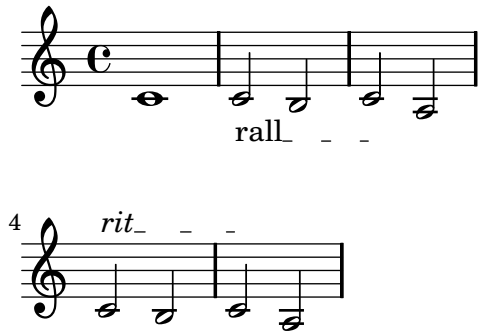
Exemples : Section “Expressive marks” dans *Exemples de code*.

Extensions de texte

Certaines indications d’interprétation comme *rallentando* ou *accelerando* s’incrivent en toutes lettres et se prolongent sur plusieurs mesures grâce à une ligne pointillée. Les commandes `\startTextSpan` et `\stopTextSpan`, respectivement attachées à la première et à la dernière note qu’elles concernent, déterminent l’envergure de ces prolongateurs, ou extenseurs.

La chaîne à imprimer, ainsi que son style, sont définis par des propriétés, comme ici :

```
c1
\textSpannerDown
\override TextSpanner #'bound-details #'left #'text =
  \markup { \upright "rall" }
c2\startTextSpan b c\stopTextSpan a
\break
\textSpannerUp
\override TextSpanner #'bound-details #'left #'text =
  \markup { \italic "rit" }
c2\startTextSpan b c\stopTextSpan a
```



Commandes prédéfinies

`\textSpannerUp`, `\textSpannerDown`, `\textSpannerNeutral`.

Propriétés couramment modifiées

Pour obtenir une ligne pleine, utilisez

```
\override TextSpanner #'style = #'line
```

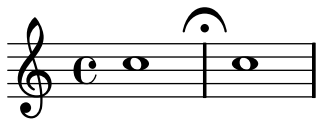
Voir aussi

Référence du programme : Section “TextSpanner” dans *Référence des propriétés internes*.

Indications textuelles

La commande `\mark` est tout d'abord conçue pour les Section 1.2.5.4 [Indications de repère], page 43. Elle peut néanmoins servir à insérer des signes de coda ou de segno, ou bien un point d'orgue, au dessus d'une barre de mesure. Couplez-la alors à la commande `\markup` pour avoir accès au symbole approprié (ils sont répertoriés dans Section B.4 [La fonte Feta], page 196).

```
c1 \mark \markup { \musicglyph #"scripts.ufermata" }
c1
```



Le résultat de `\mark` n'apparaîtra que sur la portée supérieure d'un système. Si vous introduisez la commande `\mark` au moment d'une barre de mesure, la marque se placera au dessus de la barre. Si vous y faites appel au milieu d'une mesure, cette marque sera positionnée entre les notes. Si elle intervient en début de ligne, elle sera placée juste avant la première note de cette portée. Enfin, une marque qui tomberait sur un saut de ligne sera imprimée au début de la ligne suivante. Au cas où il n'y aurait pas de ligne à suivre, la marque ne sera pas imprimée.

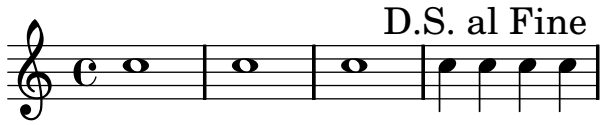
Propriétés couramment modifiées

Pour imprimer une marque à la fin de la portée en cours, procédez ainsi :

```
\override Score.RehearsalMark
  #'break-visibility = #begin-of-line-invisible
```

`\mark` est souvent bien utile pour porter une indication à la fin d'une mesure. Pensez alors à modifier la propriété `#'self-alignment`.

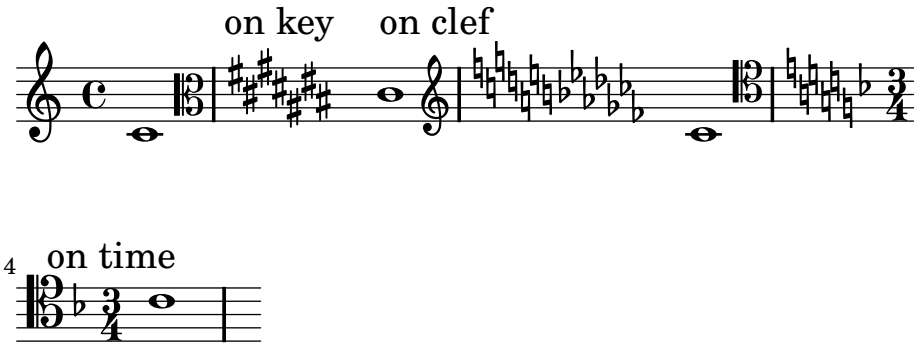
```
\override Score.RehearsalMark
  #'break-visibility = #begin-of-line-invisible
c1 c c c4 c c c
\once \override Score.RehearsalMark #'self-alignment-X = #right
\mark "D.S. al Fine "
```



Les indications textuelles peuvent s'aligner par rapport à d'autres objets que des barres de mesure, tels que l'armure, la clé ou le chiffre de mesure :

```
\relative {
  c1
  \key cis \major
  \clef alto
  \override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
  \mark "on key"
  cis
  \key ces \major
  \override Score.RehearsalMark #'break-align-symbols = #'(clef)
  \clef treble
  \mark "on clef"
```

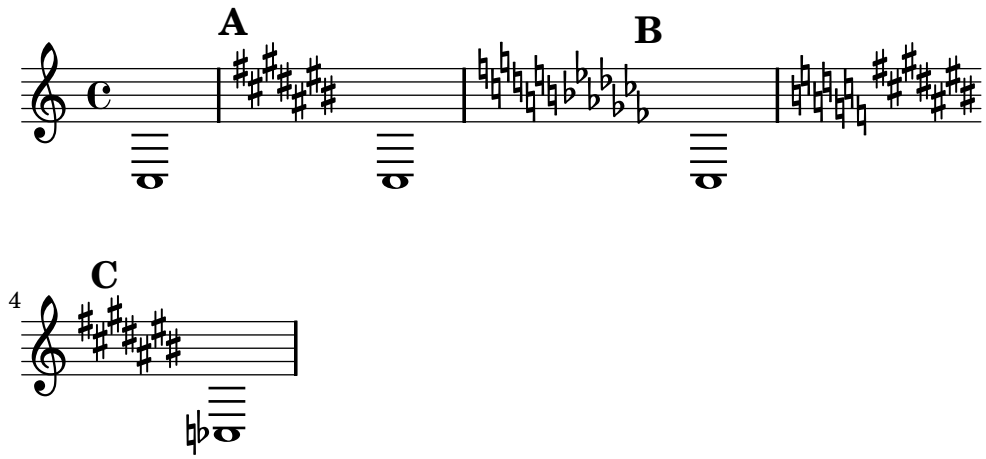
```
  ces
  \override Score.RehearsalMark #'break-align-symbols = #'(time-signature)
  \key d \minor
  \clef tenor
  \time 3/4
  \mark "on time"
  c
}
```



Les symboles pris en charge par `break-align-symbols` sont : `ambitus`, `breathing-sign`, `clef`, `custos`, `staff-bar`, `left-edge`, `key-cancellation`, `key-signature`, et `time-signature`.

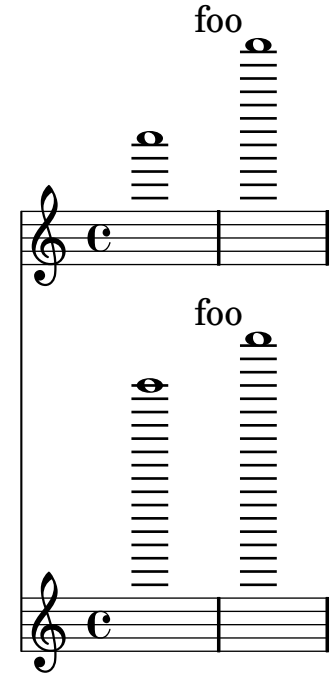
Par défaut, les indications textuelles sont alignées avec le milieu des objets de notation. Bien entendu, vous pouvez modifier les propriétés `break-align-anchor-alignment` ou `break-align-anchor` des objets en question pour obtenir un autre résultat.

```
{
  \override Score.RehearsalMark #'break-align-symbols = #'(key-signature)
  c1
  \key cis \major

  % La marque sera alignée sur le côté gauche de l'armure
  \once \override Staff.KeySignature #'break-align-anchor-alignment = #LEFT
  \mark \default
  cis1
  \key ces \major

  % La marque sera alignée sur le côté droit de l'armure
  \once \override Staff.KeySignature #'break-align-anchor-alignment = #RIGHT
  \mark \default
  ces1
  \key cis \major

  %  La marque sera alignée sur le côté gauche de l'armure,
  % puis décalée vers la droite de 2 unités.
  \once \override Staff.KeySignature #'break-align-anchor = #2
  \mark \default
  ces1
}
```



Bien que ces indications textuelles ne soient habituellement imprimées qu'au niveau de la portée supérieure, vous pouvez forcer leur affectation à chacune des portées :

```
{
  \new Score \with {
    \remove "Mark_engraver"
  }
  <<
    \new Staff \with {
      \consists "Mark_engraver"
    }
    { c''1 \mark "foo" c'' }
    \new Staff \with {
      \consists "Mark_engraver"
    }
    { c'1 \mark "foo" c' }
  >>
}
```



Voir aussi

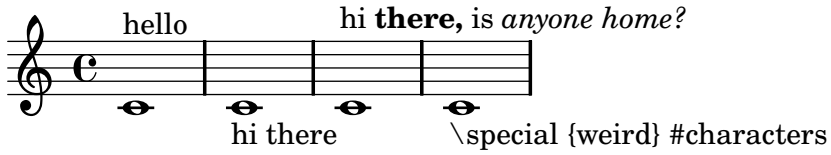
Référence du programme : Section “RehearsalMark” dans *Référence des propriétés internes*.

1.8.2 Mise en forme du texte

Introduction aux étiquettes de texte

La commande `\markup` permet d’ajouter du texte. Vous pouvez y inclure des commandes, précédées d’un antislash `\` ; les caractères `\` et `#` doivent être encadrés de guillemets informatiques `"`.

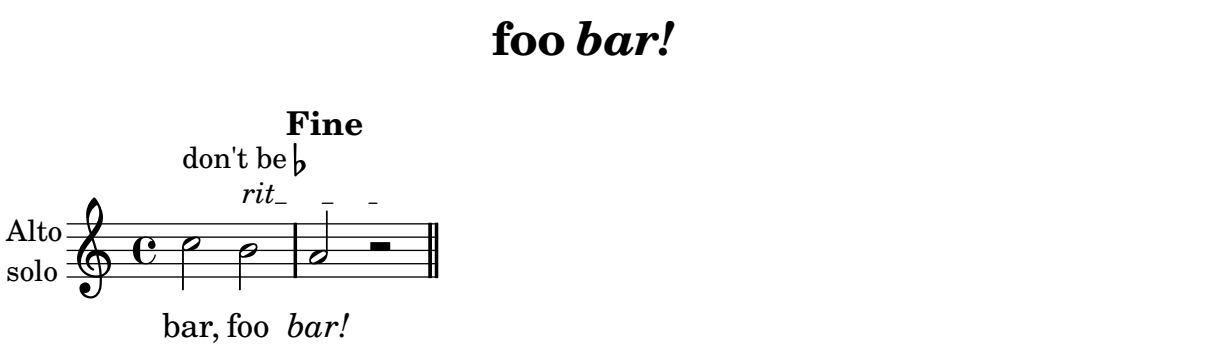
```
c1^\markup { hello }
c1_\markup { hi there }
c1^\markup { hi \bold there, is \italic {anyone home?} }
c1_\markup { "\special {weird} #characters" }
```



Pour une liste des différentes commandes disponibles, consultez Section B.6 [Text markup commands], page 196.

`\markup` est avant tout conçu pour gérer les Section “TextScript” dans *Référence des propriétés interne*s, mais rien ne s’oppose à son utilisation pour traiter du texte avec LilyPond.

```
\header{ title = \markup{ \bold { foo \italic { bar! } } } }
\score{
  \relative c'' {
    \override Score.RehearsalMark
      #'break-visibility = #begin-of-line-invisible
    \override Score.RehearsalMark #'self-alignment-X = #right

    \set Staff.instrumentName = \markup{ \column{ Alto solo } }
    c2^\markup{ don't be \flat }
    \override TextSpanner #'bound-details #'left #'text = \markup{\italic rit }
    b2\startTextSpan
    a2\mark \markup{ \large \bold Fine }
    r2\stopTextSpan
    \bar "||"
  }
  \addlyrics { bar, foo \markup{ \italic bar! } }
}
```



La commande `\markup` peut intervenir à tout moment, y compris en dehors d'un bloc `\score`. Voir à ce sujet Section 3.1.2 [Plusieurs partitions dans un même ouvrage], page 167.

```
\markup{ Here is some text. }
```

Here is some text.

Le *markup* de l'exemple précédent montre comment utiliser les commandes de changement de police. Les commandes `\bold` et `\italic` n'affectent que le premier mot qui les suit ; encadrez les par des accolades si vous désirez que ces commandent s'appliquent à plusieurs mots.

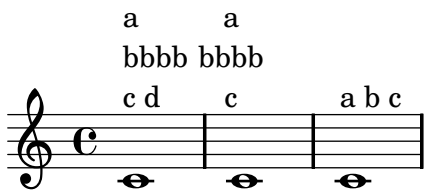
```
\markup { \bold { c'est moi } }
```

Une bonne habitude à prendre consiste à utiliser des accolades même pour un seul mot, comme ici :

```
\markup { qui \italic { est } là ? }
```

En mode *markup*, vous pouvez composer des expressions comme en mathématiques, des documents XML ou bien les expressions musicales. Vous pouvez empiler ces expressions grâce à la commande `\column`, ou les centrer par rapport à leur milieu avec `\center-column`.

```
c1^\markup { \column { a bbbb \line { c d } } }
c1^\markup { \center-column { a bbbb c } }
c1^\markup { \line { a b c } }
```



Des listes non précédées de commande ne sont pas isolées. Ainsi,

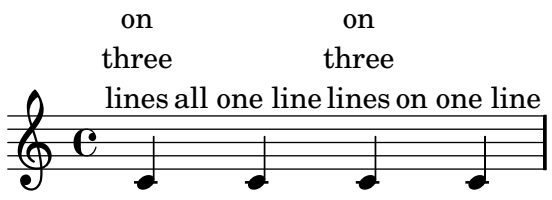
```
\center-column { { a b } { c d } }
```

est la même expression que

```
\center-column { a b c d }
```

L'utilisation des `"` ou de la commande `\line` permet de différencier les listes de mots.

```
\textLengthOn
c4^\markup{ \center-column { on three lines } }
c4^\markup{ \center-column { "all one line" } }
c4^\markup{ \center-column { { on three lines } } }
c4^\markup{ \center-column { \line { on one line } } }
```



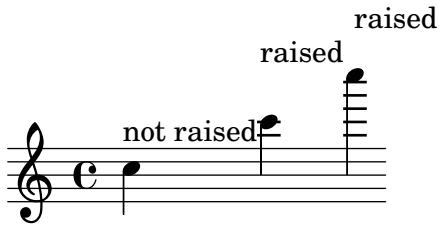
Vous pouvez stocker les étiquettes textuelles en tant que variables, et attacher ces identificateurs à des notes, comme

```
allegro = \markup { \bold \large { Allegro } }
 { a^\allegro b c d }
```

Certains objets possèdent leurs propres procédures d'alignement, qui annuleront toute spécification d'alignement que vous pourriez affecter à un argument de type *markup* que vous leur auriez fourni. Par exemple, les Section "RehearsalMark" dans *Référence des propriétés internes* sont centrées horizontalement ; de fait, utiliser `\mark \markup { \left-align .. }` sera sans effet.

Par ailleurs, le placement vertical n'est effectué qu'après la création de l'objet *étiquette textuelle*. Si donc vous souhaitez déplacer une étiquette, il vous faudra utiliser la propriété `#'padding` ou créer un « point d'ancrage » à l'intérieur même de l'étiquette (généralement avec `\hspace #0`).

```
\textLengthOn
c'4^\markup{ \raise #5 "not raised" }
\once \override TextScript #'padding = #3
c'4^\markup{ raised }
c'4^\markup{ \hspace #0 \raise #1.5 raised }
```



Certaines situations particulières, telles que les indications de nuance, possèdent des propriétés prédéfinies quant à leur police. Nous vous conseillons, en pareil cas, de réinitialiser ces propriétés en utilisant `normal-text`. Pour plus d'informations, consultez Section B.6 [Text markup commands], page 196.

Voir aussi

Dans ce manuel : Section B.6 [Text markup commands], page 196.

Référence du programme : Section "TextScript" dans *Référence des propriétés internes*.

Fichiers d'initialisation : '`scm/new-markup.scm`'.

Problèmes connus et avertissements

Le crénage ou la génération de ligatures ne sont accessibles que lors d'un retraitement par TEX. Dans ce cas, LilyPond n'en tient pas compte, et l'espacement de tels textes sera trop large.

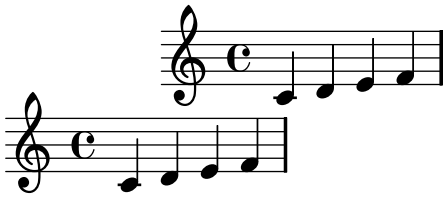
Les erreurs de syntaxe sont peu loquaces.

Partitions emboîtées

Rien ne s'oppose à ce qu'une étiquette ne comporte de la musique. Il suffit que l'expression en question contienne un bloc `\score` et un bloc `\layout`.

```
\relative {
  c4 d^\markup {
    \score {
      \relative { c4 d e f }
      \layout { }
```

```
    }
  }
  e f
}
```



Texte avec sauts de page

Alors que `\markup` s'utilise pour traiter un bloc de texte insécable, `\markuplines` permet, employé en tête de partition, d'obtenir un bloc de lignes réparties différemment et au cas où sur plusieurs pages.

```
\markuplines {
  \justified-lines {
    Un long texte constitué de lignes justifiées.
    ...
  }
  \justified-lines {
    Un autre grand paragraphe justifié.
    ...
  }
  ...
}
```

`\markuplines` prend en argument une liste de lignes de texte, qui peut elle-même consister en une suite de commandes générant à leur tour des lignes de texte, comme ici :

```
\markuplines {
  \line { ... }    % une ligne alignée à gauche
  \fill-line { \line { ... } } % une ligne centrée
  \wordwrap-lines { ... } % une liste de lignes alignées à gauche
  \justified-lines { ... } % une liste de lignes justifiées
}
```

Les différentes commandes permettant de générer des listes de lignes se trouve dans Section B.7 [Text markup list commands], page 233.

Voir aussi

Dans ce manuel : Section B.7 [Text markup list commands], page 233, Section 6.4.4 [Définition d'une nouvelle commande de liste de marqueurs], page 194.

Commandes prédéfinies

`\markuplines`

1.8.3 Fontes

C'est en jouant sur les propriétés des objets décrites ci-après que vous pourrez sélectionner une police parmi les familles de fontes préconfigurées. LilyPond utilise par défaut la police musicale

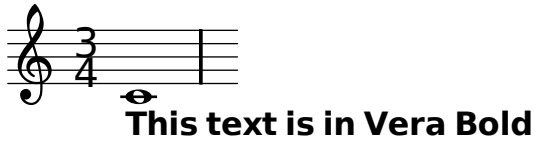
feta. Pour le texte, les polices sont sélectionnées par Pango/Fontconfig. C'est New Century Schoolbook qui sert de police sérif par défaut, et celles définies lors de l'installation de Pango pour ce qui est du sans-serif et du 'typewriter'.

- `font-encoding` symbolise le tracé des glyphes. N'utilisez cette propriété que pour traiter des éléments non textuels, comme :

  `fetaBraces` pour les accolades de partition pianistique, `fetaMusic` pour de la musique (y compris musique ancienne), `fetaDynamic` pour les nuances et `fetaNumber` pour les chiffres.
- `font-family` symbolise les différentes familles de police : `roman` (Computer Modern), `sans-serif` et `typewriter` (espacement fixe).
- `font-shape` symbolise le style des caractères. En pratique, chaque famille de police dispose de `italic`, `caps` (petites capitales) ou `upright` (droit).
- `font-series` symbolise le niveau de gras des caractères. Chaque style dispose, pour chaque famille, de `medium` et `bold` (gras).

Les variantes ci-dessus mentionnées font référence à une feuille de style prédéfinie. Vous pouvez cependant faire appel à une autre police, en utilisant la propriété `font-name` :

```
{
  \override Staff.TimeSignature #'font-name = #"Charter"
  \override Staff.TimeSignature #'font-size = #2
  \time 3/4
  c'1_\markup {
    \override #'(font-name . "Vera Bold")
      { This text is in Vera Bold }
  }
}
```



Vous pouvez utiliser n'importe quelle police, du moment qu'elle est accessible par Pango/Fontconfig. Pour obtenir la liste de toutes les polices disponibles sur votre machine, lancez

```
lilypond -dshow-available-fonts blabla
```

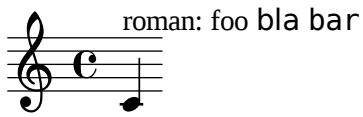
(quel qu'il soit, le dernier argument est obligatoire).

La propriété `font-size` permet de régler la taille de la police. La taille effective que vous obtiendrez dépend de `text-font-size` tel que défini dans le bloc `\paper`.

Vous pouvez aussi changer la police par défaut au niveau du document. Il suffit alors de faire appel à `make-pango-font-tree` au sein du bloc `paper`. Vous définirez alors la police à utiliser pour du texte respectivement en roman, sans serif et monospace, comme ici :

```
\paper  {
  myStaffSize = #20

  #(define fonts
    (make-pango-font-tree "Times New Roman"
                          "Nimbus Sans"
                          "Luxi Mono"
                           (/ myStaffSize 20)))
}
```

```
{
  c'^\markup { roman: foo \sans bla \typewriter bar }
}
```



Voir aussi

Exemples : Section “Text” dans *Exemples de code*.

2 Notation spécialisée

Ce chapitre explique comment créer la notation musicale spécifique à certains intruments ou certaines époques.

2.1 Musique vocale

Dans la mesure où les fichiers LilyPond sont constitués de texte, traiter de la musique vocale demande de prendre en compte deux spécificités :

- Les paroles sont saisies comme étant du texte, non des notes. Ainsi, le code `d` sera interprété comme une syllabe, et non comme la note ré (D pour les non latinistes).
- Les paroles doivent s'aligner avec les notes de la mélodie correspondante.

Plusieurs solutions existent pour ajouter des paroles ; nous les examinerons par degré croissant de complexité.

Propriétés couramment modifiées

S'assurer que tous les éléments textuels et les paroles respectent les marges du document requiert des calculs relativement lourds ; c'est la raison pour laquelle LilyPond, par défaut, ne s'en préoccupe pas. Vous pouvez cependant l'y forcer en ajoutant le code

```
\override Score.PaperColumn #'keep-inside-line = ##t
```

Pour que les paroles évitent également les barres de mesure, ajoutez

```
\layout {
  \context {
    \Lyrics
      \consists "Bar_engraver"
      \consists "Separating_line_group_engraver"
      \override BarLine #'transparent = ##t
  }
}
```

2.1.1 Vue d'ensemble de la musique vocale

2.1.1.1 Références en matière de musique vocale

Le *parlato* est du texte scandé en rythme, mais sans hauteurs définies ; il est indiqué par des notes en croix, à l'instar des percussions — voir [Têtes de note spécifiques], page 16.

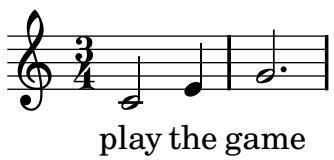
2.1.1.2 Écriture de chants simples

Le plus simple pour ajouter des paroles à une mélodie est d'adjoindre

```
\addlyrics { les paroles }
```

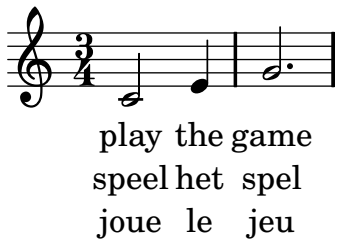
à la mélodie. En voici un exemple :

```
\time 3/4
\relative { c2 e4 g2. }
\addlyrics { play the game }
```



On peut ajouter davantage de couplets en multipliant le nombre de sections `\addlyrics`.

```
\time 3/4
\relative { c2 e4 g2. }
\addlyrics { play the game }
\addlyrics { speel het spel }
\addlyrics { joue le jeu }
```



Cependant, la commande `\addlyrics` ne peut gérer les constructions polyphoniques. Dans ce cas, mieux vaut employer `\lyricsto` et `\lyricmode`, comme nous le verrons dans Section 2.1.1.3 [Saisie des paroles], page 106.

2.1.1.3 Saisie des paroles

Il existe un mode de saisie spécialement adapté aux paroles. On l'introduit avec le mot-clé `\lyricmode`, ou en utilisant `\addlyrics` ou `\lyricsto`. Ce mode vous permet de saisir des paroles, ainsi que leur ponctuation, et le caractère `d` ne sera plus interprété comme une note, mais comme une syllabe. Les syllabes sont saisies comme des notes, mais les hauteurs sont alors remplacées par du texte. Exemple avec une comptine anglaise :

```
\lyricmode { Twin-4 kle4 twin- kle litt- le star2 }
```

Ici on a choisi de saisir explicitement la durée de chaque syllabe. Cependant, il est aussi possible d'attribuer automatiquement chaque syllabe à une note d'une mélodie ou d'une voix existante, avec `\addlyrics` ou `\lyricsto`.

Dans les paroles, un mot ou une syllabe commence par une lettre de l'alphabet, et se termine par un espace (éventuellement précédé d'un chiffre). Toute syllabe doit donc être séparée d'une autre par un espace, tout autre caractère étant considéré comme partie intégrante de cette même syllabe. L'exemple suivant comporte une faute de frappe évidente :

```
\lyricmode { lah- lah}
```

la dernière syllabe contient une `}`, il y a de fait un défaut de parité avec l'accolade ouvrante, et la compilation échouera fort probablement.

De la même manière, un point concluant une suite de lettres sera partie intégrante de la chaîne résultante. Par conséquent, il est impératif d'insérer des espaces lorsque vous modifiez les propriétés d'une commande. En d'autres termes, ne saisissez pas

```
\override Score.LyricText #'font-shape = #'italic
```

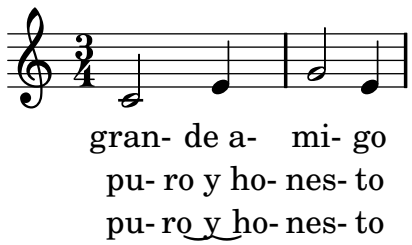
mais plutôt

```
\override Score . LyricText #'font-shape = #'italic
```

Pour attribuer plus d'une syllabe à une même note, vous pouvez mettre ces syllabes entre guillemets, ou bien remplacer l'espace par un caractère souligné (`_`), ou encore utiliser un tilde (`~`) pour obtenir une liaison entre les syllabes.

```
\time 3/4
\relative c' { c2 e4 g2 e4 }
\addlyrics { gran- de_a- mi- go }
\addlyrics { pu- "ro y ho-" nes- to }
```

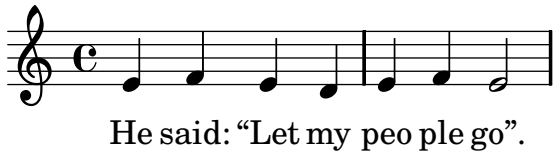
```
\addlyrics { pu- ro~y~ho- nes- to }
```



Cette liaison adaptée aux paroles correspond au caractère Unicode U+203F, et n'apparaîtra dans la partition que s'il existe une police incluant ce symbole (par exemple DejaVuLGC) installée sur le système.

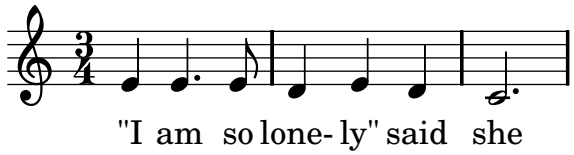
Pour utiliser des lettres accentuées ou des caractères spéciaux — cœurs ou guillemets inversés par exemple — il suffit de les insérer dans le fichier et de veiller à sauvegarder ce dernier avec le codage UTF-8. Voir à ce sujet Section 3.3.3 [Codage du texte], page 168 pour plus de détails.

```
\relative c' { e4 f e d e f e2 }
\addlyrics { He said: \Let my peo ple go". }
```



Pour utiliser des guillemets informatiques standard, faites-les précéder d'une barre oblique inverse :

```
\relative c' { \time 3/4 e4 e4. e8 d4 e d c2. }
\addlyrics { "\"I" am so lone- "ly\"" said she }
```



Expliquer exactement comment LilyPond repère le début d'un mot en mode paroles (Lyrics) est quelque peu compliqué.

En mode Lyrics, un mot peut commencer par : tout caractère alphabétique, `_`, `?`, `!`, `:`, `'`, un des codes de contrôle `^A` à `^F` et `^Q` à `^W`, `^Y`, `^^`, tout caractère ASCII de code strictement supérieur à 127, ou enfin un des symboles `` ` ``, `'`, `"`, ou `^`, s'il est précédé d'une barre oblique inverse.

Pour définir un identificateur contenant des paroles, il faut utiliser la fonction `lyricmode`.

```
verseOne = \lyricmode { Joy to the world the Lord is come }
\score {
  <<
    \new Voice = "one" \relative c'' {
      \autoBeamOff
      \time 2/4
      c4 b8. a16 g4. f8 e4 d c2
    }
    \addlyrics { \verseOne }
```

```
  >>
}
```

Voir aussi

Référence du programme : Section “LyricText” dans *Référence des propriétés internes*, Section “LyricSpace” dans *Référence des propriétés internes*.

2.1.1.4 Travail avec des paroles et variables

La fonction `\lyricmode` permet de définir une variable pour les paroles. Point n’est besoin de spécifier les durées si vous utilisez `\addlyrics` ou `\lyricsto` lorsque vous y faites référence.

```
verseOne = \lyricmode { Joy to the world the Lord is come }
\score {
 <<
   \new Voice = "one" \relative c'' {
     \autoBeamOff
     \time 2/4
     c4 b8. a16 g4. f8 e4 d c2
   }
   \addlyrics { \verseOne }
 >>
}
```

Pour une organisation différente ou plus complexe, mieux vaut commencer par définir la hiérarchie des portées et des paroles,

```
\new ChoirStaff <<
  \new Voice = "soprano" { musique }
  \new Lyrics = "sopranoParoles" { s1 }
  \new Lyrics = "tenorParoles" { s1 }
  \new Voice = "tenor" { musique }
>>
```

puis combiner correctement les mélodies et les paroles :

```
\context Lyrics = sopranoParoles \lyricsto "soprano"
les paroles
```

Le résultat donnera ainsi quelque chose comme

```
<<\new ChoirStaff << définition de la musique >>
 \lyricsto "soprano" etc
 \lyricsto "alto" etc
etc
>>
```

Voir aussi

Référence du programme : Section “LyricCombineMusic” dans *Référence des propriétés internes*, Section “Lyrics” dans *Référence des propriétés internes*.

2.1.2 Alignement des paroles sur une mélodie

Avant d’être imprimées, les paroles sont interprétées par le programme dans le contexte Section “Lyrics” dans *Référence des propriétés internes*.

`\new Lyrics \lyricmode ...`

Il y a deux grandes méthodes pour gérer le placement horizontal des syllabes :

- en alignant automatiquement les paroles sur une mélodie ou une autre voix, en ayant recours à `\addlyrics` ou `\lyricsto` ;
- en affectant explicitement à chaque syllabe une durée, au sein du contexte `\lyricmode`

2.1.2.1 Durée automatique des syllabes

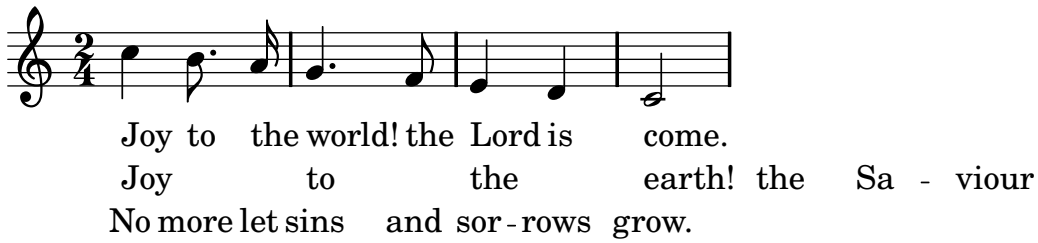
Les paroles peuvent être automatiquement alignées sous une mélodie. Il suffit pour cela de combiner la mélodie et les paroles avec la commande `\lyricsto`.

`\new Lyrics \lyricsto` *nom* `...`

Cette commande adapte les paroles aux notes de la voix (contexte Section “Voice” dans *Référence des propriétés internes* dans le jargon LilyPond) *nom*. Ce contexte `Voice` doit exister avant l'indication des paroles avec `\lyricsto`. La commande `\lyricsto` introduit automatiquement le mode `\lyricmode`, donc dans ce cas vous n'avez pas à ajouter vous-même `\lyricmode`.

L'exemple suivant récapitule les différentes manières de saisir des paroles.

```
<<
  \new Voice = "one" \relative c'' {
    \autoBeamOff
    \time 2/4
    c4 b8. a16 g4. f8 e4 d c2
  }
  \new Lyrics \lyricmode { Joy4 to8. the16 world!4. the8 Lord4 is come.2 }
  \new Lyrics \lyricmode { Joy to the earth! the Sa -- viour reigns. }
  \new Lyrics \lyricsto "one" { No more let sins and sor -- rows grow. }
>>
```



Le deuxième couplet n'est pas correctement disposé, aucune durée n'ayant été spécifiée. Dans un tel cas, il aurait mieux valu utiliser `\lyricsto`, comme dans le troisième couplet.

La commande `\addlyrics` n'est en fait qu'une simplification de la structure employée par LilyPond pour définir les paroles.

```
{ MUSIQUE }
\addlyrics { PAROLES }
```

est exactement la même chose que

```
\new Voice = "blah" { musique }
\new Lyrics \lyricsto "blah" { PAROLES }
```

2.1.2.2 Durée explicite des syllabes

On peut aussi se passer de `\addlyrics` et de `\lyricsto` pour saisir des paroles. Dans ce cas, les syllabes sont entrées comme des notes — du texte remplaçant les hauteurs — ce qui veut dire que vous devez définir leur durée explicitement :

```
play2 the4 game2.
sink2 or4 swim2.
```

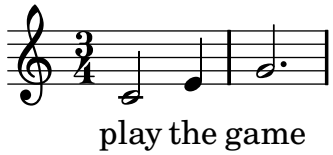
La mélodie associée à ces paroles peut être spécifiée par la propriété `associatedVoice` :

```
\set associatedVoice = #"lala"
```

La valeur que vous attribuez à cette propriété (ici `"lala"`) doit désigner un contexte Section "Voice" dans *Référence des propriétés internes*, sans quoi les mélismes ne seront pas imprimés correctement.

Voici un exemple de cette manière de procéder :

```
<< \new Voice = "melody" {
    \time 3/4
    c2 e4 g2.
 }
 \new Lyrics \lyricmode {
   \set associatedVoice = #"melody"
   play2 the4 game2.
 } >>
```



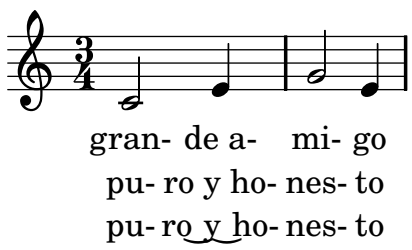
Voir aussi

Référence du programme : Section "Lyrics" dans *Référence des propriétés internes*.

2.1.2.3 Plusieurs syllabes sur une note

Pour attribuer plus d'une syllabe à une même note, vous pouvez les mettre entre guillemets, remplacer une espace par un caractère souligné (_) pour obtenir une espace, ou bien encore utiliser un tilde (~) pour obtenir une liaison entre les syllabes[1].

```
\time 3/4
\relative c' { c2 e4 g2 e4 }
\addlyrics { gran- de_a- mi- go }
\addlyrics { pu- "ro y ho-" nes- to }
\addlyrics { pu- ro~y~ho- nes- to }
```



[1] Une liaison adaptée aux paroles correspond au caractère Unicode U+203F, et n'apparaîtra dans la partition que si le système dispose d'une police installée qui contient ce symbole(par exemple DejaVuLGC).

Voir aussi

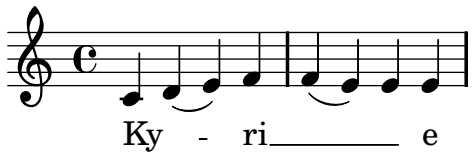
Référence du programme : Section “LyricCombineMusic” dans *Référence des propriétés internes*.

2.1.2.4 Plusieurs notes pour une même syllabe

Parfois, tout particulièrement dans la musique médiévale, plusieurs notes correspondent à une même syllabe. Ces vocalises sont appelées mélismes.

Il est possible d’indiquer tous les mélismes lors de la saisie des paroles. Il suffit pour cela d’utiliser le caractère `_` pour chaque note du mélisme.

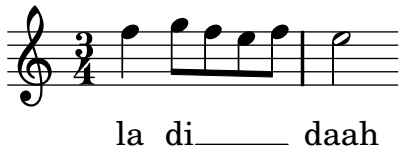
```
{ \set melismaBusyProperties = #'()
 c d( e) f f( e) e e  }
\addlyrics
 { Ky -- _ _ ri __ _ _ _  e }
```



En définissant, comme dans l’exemple ci-dessus, la propriété `melismaBusyProperties`, vous obtiendrez automatiquement toutes les liaisons de tenue et de phrasé requises.

D’autre part, la commande `\lyricsto` arrive aussi à détecter automatiquement les mélismes : une seule syllabe sera placée sous un groupe de notes comprenant des liaisons de tenue ou de phrasé. Au cas où un mélisme doit s’appliquer à des notes non liées, il suffit d’adjoindre `\melisma` à la première note du groupe, et `\melismaEnd` à la dernière :

```
<<
  \new Voice = "lala" {
    \time 3/4
    f4 g8
    \melisma
    f e f
    \melismaEnd
    e2
  }
  \new Lyrics \lyricsto "lala" {
    la di __ daah
  }
>>
```



Enfin, lorsque la fonction de ligature automatique (cf. Section 1.2.4.2 [Définition des règles de ligature automatique], page 35) est désactivée, le fait de les connecter manuellement créera un mélisme.

Vous trouverez un exemple complet de partition pour chœur dans la section Section “Vocal ensembles” dans *Manuel d’initiation*.

Commandes prédéfinies

`\melisma`, `\melismaEnd`

Voir aussi

Exemples : Section “Vocal music” dans *Exemples de code*.

Problèmes connus et avertissements

Certains mélismes ne sont pas détectés automatiquement ; vous devrez alors prolonger vous-même les syllabes concernées.

2.1.2.5 Saut de notes

Cette section n’est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

2.1.2.6 Traits d’union et de prolongation

Un mélisme est indiqué par une ligne horizontale basse centrée entre une syllabe et la suivante. Une telle ligne, que nous appellerons prolongateur ou extenseur, s’obtient en saisissant ‘ `__` ’ (notez les espaces entourant le souligné double).

Un trait d’union séparant deux syllabes d’un même mot s’obtient en saisissant ‘ `--` ’ (notez les espaces entourant le tiret double). Ce trait d’union sera centré entre les deux syllabes et sa longueur sera proportionnelle à l’espace les séparant.

Dans les partitions très serrées, les traits d’union peuvent ne pas être imprimés. Cet inconvénient peut être contrôlé par `minimum-distance` pour gérer l’espace minimum entre deux syllabes, et `minimum-length`, seuil en deçà duquel il n’y a pas de trait d’union.

Voir aussi

Référence du programme : Section “LyricExtender” dans *Référence des propriétés internes*, Section “LyricHyphen” dans *Référence des propriétés internes*

2.1.2.7 Paroles et reprises

Cette section n’est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

2.1.3 Positionnement des paroles

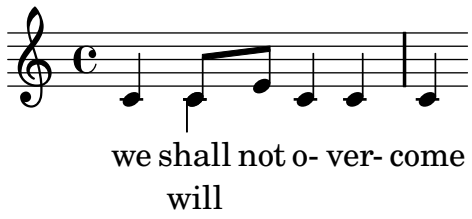
Une même mélodie peut se voir traitée différemment suivant les couplets. La commande `\lyricsto` permet de prendre en compte ces variantes.

2.1.3.1 Paroles alternatives

En donnant un nom à chaque voix et en leur attribuant spécifiquement des paroles, on peut créer des paroles alternatives — également qualifiées de ‘divisi’.

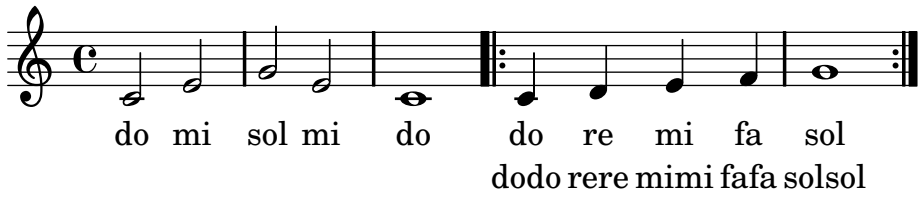
```
\score{ <<
  \new Voice = "melody" {
    \relative c' {
      c4
      <<
        { \voiceOne c8 e }
```

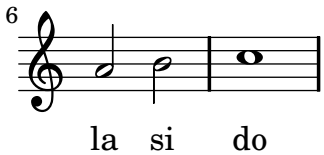
```
        \new Voice = "splitpart" { \voiceTwo c4 }
      >>
      \oneVoice c4 c | c
    }
  }
  \new Lyrics \lyricsto "melody" { we shall not o- ver- come }
  \new Lyrics \lyricsto "splitpart" { will }
>> }
```



Cette astuce permet de faire varier les paroles lorsqu'une section est reprise.

```
\score{ <<
  \new Voice = "melody" \relative c' {
    c2 e | g e | c1 |
    \new Voice = "verse" \repeat volta 2 {c4 d e f | g1 | }
    a2 b | c1}
  \new Lyrics = "mainlyrics" \lyricsto melody \lyricmode {
    do mi sol mi do
    la si do }
  \context Lyrics = "mainlyrics" \lyricsto verse \lyricmode {
   do re mi fa sol }
  \new Lyrics = "repeatlyrics" \lyricsto verse \lyricmode {
   dodo rere mimi fafa solsol }
>>
}
```

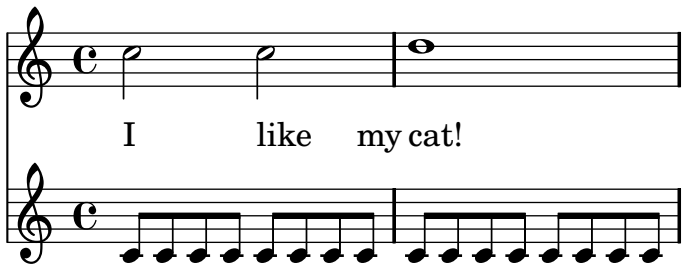




2.1.3.2 Paroles indépendantes des notes

Dans certaines musiques vocales assez complexes, on peut avoir intérêt à imprimer les paroles indépendamment des notes. La mélodie sur laquelle s'aligne les paroles — marquée par le tag `lyricrhythm` dans l'exemple suivant — peut être insérée dans un contexte `Devnull`, ce qui indique à LilyPond de na pas imprimer cette mélodie dans la partition. Seules subsistent alors de cette mélodie les valeurs rythmiques, sur lesquelles s'alignent les paroles.

```
voix = {
  c''2
  \tag #'music { c''2 }
  \tag #'lyricrhythm { c''4. c''8 }
  d''1
}

paroles = \lyricmode { I like my cat! }

<<
  \new Staff \keepWithTag #'music \voix
  \new Devnull="nowhere" \keepWithTag #'lyricrhythm \voix
  \new Lyrics \lyricsto "nowhere" \paroles
  \new Staff { c'8 c' c' c' c' c' c' c'
  c' c' c' c' c' c' c' c' }
>>
```



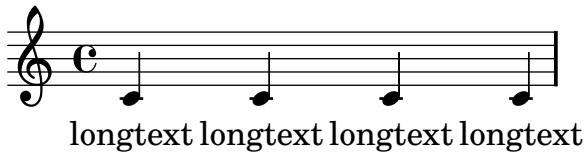
2.1.3.3 Chants

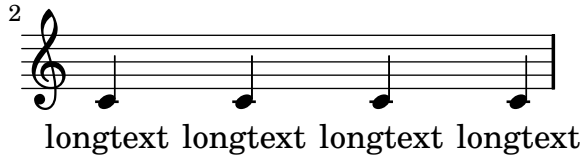
Cette section n'est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

2.1.3.4 Espacement des syllabes

La propriété `#'minimum-distance` de l'objet LyricSpace permet d'accroître l'espacement des paroles.

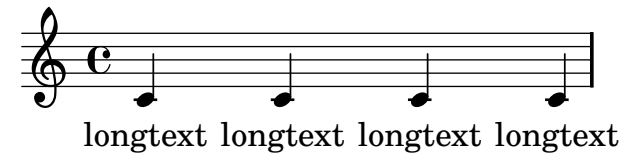
```
{
  c c c c
  \override Lyrics.LyricSpace #'minimum-distance = #1.0
  c c c c
}
\addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
}
```

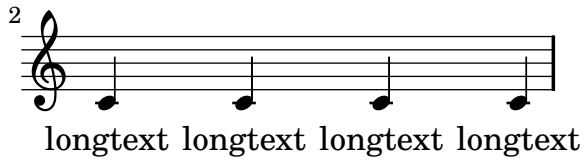




Pour que ce réglage s'applique à toute la partition, définissez-le dans le bloc `\layout`.

```
\score {
  \relative c' {
  c c c c
  c c c c
  }
  \addlyrics {
  longtext longtext longtext longtext
  longtext longtext longtext longtext
  }
  \layout {
    \context {
      \Lyrics
      \override LyricSpace #'minimum-distance = #1.0
    }
  }
}
```





2.1.3.5 Centrage des paroles entre les portées

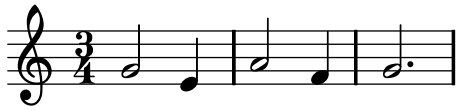
Cette section n'est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

2.1.4 Couplets

2.1.4.1 Numérotation des couplets

On peut ajouter un numéro aux couplets en définissant la variable `stanza` :

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set stanza = "1. "
  Hi, my name is Bert.
} \addlyrics {
  \set stanza = "2. "
  Oh, ché -- ri, je t'aime
}
```



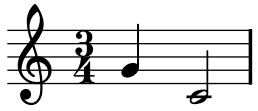
1. Hi, my name is Bert.
2. Oh, ché - ri, je t'aime

Ces numéros se placeront juste avant le début de la première syllabe.

2.1.4.2 Indication de nuance et couplets

Lorsque des couplets ont des nuances différentes, vous pouvez ajouter une nuance devant chaque numéro. L'objet `StanzaNumber` contient tout ce qui se place au début du couplet. Pour des raisons techniques, vous devrez définir la variable `stanza` en dehors du mode `\lyricmode`.

```
texte = {
  \set stanza = \markup { \dynamic "ff" "1. " }
  \lyricmode {
    Big bang
  }
}

<<
  \new Voice = "tune" {
    \time 3/4
    g'4 c'2
  }
\new Lyrics \lyricsto "tune" \texte
>>
```

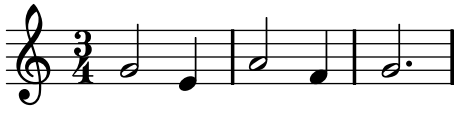


ff **1.** Big bang

2.1.4.3 Indication du personnage et couplets

On peut également ajouter le nom de chaque rôle ; ils s'imprimeront au début de chaque ligne comme les noms d'instruments. Il faut pour cela définir `vocalName`, et `shortVocalName` pour une version abrégée.

```
\new Voice {
  \time 3/4 g2 e4 a2 f4 g2.
} \addlyrics {
  \set vocalName = "Bert "
  Hi, my name is Bert.
} \addlyrics {
  \set vocalName = "Ernie "
  Oh, ché -- ri, je t'aime
}
```



Bert Hi, my name is Bert.
Ernie Oh, ché - ri, je t'aime

2.1.4.4 Rythme différent selon le couplet

Mélismes dans certaines strophes seulement

Il peut survenir que les paroles comportent un mélisme pour l'un des couplets, mais plusieurs syllabes pour d'autres. Une solution consiste à temporairement ignorer les mélismes dans le couplet ayant plus de syllabes. Il suffit pour cela de définir la propriété `ignoreMelismata` à l'intérieur du contexte Lyrics.

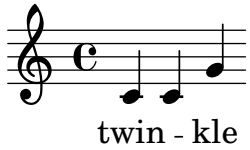
Petit détail qui a son importance : la définition de `ignoreMelismata` doit intervenir une syllabe *avant* les syllabes auxquelles elle s'appliquera :

```
%{
<<
  \relative \new Voice = "lahlah" {
    \set Staff.autoBeaming = ##f
    c4
    \slurDotted
    f8.[( g16])
    a4
  }
  \new Lyrics \lyricsto "lahlah" {
    more slow -- ly
  }
  \new Lyrics \lyricsto "lahlah" {
    \set ignoreMelismata = ##t % applies to "fas"
    go fas -- ter
    \unset ignoreMelismata
    still
  }
>>
%}
```

Ici la fonction `ignoreMelismata` concerne la syllabe 'fas', bien qu'elle ait été définie avant la syllabe 'go'.

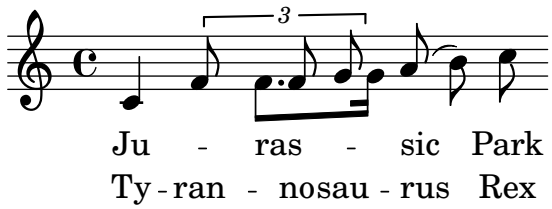
L'inverse est aussi possible : prendre une mélodie syllabique pour en faire un mélisme. Il faudra alors insérer des sauts invisibles `\skip` dans vos paroles. Chaque `\skip` décale le texte suivant d'une note :

```
\relative c' { c c g' }
\addlyrics {
  twin -- \skip 4
  kle
}
```



Basculer vers une mélodie alternative

On peut créer des variations plus complexes à partir d'une mélodie à plusieurs voix. Les paroles peuvent suivre l'une ou l'autre des lignes mélodiques, et même basculer de l'une à l'autre si l'on modifie la propriété `associatedVoice`. Dans cet exemple,



le texte du premier couplet s'aligne sur la mélodie nommée 'lahlah',

```
\new Lyrics \lyricsto "lahlah" {
  Ju -- ras -- sic Park
}
```

Le second couplet est tout d'abord rattaché au contexte `lahlah`, mais bascule sur une autre mélodie sur la syllabe 'ran'. Pour cela, nous utilisons

```
\set associatedVoice = alternative
```

où `alternative` désigne le nom du contexte `Voice` qui contient le triolet.

Encore une fois, cette commande doit être spécifiée une syllabe en avance, c'est-à-dire ici avant la syllabe 'Ty'.

```
\new Lyrics \lyricsto "lahlah" {
  \set associatedVoice = alternative % applies to "ran"
  Ty --
  ran --
  no --
  \set associatedVoice = lahlah % applies to "rus"
  sau -- rus Rex
}
```

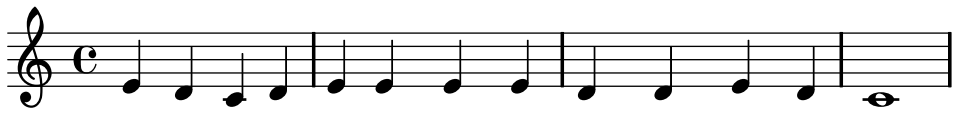
Comme vous pouvez le voir, nous revenons ensuite à l'alignement sur la mélodie d'origine, en assignant à la propriété `associatedVoice` la valeur `lahlah`.

2.1.4.5 Paroles en fin de partition

Il peut parfois s'avérer opportun d'aligner un seul couplet sur la mélodie, et de présenter tous les autres en bloc à la fin du morceau. Ces couplets additionnels peuvent être inclus dans une section `\markup` en dehors du bloc `\score` principal. Vous en trouverez un exemple ci-dessous ; notez également les deux méthodes différentes employées pour indiquer les sauts de ligne, entre les couplets ('verses' en anglais) 2 et 3.

```
melodie = \relative c' {
e d c d | e e e e |
d d e d | c1 |
}

texte = \lyricmode {
\set stanza = "1." Ma- ry had a lit- tle lamb,
its fleece was white as snow.
}

\score{ <<
  \new Voice = "one" { \melodie }
  \new Lyrics \lyricsto "one" \texte
>>
  \layout { }
}
\markup { \column{
  \line{ Verse 2. }
  \line{ All the children laughed and played }
```

```
  \line{ To see a lamb at school. }
  }
}
\markup{
  \wordwrap-string #"
  Verse 3.

  Mary took it home again,

  It was against the rule."
}
```



1. Ma- ry had a lit- tle lamb, its fleece was white as snow.

Verse 2.
All the children laughed and played
To see a lamb at school.

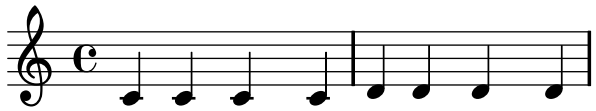
Verse 3.
Mary took it home again,
It was against the rule.

2.1.4.6 Paroles sur plusieurs colonnes en fin de partition

Si les couplets sont vraiment nombreux, il est possible de les imprimer sur plusieurs colonnes. De plus, l'exemple suivant vous montrera comment faire en sorte que le numéro du couplet soit en retrait à gauche, comme c'est traditionnellement le cas.

```
melodie = \relative c' {
  c c c c | d d d d
}

texte = \lyricmode {
  \set stanza = "1." This is verse one.
  It has two lines.
}

\score{ <<
    \new Voice = "one" { \melodie }
    \new Lyrics \lyricsto "one" \texte
   >>
  \layout { }
}

\markup {
  \fill-line {
    \hspace #0.1 % moves the column off the left margin; can be removed if
        % space on the page is tight
     \column {
```

```
      \line { \bold "2."
        \column {
          "This is verse two."
          "It has two lines."
        }
      }
      \hspace #0.1 % ajout d'espace vertical entre les couplets
      \line { \bold "3."
        \column {
          "This is verse three."
          "It has two lines."
        }
      }
    }
    \hspace #0.1  % adds horizontal spacing between columns; if they are
        % still too close, add more " " pairs until the result
        % looks good
     \column {
      \line { \bold "4."
        \column {
          "This is verse four."
          "It has two lines."
        }
      }
      \hspace #0.1 % ajout d'espace vertical entre les couplets
      \line { \bold "5."
        \column {
          "This is verse five."
          "It has two lines."
        }
      }
    }
  \hspace #0.1 % gives some extra space on the right margin; can
      % be removed if page space is tight
  }
}
```



1. This is verse one. It has two lines.

2. This is verse two.
It has two lines.

3. This is verse three.
It has two lines.

4. This is verse four.
It has two lines.

5. This is verse five.
It has two lines.

Voir aussi

Référence du programme : Section “LyricText” dans *Référence des propriétés internes*, Section “StanzaNumber” dans *Référence des propriétés internes*.

2.2 Instruments à clavier

2.2.1 Vue d’ensemble des claviers

Généralités sur les instruments à clavier

Cette section n’est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

Les systèmes de piano comprennent deux portées réunies par une accolade. Les portées sont largement autonomes, mais il arrive que des voix passent de l’une à l’autre. Cette notation sert également à la harpe ou à d’autres instruments à clavier. Le contexte `PianoStaff` est précisément conçu pour gérer la notation spécifique au piano, notamment ces croisements.

Problèmes connus et avertissements

Les nuances ne sont pas centrées verticalement, mais il existe des astuces. Voir à ce sujet le modèle ‘alignement des nuances au piano’ au chapitre Section “Piano templates” dans *Manuel d’initiation*.

Changement de portée manuel

Il est possible de passer d’une portée à l’autre de façon manuelle, au moyen de la commande

```
\change Staff = nomDeLaPortee musique
```

La valeur *nomDeLaPortee* est le nom de la portée sur laquelle va se déplacer la voix courante. Pour des raisons pratiques, on nomme la portée supérieure `"haut"` et la portée inférieure `"bas"`, donc *nomDeLaPortee* désigne habituellement `"haut"` ou `"bas"`. Dans tous les cas, le contexte de portée ainsi utilisé doit exister au préalable. C’est pourquoi il est d’usage de commencer par définir les portées

```
<<
  \new Staff = "haut" {
    \skip 1 * 10  % de façon à prolonger la portée
    }
  \new Staff = "bas" {
    \skip 1 * 10  % idem
    }
>>
```

avant d’insérer une mélodie au moyen d’un contexte `Voice` :

```
\context Staff = bas
  \new Voice { ... \change Staff = haut ... }
```

Changement de portée automatique

Les voix peuvent passer automatiquement d’une portée à l’autre, au moyen de la syntaxe suivante :

```
\autochange ...musique...
```

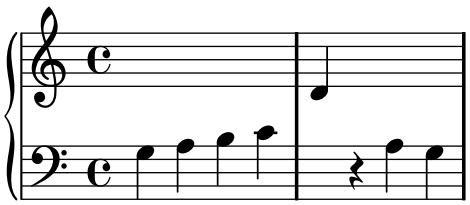
Deux portées seront alors créées au sein du contexte `PianoStaff`, nommées respectivement `up` et `down`. La portée du bas, par défaut, sera en clé de fa.

Une section en mode `\relative` se situant en dehors de la commande `\autochange` n'aura pas d'effet sur les hauteurs de l'expression *musique* ; si on utilise `\relative`, il est donc préférable de mettre `\relative` *après* `\autochange` et non avant :

```
\autochange \relative ... ...
```

La commande `\autochange` bascule les notes d'une portée à l'autre en fonction de leur hauteur (le do du milieu servant de charnière), et place les silences en fonction des notes qui les suivront. Ainsi :

```
\new PianoStaff
  \autochange \relative c'
  {
    g4 a b c d r4 a g
  }
```



Voir aussi

Dans ce même manuel : [Changement de portée manuel], page 121.

Référence du programme : Section "AutoChangeMusic" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

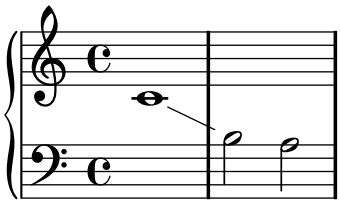
Les changements de portée automatiques n'interviennent pas toujours à l'endroit le plus opportun. Pour un résultat de meilleure qualité, il vaut mieux indiquer soi-même ces changements.

`\autochange` ne peut intervenir à l'intérieur d'une commande `\times`.

Lignes de changement de portée

Lorsqu'une voix change de portée, il est possible d'imprimer automatiquement une ligne reliant les notes, en attribuant à la variable `followVoice` la valeur *vrai* :

```
\new PianoStaff <<
  \new Staff="one" {
    \set followVoice = ##t
    c1
    \change Staff=two
    b2 a
  }
  \new Staff="two" { \clef bass \skip 1*2 }
>>
```



Voir aussi

Référence du programme : Section “VoiceFollower” dans *Référence des propriétés internes*.

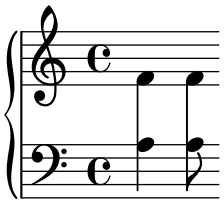
Commandes prédéfinies

`\showStaffSwitch`, `\hideStaffSwitch`.

Hampes et changements de portée

Pour écrire des accords qui enjambent deux portées, on allonge la hampe de l’accord de l’une des deux portées de façon à ce qu’elle rejoigne celle de l’autre portée.

```
stemExtend = {
  \once \override Stem #'length = #10
  \once \override Stem #'cross-staff = ##t
}
noFlag = \once \override Stem #'flag-style = #'no-flag
\new PianoStaff <<
  \new Staff {
    \stemDown \stemExtend
    f'4
    \stemExtend \noFlag
    f'8
  }
  \new Staff {
    \clef bass
    a4 a8
  }
>>
```



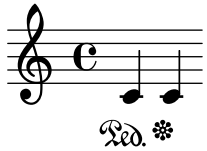
2.2.2 Piano

Pédales de piano

Le piano possède deux pédales, parfois trois, permettant de modifier l’émission du son. Il est possible d’indiquer précisément chacune d’entre elles, en ajoutant à une note ou un accord les commandes suivantes :

	pédale de tenue	pédale *una corda*	pédale tonale
enfoncer	`\sustainOn`	`\unaCorda`	`\sostenutoOn`
relâcher	`\sustainOff`	`\treCorde`	`\sostenutoOff`

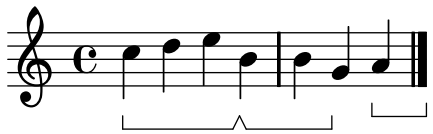
```
c'4\sustainOn c'4\sustainOff
```



Les modalités d'impression de ces indications sont définies par la propriété `pedalXStrings`, *X* étant l'une des trois pédales `Sustain`, `Sostenuto` ou `UnaCorda`. Voyez la référence du programme, section Section "SustainPedal" dans *Référence des propriétés internes*, pour en savoir plus.

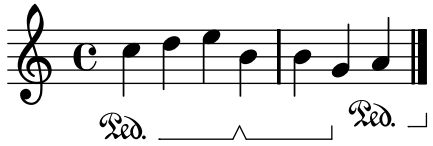
La propriété `pedalSustainStyle` permet différentes notations de pédale, en utilisant des crochets

```
\set Staff.pedalSustainStyle = #'bracket
c\sustainOn d e
b\sustainOff\sustainOn
b g \sustainOff a \sustainOn \bar "|."
```



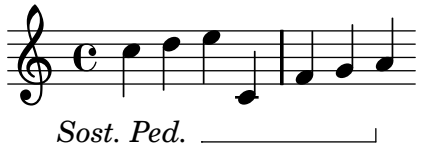
ou en mélangeant indications textuelles et crochets

```
\set Staff.pedalSustainStyle = #'mixed
c\sustainOn d e
b\sustainOff\sustainOn
b g \sustainOff a \sustainOn \bar "|."
```



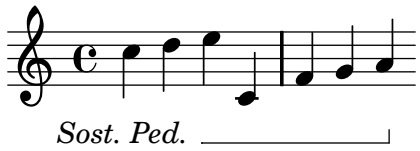
`text` est le style de notation par défaut pour la pédale de tenue — le traditionnel '*Ped.'. La pédale tonale, en revanche, utilise `mixed` par défaut.

```
c\sostenutoOn d e c, f g a\sostenutoOff
```



Il est possible de d'affiner l'apparence d'un crochet de pédale, au moyen des propriétés `edge-width`, `edge-height`, et `shorten-pair`, appliquées aux objets `PianoPedalBracket` — voir la référence du programme, section Section "PianoPedalBracket" dans *Référence des propriétés internes*. Par exemple, on peut étirer le crochet jusqu'à l'extrémité droite de la dernière note :

```
\override Staff.PianoPedalBracket #'shorten-pair = #'(0 . -1.0)
c\sostenutoOn d e c, f g a\sostenutoOff
```



Voir aussi

Dans ce manuel : Section 1.2.1.4 [Liaisons de prolongation], page 22 « laissez vibrer ».

2.2.3 Accordéon

Symboles de jeux

Cette section n'est pas encore encore traduite, veuillez vous reporter à la documentation correspondante en anglais.

2.3 Cordes non frettées

Cette section dispense des informations supplémentaires et utiles à l'écriture pour les cordes frottées.

2.3.1 Vue d'ensemble de la notation pour cordes non frettées

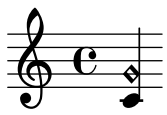
2.3.1.1 Références en matière de cordes non frettées

2.3.2 Instruments à archet

2.3.2.1 Références en matière d'instrument à archet

La plupart des instruments à cordes peuvent produire des harmoniques artificiels, lorsque l'instrumentiste effleure simplement la corde pour une note donnée. Le son émis sera un harmonique, que l'on peut indiquer par `\harmonic`.

```
<c g'\harmonic>4
```



2.3.3 Instruments à cordes pincées

2.3.3.1 Harpe

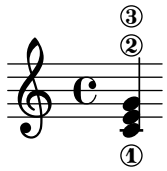
2.4 Instruments à cordes frettées

2.4.1 Vue d'ensemble des cordes frettées

Références en matière de cordes frettées

Indications du numéro de corde

On peut ajouter aux accords les numéros de cordes, en les indiquant avec *number* :



Voir aussi

Référence du programme : Section “StringNumber” dans *Référence des propriétés internes*.

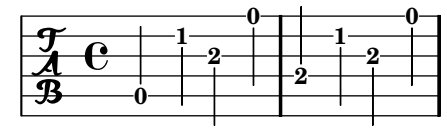
Exemples : Section “Fretted strings” dans *Exemples de code*.

Tablatures par défaut

La notation en tablature est utilisée pour certains instruments à cordes pincées. Les hauteurs n'y sont pas indisuées par des têtes de note, mais par des chiffres qui indiquent sur quelle corde, et à quelle case chaque note doit être jouée. Dans certaines limites, LilyPond vous permet d'écrire des tablatures.

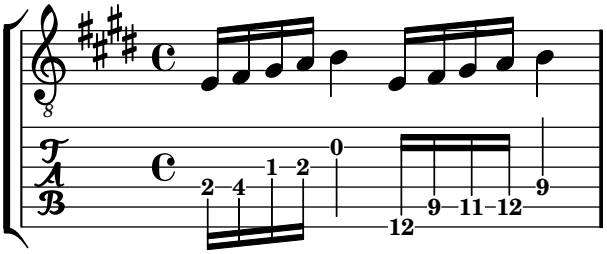
Chaque numéro de corde associé à une note doit être précédé d'une barre oblique inverse (ou « antislash »). Ainsi, `c4\3` donnera un do noire à jouer sur la troisième corde. Par défaut, la première corde est la plus aigüe, et les cordes suivent l'accord traditionnel d'une guitare à six cordes. Les notes sont imprimées sous forme de tablature, dans les contextes Section “TabStaff” dans *Référence des propriétés internes* et Section “TabVoice” dans *Référence des propriétés internes*.

```
\new TabStaff {
  a,4\5 c'\2 a\3 e'\1
  e\4 c'\2 a\3 e'\1
}
```



Quand aucune corde n'est précisée, LilyPond choisit automatiquement la corde où la position est la moins élevée et qui donne un numéro de case supérieur à la valeur de la propriété `minimumFret`. On peut régler cette propriété, selon qu'on désire une position plus ou moins haute. La valeur par défaut de cette propriété est fixée à 0, ce qui correspond à la position la plus basse.

```
e16 fis gis a b4
\set TabStaff.minimumFret = #8
e16 fis gis a b4
```



Propriétés couramment modifiées

Pour obtenir des tablatures où les hampes seront dirigées vers le bas et les ligatures horizontales, réglez le contexte `TabStaff` comme suit :

```
\stemDown
\override Beam #'damping = #100000
```

Voir aussi

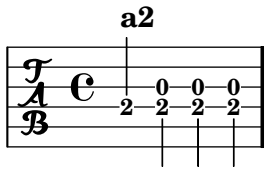
Référence du programme : Section “TabStaff” dans *Référence des propriétés internes*, Section “TabVoice” dans *Référence des propriétés internes*.

Problèmes connus et avertissements

Les accords ne subsissent ausun traitement particulier ; de ce fait, la sélection automatique des cordes peut attribuer une même corde pour deux notes différentes de l’accord.

Afin que `\partcombine` fonctionne avec des tablatures, on doit ajouter au contexte `TabStaff` des voix fantômes :

```
melodie = \partcombine { e4 g g g }{ e4 e e e }
<<
  \new TabStaff <<
    \new TabVoice = "one" s1
    \new TabVoice = "two" s1
    \new TabVoice = "shared" s1
    \new TabVoice = "solo" s1
    { \melodie }
  >>
>>
```

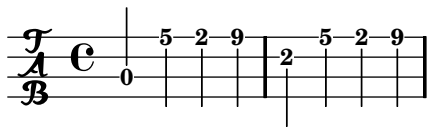


Tablatures personnalisées

Vous pouvez bien sûr accorder vos cordes différemment. Cet accord est enregistré dans la propriété `stringTunings`. La valeur de cette propriété doit être indiquée sous forme d’une liste en langage Scheme, où chaque corde est représentée par un nombre entier qui indique sa hauteur à vide, comptée en demi-tons à partir du do central. Cette propriété permet aussi à LilyPond de déterminer le nombre de cordes.

Dans l’exemple suivant, on a réglé `stringTunings` pour l’accord de la guitare basse, c’est-à-dire mi la ré sol.

```
\new TabStaff <<
  \set TabStaff.stringTunings = #'(-5 -10 -15 -20)
  {
    a,4 c' a e' e c' a e'
  }
>>
```



Toutefois, LilyPond possède des jeux de cordes prédéfinis pour le banjo, la mandoline, la guitare et la guitare basse ; ainsi, l’accord précédent peut également s’indiquer par

```
\set TabStaff.stringTunings = #bass-tuning
```

Le jeu par défaut est celui de la guitare : `guitar-tuning`, c'est-à-dire le célèbre mi la ré sol si mi. D'autres jeux prédéfinis sont `guitar-open-g-tuning`, `mandolin-tuning` et `banjo-open-g-tuning`.

Voir aussi

Vous trouverez une liste complète des jeux prédéfinis dans le fichier '`scm/output-lib.scm`'.

Référence du programme : Section "Tab_note_heads_engraver" dans *Référence des propriétés internes*.

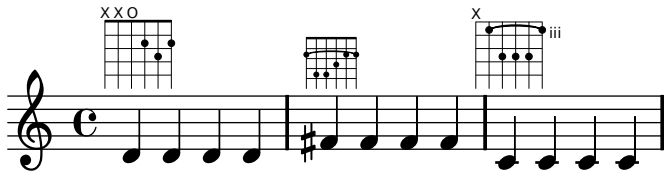
Problèmes connus et avertissements

Aucun effet spécial de guitare n'a été implémenté à ce jour.

Tablatures sous forme d'étiquette

On peut ajouter des diagrammes d'accords au-dessus de n'importe quelle note, en tant qu'objets `\markup`. Ces objets donnent toutes les informations sur le doigté et les éventuels barrés :

```
\new Voice {
  d'^\markup \fret-diagram #"6-x;5-x;4-o;3-2;2-3;1-2;"
  d' d' d'
  fis'^\markup \override #'(size . 0.75) {
    \override #'(finger-code . below-string) {
      \fret-diagram-verbose #'((place-fret 6 2 1) (barre 6 1 2)
                               (place-fret 5 4 3) (place-fret 4 4 4)
                               (place-fret 3 3 2) (place-fret 2 2 1)
                               (place-fret 1 2 1))
    }
  }
  fis' fis' fis'
  c'^\markup \override #'(dot-radius . 0.35) {
    \override #'(finger-code . in-dot) {
      \override #'(dot-color . white) {
        \fret-diagram-terse #"x;3-1-(;5-2;5-3;5-4;3-1-);"
      }
    }
  }
  c' c' c'
}
```



Vous pouvez indiquer vos diagrammes de trois manières différentes : « standard », « terse » ou « verbeux ». Ces trois interfaces produisent des schémas similaires, mais demandent d'entrer plus ou moins d'informations. Vous trouverez tout les détails dans Section B.6 [Text markup commands], page 196.

Par ailleurs, plusieurs propriétés permettent d'ajuster le graphisme à votre convenance. Vous en trouverez les détails dans Section "fret-diagram-interface" dans *Référence des propriétés internes*.

Voir aussi

Exemples : Section "Fretted strings" dans *Exemples de code*.

Doigtés pour la main droite

Les doigtés de la main droite, dans les accords, peuvent être ajoutés au moyen de *note*`-\rightHandFinger` *doigté*

```
<c-\rightHandFinger #1 e-\rightHandFinger #2 >
```

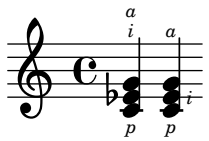
Pour plus de clarté, vous pouvez traduire ou abréger la commande `\rightHandFinger`, par exemple en `\doigtMainDroite` ou même `\MD` :

```
#(define MD rightHandFinger)
```

Propriétés couramment modifiées

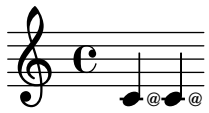
Pour contrôler plus précisément les doigtés de main droite, réglez la propriété `strokeFingerOrientations` :

```
#(define RH rightHandFinger)
{
  \set strokeFingerOrientations = #'(up down)
  <c-\RH #1 es-\RH #2 g-\RH #4 > 4
  \set strokeFingerOrientations = #'(up right down)
  <c-\RH #1 es-\RH #2 g-\RH #4 > 4
}
```



Les lettres utilisées dans ces doigtés sont définies par la propriété `digit-names`, mais on peut bien sûr les changer en utilisant une chaîne de caractères comme argument de `\rightHandFinger`.

```
#(define RH rightHandFinger)
{
  \set strokeFingerOrientations = #'(right)
  \override StrokeFinger #'digit-names = #'#("x" "y" "z" "!" "@")
  <c-\RH #5 >4
  <c-\RH "@">4
}
```



Voir aussi

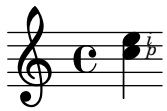
Référence du programme : Section “StrokeFinger” dans *Référence des propriétés internes*

2.4.2 Guitare

Indication de la position et du barré

Les doigtés de la main droite, dans les accords, peuvent être ajoutés au moyen de *note-*`\rightHandFinger` *doigté*

```
<c-\rightHandFinger #1 e-\rightHandFinger #2 >
```



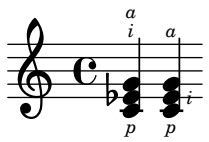
Pour plus de clarté, vous pouvez traduire ou abréger la commande `\rightHandFinger`, par exemple en `\doigtMainDroite` ou même `\MD` :

```
#(define MD rightHandFinger)
```

Propriétés couramment modifiées

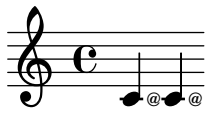
Pour contrôler plus précisément les doigtés de main droite, réglez la propriété `strokeFingerOrientations` :

```
#(define RH rightHandFinger)
{
  \set strokeFingerOrientations = #'(up down)
  <c-\RH #1 es-\RH #2 g-\RH #4 > 4
  \set strokeFingerOrientations = #'(up right down)
  <c-\RH #1 es-\RH #2 g-\RH #4 > 4
}
```



Les lettres utilisées dans ces doigtés sont définies par la propriété `digit-names`, mais on peut bien sûr les changer en utilisant une chaîne de caractères comme argument de `\rightHandFinger`.

```
#(define RH rightHandFinger)
{
  \set strokeFingerOrientations = #'(right)
  \override StrokeFinger #'digit-names = #'#("x" "y" "z" "!" "@")
  <c-\RH #5 >4
  <c-\RH "@">4
}
```



Voir aussi

Référence du programme : Section “StrokeFinger” dans *Référence des propriétés internes*

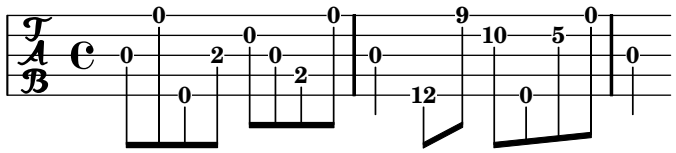
Indicating harmonics and dampened notes

2.4.3 Banjo

2.4.3.1 Tablatures pour banjo

LilyPond permet d’écrire des tablatures de base pour le banjo à cinq cordes. Pour ce faire, pensez à utiliser le format de tablature pour banjo, afin d’avoir le bon nombre de cordes et le bon accord :

```
\new TabStaff <<
  \set TabStaff.tablatureFormat = #fret-number-tablature-format-banjo
  \set TabStaff.stringTunings = #banjo-open-g-tuning
  {
    \stemDown
    g8 d' g'\5 a b g e d' |
    g4 d''8\5 b' a'\2 g'\5 e'\2 d' |
    g4
  }
>>
```



LilyPond connaît un certain nombre de manières d’accorder un banjo : `banjo-c-tuning` (sol do sol si ré), `banjo-modal-tuning` (sol ré sol do ré), `banjo-open-d-tuning` (la ré fa-dièse la ré) et `banjo-open-dm-tuning` (la ré fa la ré).

Tous ces accords peuvent être convertis en accords pour banjo à quatre cordes, si l’on utilise la fonction `four-string-banjo` :

```
\set TabStaff.stringTunings = #(four-string-banjo banjo-c-tuning)
```

Voir aussi

Vous trouverez une liste complète des jeux de cordes prédéfinis pour le banjo dans le fichier ‘`scm/output-lib.scm`’.

2.5 Percussions

2.5.1 Vue d’ensemble des percussions

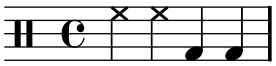
La notation rythmique sert avant tout aux parties de percussions ou de batterie, mais on peut aussi s’en servir à des fins pédagogiques, pour montrer le rythme d’une mélodie.

2.5.1.1 Références en matière de notation pour percussions

2.5.1.2 Notation de base pour percussions

Les parties de percussions peuvent être saisies avec le mode `\drummode`, qui est l'équivalent du mode standard utilisé pour les notes à hauteur déterminée. Chaque instrument de percussion peut avoir, dans le fichier LilyPond, un nom complet et un nom raccourci.

```
\drums {
  hihat hh bassdrum bd
}
```



Ces noms sont inventoriés dans le fichier d'initialisation '`ly/drumpitch-init.ly`'.

Voir aussi

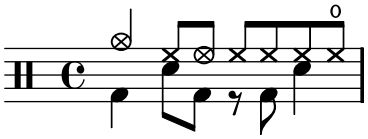
Référence du Programme : Section "note-event" dans *Référence des propriétés internes*.

2.5.1.3 Portée de percussion

Une partie de percussions utilisant plusieurs instruments requiert en général une portée de plusieurs lignes, où chaque hauteur sur la portée représente un instrument à percussion.

Pour saisir cette musique, il faut que les notes soient situées dans des contextes Section "DrumStaff" dans *Référence des propriétés internes* et Section "DrumVoice" dans *Référence des propriétés internes*.

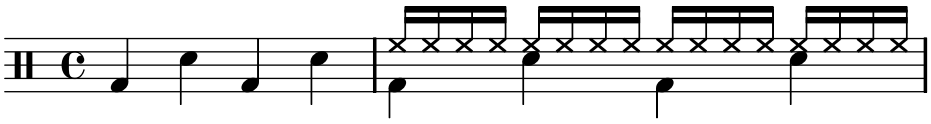
```
haut = \drummode { crashcymbal4 hihat8 halfopenhihat hh hh hh openhihat }
bas = \drummode { bassdrum4 snare8 bd r bd sn4 }
  \new DrumStaff <<
    \new DrumVoice { \voiceOne \haut }
    \new DrumVoice { \voiceTwo \bas }
  >>
```



L'exemple ci-dessus montre une notation polyphonique détaillée. La notation polyphonique abrégée peut être employée lorsque le contexte Section "DrumVoice" dans *Référence des propriétés internes* est spécifié explicitement :

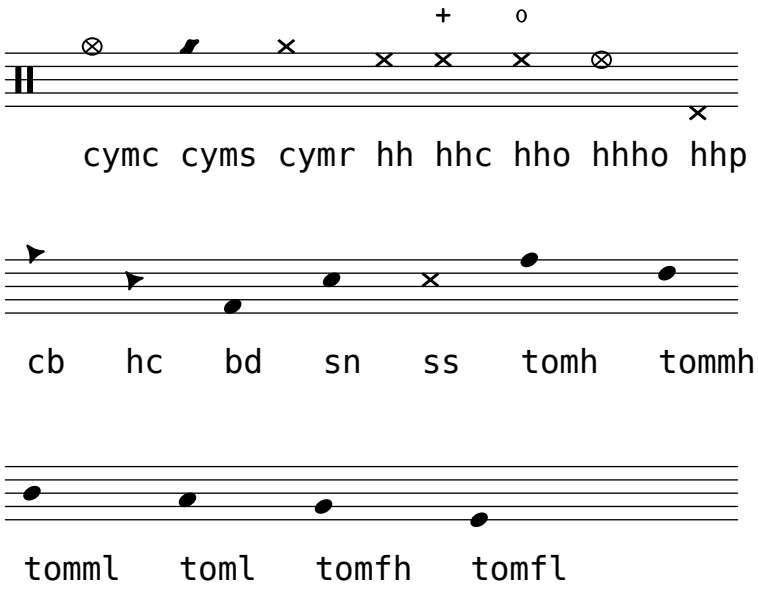
```
\new DrumStaff <<
  \new DrumVoice = "1" { s1 *2 }
  \new DrumVoice = "2" { s1 *2 }
  \drummode {
    bd4 sn4 bd4 sn4
    <<
      { \repeat unfold 16 hh16 }
      \\
      { bd4 sn4 bd4 sn4 }
    >>
  }
```

```
>>
```



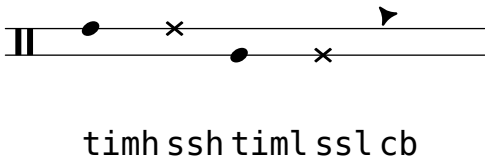
On peut choisir d'autres mises en forme si l'on définit la propriété `drumStyleTable` dans le contexte Section "DrumVoice" dans *Référence des propriétés internes*. Quelques variables prédéfinies sont disponibles :

`drums-style`
: La notation par défaut : une batterie standard, sur une portée de cinq lignes.

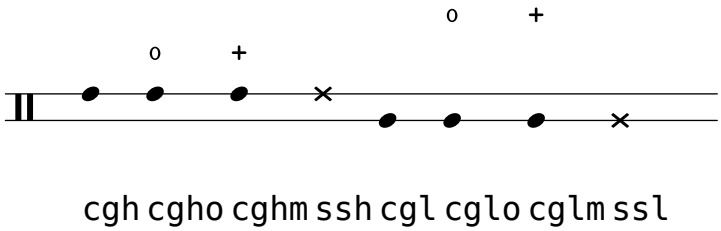


Le plan de la batterie peut inclure jusqu'à six toms différents. Bien sûr, vous n'êtes pas obligé de tous les utiliser si la musique en prévoit moins ; par exemple, les trois toms des lignes du milieu sont `tommh`, `tomml`, et `tomfh`.

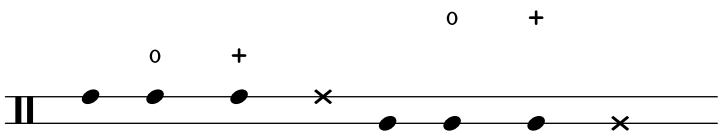
`timbales-style`
: Ce style permet de saisir une partie de timbales, sur une portée à deux lignes.



`congas-style`
: Ce style produit une portée à deux lignes pour une partie de congas.

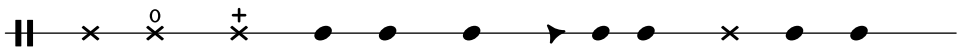


`bongos-style`
: Ce style produit une portée à deux lignes pour une partie de bongos.



boh boho bohm ssh bol bolo bolm ssl

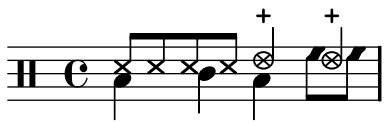
`percussion-style`
: Ce style permet de saisir toute sorte de percussions sur des portées d'une ligne.



tri trio trim gui guis guil cb cl tamb cab mar hc

Cependant, si aucun des styles prédéfinis ne vous convient, il est aisé de définir le vôtre en début de fichier.

```
#(define mydrums '(
         (bassdrum     default   #f         -1)
         (snare        default   #f         0)
         (hihat        cross     #f         1)
         (pedalhihat   xcircle   "stopped"  2)
         (lowtom       diamond   #f         3)))
haut = \drummode { hh8 hh hh hh hhp4 hhp }
bas = \drummode { bd4 sn bd toml8 toml }

\new DrumStaff <<
  \set DrumStaff.drumStyleTable = #(alist->hash-table mydrums)
  \new DrumVoice { \voiceOne \haut }
  \new DrumVoice { \voiceTwo \bas }
>>
```



Voir aussi

Fichier d'initialisation : '`ly/drumpitch-init.ly`'.

Référence du programme : Section "DrumStaff" dans *Référence des propriétés internes*, Section "DrumVoice" dans *Référence des propriétés internes*.

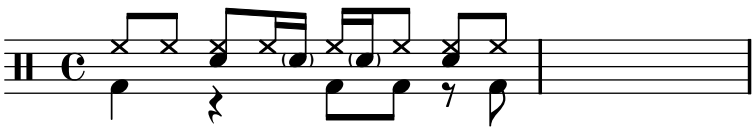
Problèmes connus et avertissements

La bibliothèque MIDI générale ne contient pas les *rimshots* — coups sur le bord de la caisse claire — donc ils sont remplacés par des *sidesticks,* largement équivalents.

2.5.1.4 Notes fantômes

Des notes fantômes peuvent être créées pour les parties de percussion, grâce à la commande `\parenthesize` décrite dans [Parenthèses], page 88. Cependant, le mode `\drummode` n'inclut pas par défaut le graveur `Parenthesis_engraver` qui permet d'imprimer ces signes. Il faut donc l'ajouter explicitement dans la définition du contexte, suivant la manœuvre indiquée dans Section 5.1.2 [La commande set], page 175.

```
\new DrumStaff \with {
  \consists "Parenthesis_engraver"
} <<
  \context DrumVoice  = "1"  { s1 *2 }
  \context DrumVoice  = "2" { s1 *2 }
  \drummode {
    <<
      {
        hh8[ hh] <hh sn> hh16
        < \parenthesize sn > hh < \parenthesize
        sn > hh8 <hh sn> hh
      } \\ {
        bd4 r4 bd8 bd r8 bd
      }
    >>
  }
>>
```



Notez que les commandes `\parenthesize` obligent à ajouter des accords — sous la forme `< >` — autour de chaque élément.

2.6 Instruments à vent

2.6.1 Vue d'ensemble des instruments à vent

Références en matière d'instruments à vent

Doigtés

2.6.2 Cornemuse

2.6.2.1 Définitions pour la cornemuse

LilyPond inclut des définitions spécifiques destinées à la notation pour cornemuse écossaise ; pour les utiliser, il suffit d'ajouter

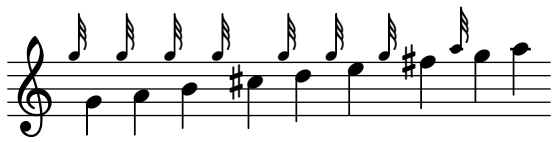
```
\include "bagpipe.ly"
```

en début de fichier. Ainsi, vous bénéficierez de commandes courtes pour les appogiatures spéciales et idiomatiques de la cornemuse. Par exemple, `\taor` est un raccourci pour

```
\grace { \small G32[ d G e] }
```

`bagpipe.ly` prend également en charge les définitions de hauteurs pour la cornemuse ; vous n'avez donc pas à vous soucier d'employer `\relative` ou `\transpose`.

```
\include "bagpipe.ly"
{ \grg G4 \grg a \grg b \grg c \grg d \grg e \grg f \grA g A }
```



La musique pour cornemuse est traditionnellement écrite en ré majeur. Dans la mesure où c'est la seule tonalité possible, l'usage est de ne pas indiquer l'armure explicitement. À cet effet, pensez à commencer votre partition par `\hideKeySignature` – ou bien `\showKeySignature` si, pour une raison ou pour une autre, vous tenez à afficher l'armure.

Des compositions actuelles peuvent ne pas respecter cette tonalité traditionnelle, auquel cas les do et fa devraient être abaissés en utilisant `cflat` ou `fflat` ; ils seront représenté par une note en forme de croix.

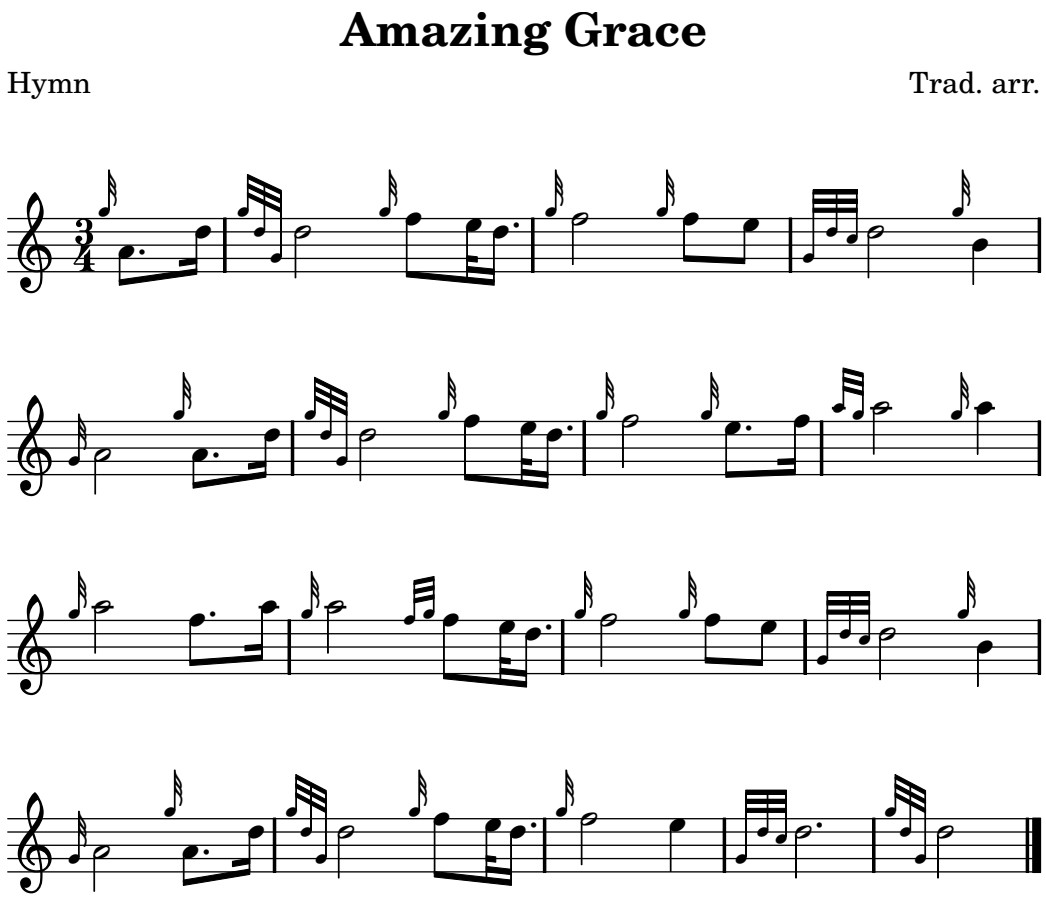
Lorsqu'on joue des œuvres moins cérémonieuses que pour une parade ou un défilé, peut survenir un sol aigu, au doux nom de « piobaireachd », et que l'on indiquera par `gflat`.

2.6.2.2 Exemple pour la cornemuse

Et voici en guise d'exemple, à quoi ressemble le chant populaire *Amazing Grace*, noté dans l'idiome de la cornemuse.

```
\include "bagpipe.ly"
\layout {
  indent = 0.0\cm
  \context { \Score \remove "Bar_number_engraver" }
}

\header {
  title = "Amazing Grace"
  meter = "Hymn"
  arranger = "Trad. arr."
}

{
  \hideKeySignature
  \time 3/4
  \grg \partial 4 a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 \grg e8. f16
  \dblA A2 \grg A4
  \grg A2 f8. A16
  \grg A2 \hdblf f8[ e32 d16.]
  \grg f2 \grg f8 e
  \thrwd d2 \grg b4
  \grG a2 \grg a8. d16
  \slurd d2 \grg f8[ e32 d16.]
  \grg f2 e4
```

```
  \thrwd d2.
  \slurd d2
  \bar "|."
}
```



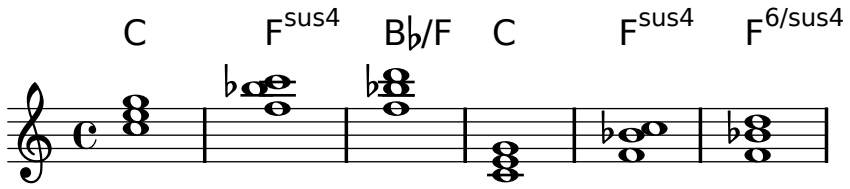
2.7 Notation des accords

2.7.1 Mode accords

Généralités sur le mode accords

LilyPond permet de désigner les accords par leur chiffrage jazz. S'il est possible d'entrer un accord sous sa forme musicale, avec <. . >, on peut aussi le saisir par son nom. Le logiciel traite les accords comme des ensembles de hauteurs, donc les accords peuvent être transposés.

```
twoWays = \transpose c c' {
  \chordmode {
    c1 f:sus4 bes/f
  }
  <c e g>
  <f bes c'>
  <f bes d'>
}

<< \new ChordNames \twoWays
   \new Voice \twoWays >>
```



Cet exemple montre également que les jeux d'instructions qui permettent à LilyPond d'imprimer des accords ne cherchent pas à se montrer intelligents. Ici, le dernier accord n'est pas interprété comme étant renversé.

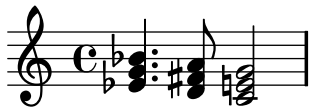
Notez bien que la valeur rythmique des accords doit être indiquée *à l'extérieur* des symboles `<>`.

```
<c e g>2
```

Accords courants

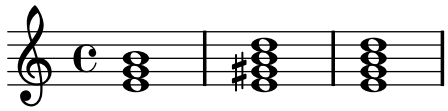
Dans le mode accords, introduit par la commande `\chordmode`, les accords ne sont indiqués que par leur note fondamentale.

```
\chordmode { es4. d8 c2 }
```



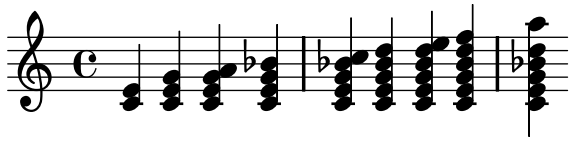
On peut cependant obtenir des accords différents, en utilisant le caractère deux points (`:`).

```
\chordmode { e1:m e1:7 e1:m7 }
```



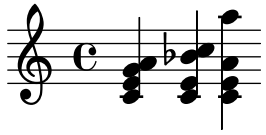
Le nombre accolé à la note fondamentale est considéré comme chiffrage jazz de l'accord, qui est de ce fait construit par un empilement de tierces. Notez l'exception que constitue `c:13` dans lequel la onzième est délibérément omise.

```
\chordmode { c:3 c:5 c:6 c:7 c:8 c:9 c:10 c:11 c:13 }
```



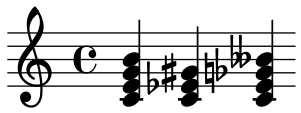
Des accords encore plus complexes peuvent être élaborés en plusieurs étapes séparées. Ainsi, on peut enrichir l'accord par des notes ajoutées, que l'on indique après le chiffrage principal et que l'on sépare par des points :

```
\chordmode { c:5.6 c:3.7.8 c:3.6.13 }
```



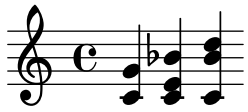
On peut augmenter ou diminuer certains intervalles au moyen des signes `-` ou `+` :

```
\chordmode { c:7+ c:5+.3- c:3-.5-.7- }
```



On peut aussi enlever certaines notes de l'accord, en les spécifiant après un signe `^` — les notes ajoutées doivent être indiquées *avant* les notes à enlever.

```
\chordmode { c^3 c:7^5 c:9^3.5 }
```



Voici les différents chiffrages disponibles, en plus des nombres que nous venons de voir :

`m` Accord mineur. Ce chiffrage minorise la tierce, et la septième s'il y en a une.

`dim` Accord diminué. Ce chiffrage minorise la tierce, diminue la quinte et la septième s'il y en a.

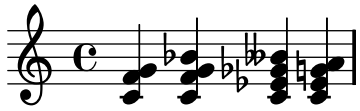
`aug` Accord augmenté. Ce chiffrage augmente la quinte.

`maj` Accord de septième majeure. Ce chiffrage majorise la septième s'il y en a une (dans le cas d'un accord parfait, ce chiffrage est facultatif).

`sus` Accord de suspension. Ce chiffrage supprime la tierce, et y ajoute, suivant que vous spécifiez `2` ou `4`, la seconde ou la quarte.

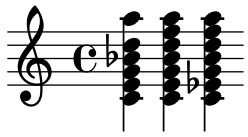
Il est bien sûr possible de mélanger ces chiffrages avec des notes ajoutées.

```
\chordmode { c:sus4 c:7sus4 c:dim7 c:m6 }
```



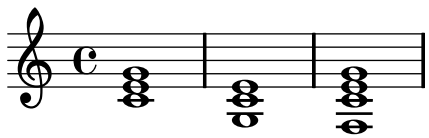
Dans la mesure où un accord de treizième majeure ne sonne pas très bien avec la onzième, la onzième est enlevée automatiquement, à moins que vous ne le spécifiez explicitement.

```
\chordmode { c:13 c:13.11 c:m13 }
```



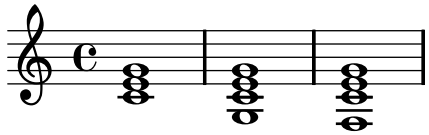
Les accords peuvent être renversés ou combinés avec une note étrangère à la basse, au moyen de *accord*`/`*note*

```
\chordmode { c1 c/g c/f }
```



Si la note de basse précisée appartient à l'accord, la doublure supérieure sera supprimée. Pour l'éviter, utilisez la syntaxe `/+`*note*.

```
\chordmode { c1 c/+g c/+f }
```

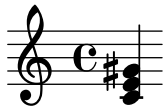


Le mode accords est semblable à `\lyricmode` ou autre, c'est-à-dire que la plupart des commandes sont encore disponibles ; ainsi, `r` ou `\skip` peuvent servir à insérer des silences ou des silences invisibles. De plus, les propriétés des objets peuvent être ajustées ou modifiées.

Problèmes connus et avertissements

Aucun nom de note ne peut être indiqué deux fois dans un accord. Ainsi, dans l'accord suivant, seule la quinte augmentée est prise en compte, car elle est indiquée en dernier :

```
\chordmode { c:5.5-.5+ }
```



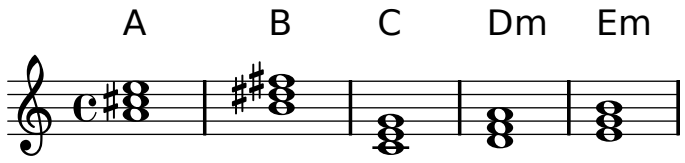
Extensions et altération d'accords

2.7.2 Gravure des accords

Impression de noms d'accords

Les chiffrages d'accords sont liés au contexte Section "ChordNames" dans *Référence des propriétés internes*. Les accords peuvent être saisis soit au moyen de la notation indiquée ci-dessus, soit directement avec les symboles `<` et `>`.

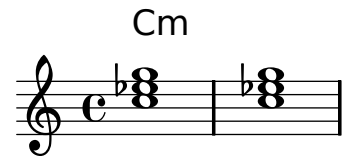
```
harmonies = {
  \chordmode {a1 b c} <d' f' a'> <e' g' b'>
}
<<
  \new ChordNames \harmonies
  \new Staff \harmonies
>>
```

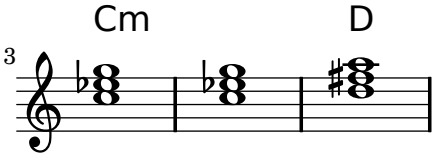


Vous pouvez faire ressortir les chiffrages d'accords en assignant la valeur vrai à la propriété `chordChanges` dans le contexte Section "ChordNames" dans *Référence des propriétés internes*. De cette façon, les chiffrages ne sont imprimés qu'aux changements d'accord ou en début de ligne.

```
harmonies = \chordmode {
  c1:m c:m \break c:m c:m d
}
<<
```

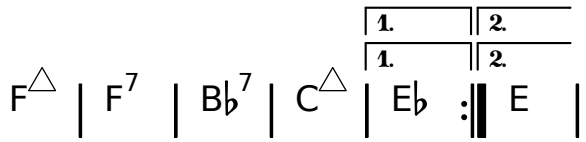
```
  \new ChordNames {
    \set chordChanges = ##t
    \harmonies }
  \new Staff \transpose c c' \harmonies
>>
```





Les exemples précédents montrent des chiffrages placés au-dessus de la portée, mais ce n'est pas obligatoire : les accords peuvent également être imprimés séparément – auquel cas vous aurez peut-être besoin d'ajouter le Section "Volta_engraver" dans *Référence des propriétés internes* et le Section "Bar_engraver" dans *Référence des propriétés internes* afin que les éventuelles barres de reprises s'affichent correctement.

```
\new ChordNames \with {
  \override BarLine #'bar-size = #4
  \consists Bar_engraver
  \consists "Volta_engraver"
}
\chordmode { \repeat volta 2 {
  f1:maj7 f:7 bes:7
  c:maj7
} \alternative {
  es e
}
}
```



Le modèle par défaut des chiffrages d'accord est celui de Klaus Ignatzek pour le jazz (cf. Annexe A [Bibliographie], page 195). Il s'agit d'une notation anglo-saxonne ; cependant vous pouvez indiquer vos chiffrages en notation française au moyen de la commande \frenchChords (voir plus bas).

Il est possible de créer votre propre modèle de chiffrages en réglant les propriétés suivantes :

`chordNameExceptions`

C'est la liste des accords mis en forme de manière particulière.

Cette liste d'exceptions s'indique de la façon suivante. On commence par créer une expression musicale telle que

```
chExceptionMusic = { <c f g bes>1 \markup { \super "7" "wahh" } }
```

puis on la transforme en liste d'exceptions au moyen du code

```
(sequential-music-to-chord-exceptions chExceptionMusic #t)
```

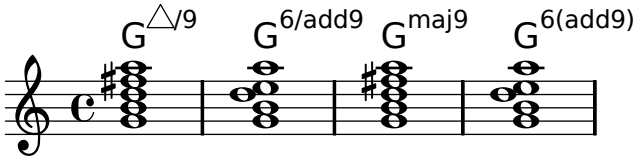
Pour qu'elles soient effectives, on ajoute ces exceptions aux exceptions par défaut définies dans '`ly/chord-modifier-init.ly`' :

```
(append
 (sequential-music-to-chord-exceptions chExceptionMusic #t)
 ignatzekExceptions)
```

Chord name exceptions

The property `chordNameExceptions` can be used to store a list of special notations for specific chords.

```
% modify maj9 and 6(add9)
% Exception music is chords with markups
chExceptionMusic = {
  <c e g b d'>1-\markup { \super "maj9" }
  <c e g a d'>1-\markup { \super "6(add9)" }
}

% Convert music to list and prepend to existing exceptions.
chExceptions = #( append
  ( sequential-music-to-chord-exceptions chExceptionMusic #t)
  ignatzekExceptions)

theMusic = \chordmode {
  g1:maj9 g1:6.9
  \set chordNameExceptions = #chExceptions
  g1:maj9 g1:6.9
}

\layout {
  ragged-right = ##t
}

<< \context ChordNames \theMusic
   \context Voice \theMusic
>>
```



`majorSevenSymbol`

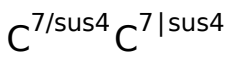
Cette propriété définit l'objet employé pour indiquer une septième majeure. Les options prédéfinies sont `whiteTriangleMarkup` (triangle blanc) et `blackTriangleMarkup` (triangle noir).

`chordNameSeparator`

Les différents termes d'un chiffrage jazz (par exemple les notes de l'accord et la basse) sont habituellement séparés par une barre oblique. La propriété `chordNameSeparator` permet d'indiquer un autre séparateur, par exemple

```
\new ChordNames \chordmode {
  c:7sus4
```

```
  \set chordNameSeparator
    = \markup { \typewriter "|" }
  c:7sus4
}
```

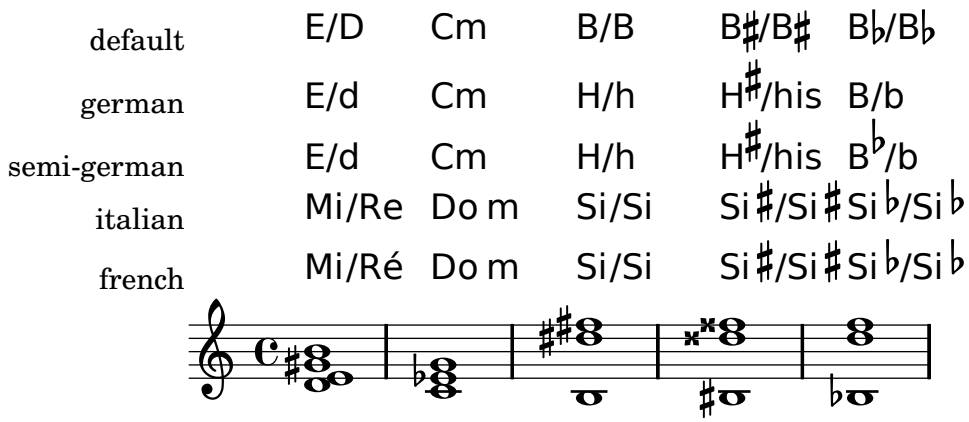


`chordRootNamer`
: Dans les chiffrages d'accord jazz, la note fondamentale de chaque accord est exprimée par une lettre capitale, parfois suivie d'une altération, correspondant à la notation anglo-saxonne de la musique. Cette propriété a pour valeur la fonction qui transforme la hauteur de la note foncdamentale en nom de note ; c'est donc en assignant une nouvelle fonction à cette propriété que l'on peut produire des noms de note spéciaux, adaptés par exemple aux systèmes de notation d'autres pays.

`chordNoteNamer`
: Lorsqu'un chiffrage mentionne une note ajoutée (par exemple la basse), les règles utilisées sont par défaut celles définies par la propriété `chordRootNamer` ci-dessus. Cependant, la propriété `chordNoteNamer` permet de régler cet élément indépendamment, par exemple pour imprimer la basse en caractères minuscules.

`chordPrefixSpacer`
: Le petit 'm' qui indique un accord mineur est, dans la notation anglo-saxonne, attaché à la lettre de l'accord. Cependant il est possible d'ajouter un espace en assignant la valeur vrai à la propriété `chordPrefixSpacer`. Cet espace sera omis si une altération est présente.

Les propriétés ci-dessus font l'objet de commandes prédéfinies adaptées aux notations de différents pays : `\germanChords` et `\semiGermanChords` pour la notation allemande, `\italianChords` pour la notation italienne et enfin `\frenchChords` pour la notation française.



LilyPond intègre également deux autres modèles de chiffrages : une notation Jazz alternative, et un modèle systématique appelé système Banter. Pour la mise en œuvre de ces modèles, voir Section B.1 [Table des noms d'accord], page 196.

Commandes prédéfinies

`\germanChords`, `\semiGermanChords`, `\italianChords`, `\frenchChords`.

Voir aussi

Exemples : Section “Chords” dans *Exemples de code*.

Fichiers internes : ‘`scm/chords-ignatzek.scm`’, et ‘`scm/chord-entry.scm`’.

Problèmes connus et avertissements

Les chiffrages d’accords ne sont déterminés que par la succession des hauteurs de notes. En d’autres termes, les accords inversés ne sont pas reconnus, ni les notes ajoutées à la basse. C’est pourquoi les accords saisis au moyen de la syntaxe `<.. >` peuvent produire des chiffrages étranges.

Personnalisation des noms d’accords

2.7.3 Basse chiffrée

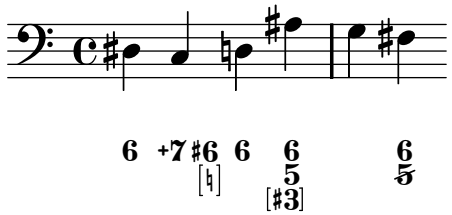
Introduction à la basse chiffrée

Les parties de basse continue étaient très répandues dans la musique baroque et jusqu’à la fin du XVIIIe siècle. Comme son nom l’indique, le *continuo* constitue une partie à lui seul, qui se déroule tout au long de l’œuvre pour en donner la structure harmonique.

Les musiciens du *continuo* jouent des claviers (clavecin, orgue) ou de tout autre instrument pouvant réaliser des accords. Leur partie est constituée d’une portée de basse aux notes agrémentées de combinaisons de chiffres et signes indiquant le développement des accords à jouer, ainsi que leur éventuel renversement. Cette notation était avant tout un guide, invitant le musicien à improviser de lui-même l’accompagnement.

LilyPond gère la basse chiffrée.

```
<<
  \new Voice { \clef bass dis4 c d ais g fis}
  \new FiguredBass \figuremode {
    < 6 >4 < 7\+ >8 < 6+ [_!] >
    < 6 >4 <6 5 [3+] >
    < _ >4 < 6 5/>4
  }
>>
```



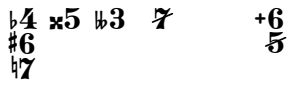
La gestion de la basse chiffrée se décompose en deux parties. Dans un premier temps, le mode `\figuremode` permet de saisir les accords sous forme chiffrée. Le contexte Section “FiguredBass” dans *Référence des propriétés internes* s’occupera ensuite de gérer les objets Section “BassFigure” dans *Référence des propriétés internes*.

En mode de saisie, un chiffrage est délimité par `<` et `>`. La durée est indiquée après le `>` :

```
<4 6>
```

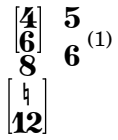
Les altérations s'obtiennent en ajoutant aux chiffres les caractères `-`, `!` ou `+`. Un signe plus s'obtient grâce à `\+`, et une quinte ou septième diminuée par `5/` ou `7/` respectivement.

```
<4- 6+ 7!> <5++> <3--> <7/> r <6\+ 5/>
```



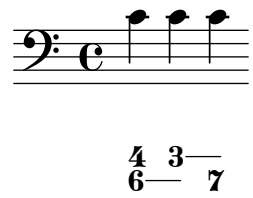
Le caractère `_` insère un espace, et l'on peut imprimer des crochets avec `[` et `]`. Vous pouvez aussi ajouter des chaînes de caractères ou des étiquettes — cf. Section B.6 [Text markup commands], page 196.

```
<[4 6] 8 [_! 12] > <5 \markup { \number 6 \super (1) } >
```



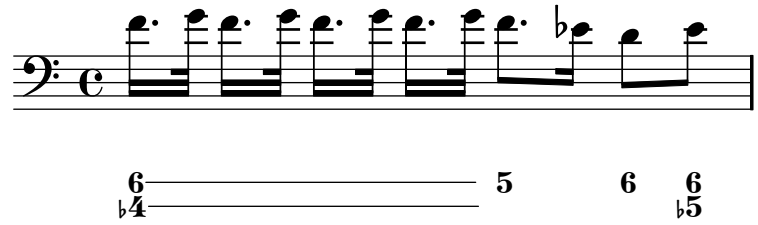
Lorsque des chiffrages se répètent, vous pouvez utiliser des lignes de prolongation.

```
<<
  \new Staff {
    \clef bass
    c4 c c
  }
  \figures {
    \set useBassFigureExtenders = ##t
    <4 6> <3 6> <3 7>
  }
>>
```



En pareil cas, la ligne de prolongation masquera toujours le chiffre qu'elle rappelle dans le chiffrage suivant.

Le contexte `FiguredBass` ne tient aucun compte de la ligne de basse. Par conséquent, il vous faudra peut être insérer des chiffrages supplémentaires pour imprimer les prolongations, ou utiliser des `\!` pour les éviter, comme dans l'exemple suivant :



Lorsque vous utilisez des lignes de prolongation, les chiffres communs seront verticalement alignés. Pour l'éviter, insérez un silence avec `r` afin de réinitialiser l'alignement. Par exemple, saisissez

```
<4 6>8 r8
```

au lieu de

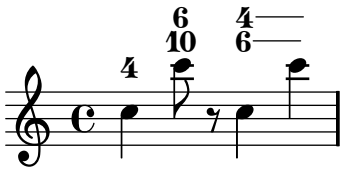
```
<4 6>4
```

On peut choisir d'impriemr les altérations et signes plus aussi bien avant qu'après les chiffres, en réglant les propriétés `figuredBassAlterationDirection` et `figuredBassPlusDirection`.

Bien que la gestion de la basse chiffrée ressemble beaucoup à celle des accords, elle est beaucoup plus simpliste. Le mode `\figuremode` ne fait que stocker des chiffres que le contexte Section "FiguredBass" dans *Référence des propriétés internes* se chargera d'imprimer tels quels. En aucune manière ils ne sont transformés en son, et ils ne sont pas rendus dans un fichier MIDI.

En interne, ce code produit des étiquettes de texte que vous pouvez formater à votre convenance grâce aux propriétés des étiquettes. Par exemple, l'espacement vertical des chiffrages est déterminé par la propriété `baseline-skip`.

On peut également ajouter une basse chiffrée directement à un contexte `Staff`. L'alignement vertical est alors automatiquement ajusté.



Propriétés couramment modifiées

Par défaut, les chiffres sont imprimés au-dessus de la portée. Pour les imprimer dessous, ajoutez

```
\override Staff.BassFigureAlignmentPositioning #'direction = #DOWN
```

Problèmes connus et avertissements

Si vous positionnez la basse chiffrée au dessus de la portée en ayant recours aux lignes d'extension et `implicitBassFigures`, les lignes peuvent se mélanger. Préserver l'ordre des prolongateurs peut s'avérer impossible lorsque plusieurs chiffrages qui se chevauchent en possèdent. Ce problème peut être contourné en jouant sur l'empilement, avec la propriété `stacking-dir` de l'objet `BassFigureAlignment`.

Voir aussi

Référence du programme : les objets Section "BassFigure" dans *Référence des propriétés internes*, Section "BassFigureAlignment" dans *Référence des propriétés internes*, Section "BassFigureLine" dans *Référence des propriétés internes*, Section "BassFigureBracket" dans *Référence des propriétés internes* et Section "BassFigureContinuation" dans *Référence des propriétés internes*, ainsi que le contexte Section "FiguredBass" dans *Référence des propriétés internes*.

Saisie de la basse chiffrée

Gravure de la basse chiffrée

2.8 Notations anciennes

2.8.1 Introduction aux notations anciennes

La gestion par LilyPond des formes de notation ancienne inclut des fonctionnalités spécifiques à la notation mensurale et au chant grégorien. La basse chiffrée est également partiellement prise en charge.

De nombreux objets graphiques — « grobs » dans le jargon de LilyPond — disposent d'une propriété `style`, comme nous le verrons dans

- Section 2.8.2.1 [Têtes de note anciennes], page 147,
- Section 2.8.2.2 [Altérations anciennes], page 148,
- Section 2.8.2.3 [Silences anciens], page 149,
- Section 2.8.2.4 [Clefs anciennes], page 150,
- Section 2.8.2.5 [Crochets anciens], page 152,
- Section 2.8.2.6 [Métriques anciennes], page 153.

Manipuler cette propriété permet d'adapter l'aspect typographique des grobs à une forme de notation particulière, ce qui évite la création de nouveaux concepts de notation.

En plus des signes d'articulation standards décrits à la section [Articulations et ornements], page 50, la notation ancienne dispose de signes particuliers.

- Section 2.8.3.1 [Articulations anciennes], page 154

D'autres aspects de la notation ancienne ne peuvent pas être gérés aussi simplement qu'en jouant sur les propriétés d'un style appliqué à un objet graphique ou en lui ajoutant des articulations. Certains concepts sont spécifiques à la notation ancienne.

- Section 2.8.3.2 [Guidons], page 155,
- Section 2.8.3.3 [Divisions], page 155,
- Section 2.8.3.4 [Ligatures], page 156.

Si tout cela vous dépasse et que vous désirez plonger dans le vif du sujet sans trop vous préoccuper d'ajuster des contextes, consultez les pages dédiées aux contextes prédéfinis. Ils vous permettront d'adapter vos contextes de voix et de portée, et vous n'aurez plus qu'à saisir les notes.

- Section 2.8.4.1 [Contextes du chant grégorien], page 164,
- Section 2.8.4.2 [Le contexte mensural], page 164.

LilyPond gère partiellement la représentation de basses chiffrées, typiques de l'époque baroque, mais également employées de nos jours en harmonie et en analyse.

- Section 2.7.3 [Basse chiffrée], page 144

Voici les points que nous allons aborder :

2.8.1.1 Formes de notation ancienne prises en charge

2.8.2 Signes de note alternatifs

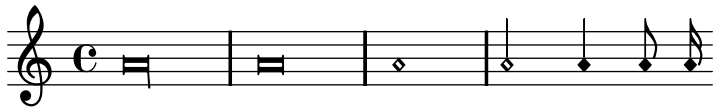
2.8.2.1 Têtes de note anciennes

Pour de la musique ancienne, vous disposez de plusieurs styles de tête de note, en plus du style par défaut `default`. Vous pouvez affecter à la propriété `style` de l'objet Section "NoteHead" dans *Référence des propriétés internes* les valeurs `baroque`, `neomensural`, `mensural` ou `petrucci`. En style `baroque`, la seule différence par rapport au style `default` concerne la `\breve` qui sera carrée et non pas ovoïde. Le style `neomensural` ajoute au `baroque` le fait que les notes de durée

inférieure ou égale à une ronde sont en forme de losange, et les hampes centrées sur la tête. Ce style est particulièrement adapté à la transcription de la musique mesurée dans les incipits. Le style `mensural` permet de reproduire les têtes de note telles qu'imprimées au XVIe siècle. Enfin, le style `petrucci` imite des partitions historiques, bien qu'il utilise de plus grosses têtes de note.

L'exemple suivant illustre le style `neomensural`.

```
\set Score.skipBars = ##t
\override NoteHead #'style = #'neomensural
a'\longa a'\breve a'1 a'2 a'4 a'8 a'16
```



Si vous écrivez en notation grégorienne, le Section "Vaticana_ligature_engraver" dans *Référence des propriétés internes* se chargera de sélectionner les têtes de note appropriées ; il est donc inutile de spécifier le style à utiliser. Vous pouvez cependant spécifier par exemple le style `vaticana_punctum` pour obtenir des neumes punctums. De même, c'est le Section "Mensural_ligature_engraver" dans *Référence des propriétés internes* qui se chargera des ligatures mensurales. Consultez la section Section 2.8.3.4 [Ligatures], page 156 pour savoir comment fonctionnent les graveurs de ligature.

Voir aussi

Pour un aperçu de toutes les possibilités, consultez Section B.5 [Styles de tête de note], page 196.

2.8.2.2 Altérations anciennes

Pour utiliser les formes anciennes d'altération, utilisez la propriété `glyph-name-alist` des objets graphiques Section "Accidental" dans *Référence des propriétés internes* et Section "KeySignature" dans *Référence des propriétés internes*.

vaticana medicaea hufnagel mensural

Vous noterez que chacun de ces styles ne comporte pas toutes les altérations. LilyPond changera de style s'il y a besoin d'une altération indisponible dans le style utilisé.

À l'instar des altérations accidentelles, le style d'armure est géré par la propriété `glyph-name-alist` de l'objet Section "KeySignature" dans *Référence des propriétés internes*.

Voir aussi

Dans ce manuel : Section 1.1 [Hauteurs], page 1, [Altérations], page 3, et [Altérations accidentelles automatiques], page 11, pour les principes généraux d'utilisation des altérations ; [Armure], page 9 pour les armures.

Référence du programme : Section "KeySignature" dans *Référence des propriétés internes*.

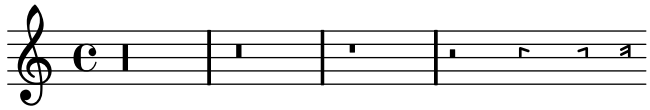
Exemples : Section "Ancient notation" dans *Exemples de code*.

2.8.2.3 Silences anciens

La propriété `style` de l'objet Section "Rest" dans *Référence des propriétés internes* permet d'obtenir des silences de type ancien. Vous disposez des styles `classical`, `neomensural` et `mensural`. Le style `classical` ne se distingue du style `default` que par le soupir (demi-soupir en miroir). Le style `neomensural` convient tout à fait à l'incipit lors de la transcription de musique mensurale. Le style `mensural`, enfin, imite la gravure des silences dans certaines éditions du XVIe siècle.

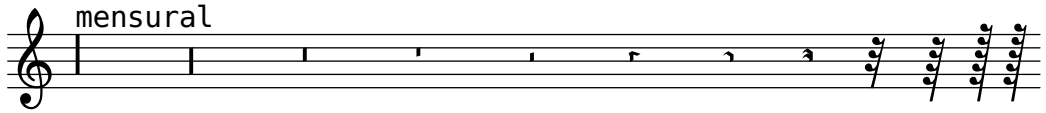
L'exemple suivant illustre le style `neomensural`.

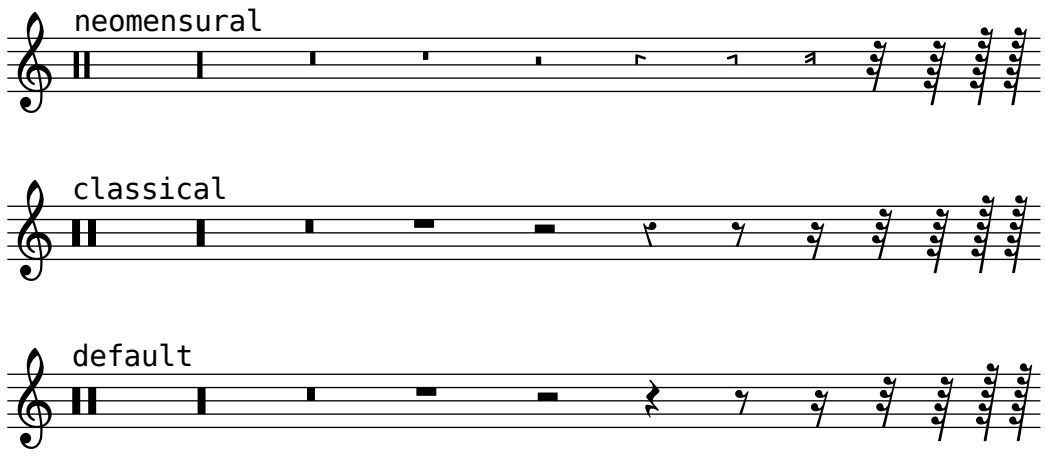
```
\set Score.skipBars = ##t
\override Rest #'style = #'neomensural
r\longa r\breve r1 r2 r4 r8 r16
```



Les styles `mensural` et `neomensural` ne disposent pas des 8e et 16e de soupir ; LilyPond utilise dans de tels cas le style par défaut. Voici une liste des styles de silences disponibles.

```
\layout {
  indent = 0.0
  \context {
    \Staff
    \remove "Time_signature_engraver"
  }
}

\relative c {
  \set Score.timing = ##f
  \override Staff.Rest  #'style = #'mensural
  r\maxima^\markup \typewriter { mensural }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 r128
  \bar ""

  \override Staff.Rest  #'style = #'neomensural
  r\maxima^\markup \typewriter { neomensural }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 r128
  \bar ""

  \override Staff.Rest  #'style = #'classical
  r\maxima^\markup \typewriter { classical }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 r128
  \bar ""

  \override Staff.Rest  #'style = #'default
  r\maxima^\markup \typewriter { default }
  r\longa r\breve r1 r2 r4 r8 r16 r32 r64 r128 r128
}
```





Les silences sont absents de la notation grégorienne ; par contre, cette notation utilise des Section 2.8.3.3 [Divisions], page 155.

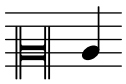
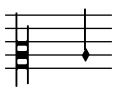
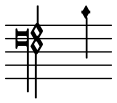
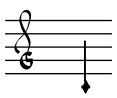
Voir aussi

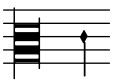
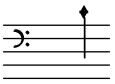
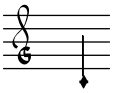
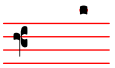
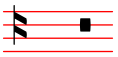
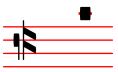
Dans ce manuel : les principes généraux sur l'utilisation des silences sont exposés dans Section 1.2.2.1 [Silences], page 25.

2.8.2.4 Clefs anciennes

Avec LilyPond, de nombreuses clés sont disponibles, dont certaines sont dédiées à la musique ancienne.

Le tableau suivant présente les différentes clés anciennes que vous pouvez sélectionner avec la commande `\clef`. Certaines de ces clés utilisent le même glyphe, attaché à l'une ou l'autre des lignes de la portée. Le chiffre porté en suffixe permet alors de les différencier. Vous pouvez néanmoins forcer le positionnement du glyphe sur une ligne, comme expliqué à la section [Clefs], page 7. Dans la colonne exemple, la note suivant la clé montre le do médium.

Description	**Clé disponible**	**Exemple**
Clé d'ut, style mensural moderne	`neomensural-c1`, `neomensural-c2`, `neomensural-c3`, `neomensural-c4`	
Clé d'ut, style mensural Petrucci, positionnable sur différentes lignes (clé d'ut seconde pour l'exemple)	`petrucci-c1`, `petrucci-c2`, `petrucci-c3`, `petrucci-c4`, `petrucci-c5`	
Clé de fa, style mensural Petrucci	`petrucci-f`	
Clé de sol, style mensural Petrucci	`petrucci-g`	

Clé d'ut, style mensural historique	`mensural-c1`, `mensural-c2`, `mensural-c3`, `mensural-c4`	
Clé de fa, style mensural historique	`mensural-f`	
Clé de sol, style mensural historique	`mensural-g`	
Clé d'ut, style Editio Vaticana	`vaticana-do1`, `vaticana-do2`, `vaticana-do3`	
Clé de fa, style Editio Vaticana	`vaticana-fa1`, `vaticana-fa2`	
Clé d'ut, style Editio Medicaea	`medicaea-do1`, `medicaea-do2`, `medicaea-do3`	
Clé de fa, style Editio Medicaea	`medicaea-fa1`, `medicaea-fa2`	
Clé d'ut, style historique Hufnagel	`hufnagel-do1`, `hufnagel-do2`, `hufnagel-do3`	
Clé de fa, style historique Hufnagel	`hufnagel-fa1`, `hufnagel-fa2`	

Clé combinée ut/fa, style historique `hufnagel-do-fa`
Hufnagel

Moderne signifie « gravé comme dans les transcriptions contemporaines de musique mesurée. »

Petrucci signifie « inspiré des éditions réalisées par le maître graveur Petrucci (1466-1539). »

Historique signifie « gravé comme dans les éditions historiques, manuscrites ou non, autres que celles de Petrucci. »

Editio XXX signifie « gravé comme dans les ouvrages estampillés Editio XXX. »

Les clés d'ut de Petrucci avaient une hampe gauche différente selon leur ligne de rattachement.

Voir aussi

Dans ce manuel : voir [Clefs], page 7.

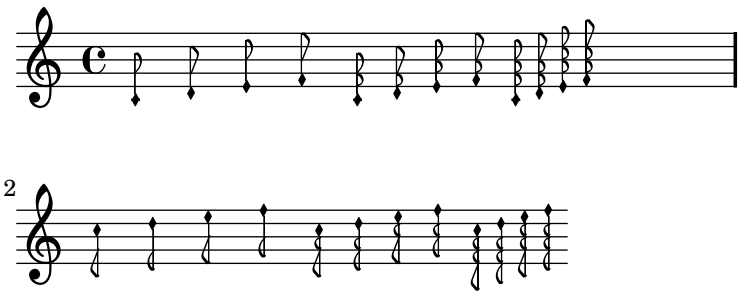
Problèmes connus et avertissements

La clé de sol mensurale est calquée sur celle de Petrucci.

2.8.2.5 Crochets anciens

Le réglage de la propriété `flag-style` de l'objet hampe (Section "Stem" dans *Référence des propriétés internes*) donne accès aux crochets de style ancien. Les seuls styles actuellement pris en charge sont `default` et `mensural`.

```
\override Stem #'flag-style = #'mensural
\override Stem #'thickness = #1.0
\override NoteHead #'style = #'mensural
\autoBeamOff
c'8 d'8 e'8 f'8 c'16 d'16 e'16 f'16 c'32 d'32 e'32 f'32 s8
c''8 d''8 e''8 f''8 c''16 d''16 e''16 f''16 c''32 d''32 e''32 f''32
```



Notez que pour chaque crochet mensural, l'extrémité la plus proche de la tête de note sera toujours attachée à une ligne de la portée.

Il n'existe pas de crochet spécifique au style néo-mensural. Nous vous conseillons donc, lorsque vous réalisez l'incipit d'une transcription, d'utiliser le style par défaut. Les crochets n'existent pas en notation grégorienne.

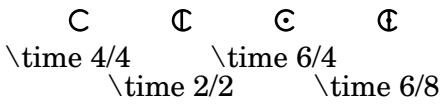
Problèmes connus et avertissements

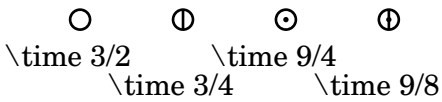
Les crochets anciens s'attachent aux hampes avec un léger décalage, suite à des modifications intervenues au début de la série 2.3.

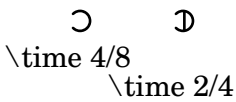
L'alignement vertical des crochets par rapport aux lignes de la portée sous-entend que les hampes se terminent toujours soit sur une ligne, soit à l'exact milieu d'un interligne. Ceci n'est pas toujours réalisable, surtout si vous faites appel à des fonctionnalités avancées de présentation de la notation classique, qui, par définition, ne sont pas prévues pour être appliquées à la notation mensurale.

2.8.2.6 Métriques anciennes

Les chiffrages de métrique mensurale sont partiellement pris en charge. Les glyphes ne font que représenter des métriques particulières. En d'autres termes, pour obtenir le glyphe correspondant à une métrique mensurale particulière à l'aide de la commande `\time n/m`, vous devez choisir la paire `(n,m)` parmi les valeurs suivantes :

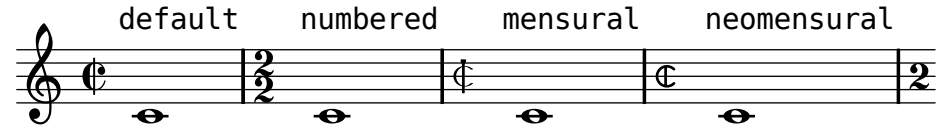


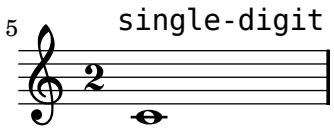




La propriété `style` de l'objet Section "TimeSignature" dans *Référence des propriétés internes* permet d'accéder aux indicateurs de métrique anciens. Les styles `neomensural` et `mensural` sont disponibles. Vous avez vu ci-dessus le style `neomensural`, particulièrement utilisé pour l'incipit des transcriptions. Le style `mensural` imite l'aspect de certaines éditions du XVIe siècle.

Voici les différences entre les styles :





Voir aussi

Dans ce manuel : Section 1.2.3.1 [Métrique], page 28 expose les principes généraux sur l'utilisation des indications de métrique.

Problèmes connus et avertissements

Les équivalences de durées de note ne sont pas modifiées par un changement de métrique. Par exemple, l'équivalence 1 brève pour 3 semi-brèves (tempus perfectum) doit s'effectuer à la main en entrant :

```
breveTP = #(ly:make-duration -1 0 3 2)
...
{ c\breveTP f1 }
```

Ce qui définira `breveTP` à 3/2 fois 2 = 3 fois une ronde.

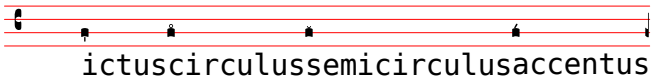
Le symbole `old6/8alt` — symbole alternatif pour la métrique 6/8 — ne peut être utilisé grâce à une commande `\time`. Utilisez plutôt un `\markup`.

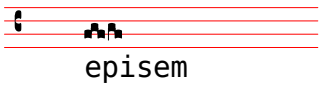
2.8.3 Signes de note supplémentaires

2.8.3.1 Articulations anciennes

En plus des signes d'articulation standards décrits à la section [Articulations et ornements], page 50, LilyPond fournit des articulations pour la musique ancienne. Elles sont spécifiquement destinées au style Editio Vaticana.

```
\include "gregorian-init.ly"
\score {
  \new VaticanaVoice {
    \override TextScript #'font-family = #'typewriter
    \override TextScript #'font-shape = #'upright
    \override Script #'padding = #-0.1
    a\ictus_"ictus" \break
    a\circulus_"circulus" \break
    a\semicirculus_"semicirculus" \break
    a\accentus_"accentus" \break
    \[ a_"episem" \episemInitium \pes b \flexa a b \episemFinis \flexa a \]
  }
}
```





Problèmes connus et avertissements

Certaines articulations sont verticalement trop proches de leurs têtes de note.

Le trait d'un episem n'est bien souvent pas apparent et, lorsqu'il l'est, son extension à droite est trop longue.

2.8.3.2 Guidons

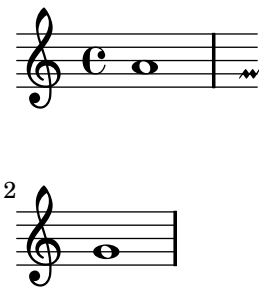
Un guidon — *custos*, pluriel *custodes* en latin — est un symbole qui apparaît à la fin d'une portée. Il montre la hauteur de la ou des premières notes de la portée suivante, donnant une indication judicieuse à l'exécutant.

Les guidons étaient couramment utilisés jusqu'au XVIIe siècle. De nos jours, on les retrouve uniquement dans quelques formes particulières de notation telles que les éditions contemporaines de chant grégorien comme les *editio vaticana*. Différents glyphes existent selon le style de notation.

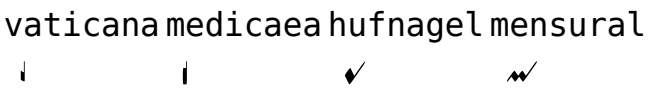
L'impression de guidons s'obtient en affectant, dans un bloc `\layout`, le Section "Custos_engraver" dans *Référence des propriétés internes* au contexte `Staff`, comme le montre l'exemple suivant.

```
\layout {
  \context {
    \Staff
    \consists Custos_engraver
    Custos \override #'style = #'mensural
  }
}
```

Le résultat ressemblera à



Le glyphe du guidon est déterminé par la propriété `style`. Les styles disponibles sont `vaticana`, `medicaea`, `hufnagel` et `mensural`. En voici un aperçu :



Voir aussi

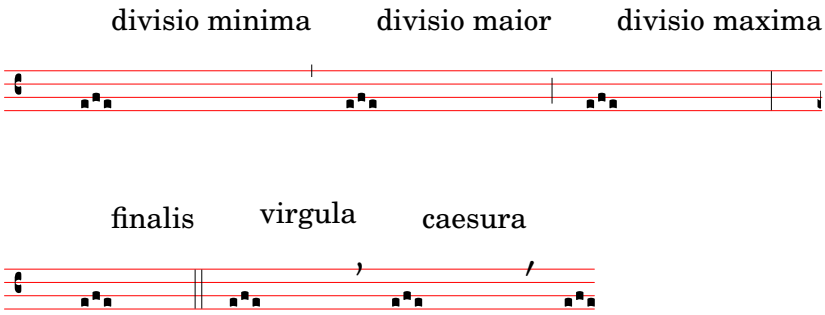
Référence du programme : Section "Custos" dans *Référence des propriétés internes*.

Exemples : Section "Ancient notation" dans *Exemples de code*.

2.8.3.3 Divisions

Une division — *divisio*, pluriel *divisiones* en latin — est un symbole ajouté à la portée et utilisé en chant grégorien pour séparer les phrases ou parties. *Divisio minima*, *divisio maior* et *divisio maxima* peuvent respectivement s'interpréter comme une pauses courte, moyenne ou longue, à l'image des marques de respiration — cf. [Signes de respiration], page 56. Le signe *finalis* n'est pas uniquement une marque de fin de chant ; il sert aussi à indiquer la fin de chaque partie dans une structure verset/répons.

Les divisions sont disponibles après inclusion du fichier ‘`gregorian-init.ly`’. Ce fichier définit les commandes `\divisioMinima`, `\divisioMaior`, `\divisioMaxima` et `\finalis`. Certaines éditions utilisent *virgula* ou *caesura* en lieu et place de *divisio minima* ; c’est pourquoi ‘`gregorian-init.ly`’ définit aussi `\virgula` et `\caesura`.



Commandes prédéfinies

`\virgula`, `\caesura`, `\divisioMinima`, `\divisioMaior`, `\divisioMaxima`, `\finalis`.

Voir aussi

Dans ce manuel : [Signes de respiration], page 56.

Référence du programme : Section “BreathingSign” dans *Référence des propriétés internes*.

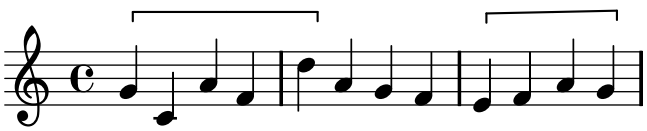
Exemples : Section “Winds” dans *Exemples de code*.

2.8.3.4 Ligatures

Une ligature est un symbole graphique qui représente un groupe d’au moins deux notes. Les ligatures ont commencé à apparaître dans les manuscrits de chant grégorien, pour indiquer des suites ascendantes ou descendantes de notes.

Les ligatures s’indiquent par une inclusion entre `\[` et `\]`. Certains styles de ligature peuvent demander un complément de syntaxe spécifique. Par défaut, le graveur Section “LigatureBracket” dans *Référence des propriétés internes* place un simple crochet au dessus de la ligature :

```
\transpose c c' {
  \[ g c a f d' \]
  a g f
  \[ e f a g \]
}
```



Selon le style de ligature désiré, il faut ajouter au contexte Section “Voice” dans *Référence des propriétés internes* le graveur de ligature approprié, comme nous le verrons plus loin. Seules sont disponibles les ligatures mensurales blanches, avec quelques limitations.

Problèmes connus et avertissements

La gestion de l'espacement spécifique aux ligatures n'est à ce jour pas implémentée. En conséquence, les ligatures sont trop espacées les unes des autres et les sauts de ligne mal ajustés. De plus, les paroles ne s'alignent pas de manière satisfaisante en présence de ligatures.

Les altérations ne pouvant être imprimées à l'intérieur d'une ligature, il faut les rassembler et les imprimer juste avant.

La syntaxe utilisée correspond à l'ancienne convention de préfixage `\[ expr. musicale\]`. Pour des raisons d'uniformité, nous opterons probablement pour le style en suffixe (postfix) `note\[ ... note\]`. En attendant, vous pouvez inclure le fichier '`gregorian-init.ly`', qui fournit une fonction Scheme

`\ligature` *`expr. musicale`*

qui produit le même résultat, et dont la pérennité est assurée.

* Ligatures mensurales:: * Neumes ligaturés grégoriens::

2.8.3.5 Ligatures mensurales

Les ligatures mensurales blanches sont prises en charge, avec des limitations.

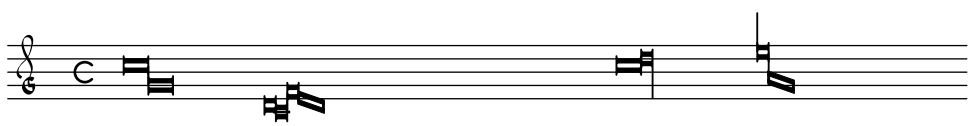
La gravure des ligatures mensurales blanches s'obtient après avoir ajouté le Section "Mensural_ligature_engraver" dans *Référence des propriétés internes* et enlevé le Section "Ligature_bracket_engraver" dans *Référence des propriétés internes* dans le contexte Section "Voice" dans *Référence des propriétés internes*, comme ici :

```
\layout {
  \context {
    \Voice
    \remove Ligature_bracket_engraver
    \consists Mensural_ligature_engraver
  }
}
```

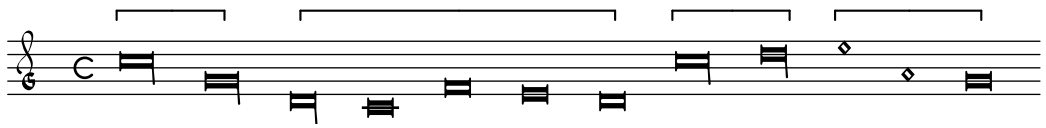
Lorsque le code ci-dessus est employé, l'aspect d'une ligature mensurale blanche est déterminé à partir des hauteurs et durées des notes qui la composent. Bien que cela demande un temps d'adaptation au nouvel utilisateur, cette méthode offre l'avantage que toute l'information musicale incluse dans la ligature est connue en interne. Ceci est non seulement important pour le rendu MIDI, mais aussi pour des questions de transcription automatisée d'une ligature.

Par exemple,

```
\set Score.timing = ##f
\set Score.defaultBarType = "empty"
\override NoteHead #'style = #'neomensural
\override Staff.TimeSignature #'style = #'neomensural
\clef "petrucci-g"
\[ c'\maxima g \]
\[ d\longa c\breve f e d \]
\[ c'\maxima d'\longa \]
\[ e'1 a g\breve \]
```



Si on ne remplace pas le Section "Ligature_bracket_engraver" dans *Référence des propriétés internes* par le Section "Mensural_ligature_engraver" dans *Référence des propriétés internes*, on obtient



Problèmes connus et avertissements

L'espacement horizontal n'est pas des meilleurs.

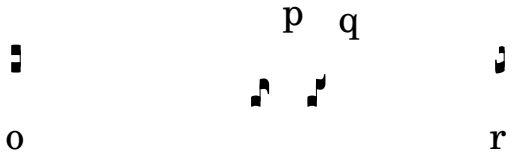
2.8.3.6 Neumes ligaturés grégoriens

Les neumes grégoriens conformément au style des Editio Vaticana sont pris en charge de façon assez limitée. Les ligatures élémentaires sont déjà disponibles, mais beaucoup de règles typographiques ne sont pas encore implémentées, notamment l'espacement horizontal des enchaînements de ligatures, l'alignement des paroles ou une gestion convenable des altérations.

Le tableau ci-dessous inventorie les différents neumes contenus dans le second tome de l'Antiphonale Romanum (*Liber Hymnarius*) publié par l'abbaye de Solesmes en 1983.

Neuma aut Neumarum Elementa	**Figurae Rectae**	**Figurae Liquescentes Auctae**	**Figurae Liquescentes Deminutae**
`1. Punctum`	a b	c d e	f
`2. Virga`	g		
`3. Apostropha vel Stropha`	h	i	
`4. Oriscus`	j		
`5. Clivis vel Flexa`	k	l m	n

6. Podatus vel Pes

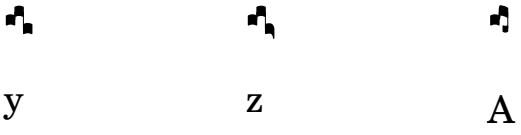


7. Pes Quassus

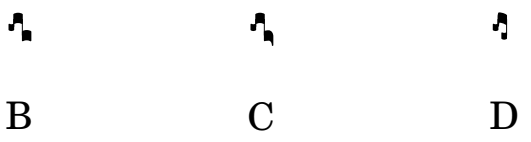
8. Quilisma Pes

9. Podatus Initio Debilis

10. Torculus



11. Torculus Initio Debilis



B C D

12. Porrectus

E F G

13. Climacus

H I J

14. Scandicus

K L M

15. Salicus

N O

`16. Trigonus`

P

Contrairement à la majorité des autres systèmes de notation neumatique, la manière de saisir les neumes n'a rien à voir avec leur apparence typographique ; elle se concentre plutôt sur le sens musical. Ainsi, `\[ a \pes b \flexa g \]` produit un *torculus* constitué de trois *punctums*, alors que `\[ a \flexa g \pes b \]` produit un *porrectus* avec une flexe incurvée et un seul *punctum*. Il n'existe pas de commande à proprement parler qui permette de spécifier la courbe d'une flexe ; c'est la source musicale qui va le déterminer. Le fondement d'une telle approche réside dans la distinction que nous faisons entre les aspects musicaux de la source et le style de notation que nous voulons obtenir. De ce fait, la même source pourra être utilisée pour imprimer dans un autre style de notation grégorienne.

Le tableau suivant présente les fragments de code qui ont permis de générer les neumes ligaturés du tableau précédent. Les lettres de la première colonne renvoient aux ligatures ci-dessus. La seconde colonne énumère le nom des ligatures, et la troisième le code ayant permis de les générer, se basant ici sur sol, la, si.

#	**Nom**	**Code source**
a	Punctum	`\[ b \]`
b	Punctum Inclinatum	`\[ \inclinatum b \]`
c	Punctum Auctum Ascendens	`\[ \auctum \ascendens b \]`
d	Punctum Auctum Descendens	`\[ \auctum \descendens b \]`
e	Punctum Inclinatum Auctum	`\[ \inclinatum \auctum b \]`
f	Punctum Inclinatum Parvum	`\[ \inclinatum \deminutum b \]`
g	Virga	`\[ \virga b \]`
h	Stropha	`\[ \stropha b \]`
i	Stropha Aucta	`\[ \stropha \auctum b \]`
j	Oriscus	`\[ \oriscus b \]`
k	Clivis vel Flexa	`\[ b \flexa g \]`
l	Clivis Aucta Descendens	`\[ b \flexa \auctum \descendens g \]`
m	Clivis Aucta Ascendens	`\[ b \flexa \auctum \ascendens g \]`

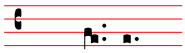
n	Cephalicus	`\[ b \flexa \deminutum g \]`
o	Podatus vel Pes	`\[ g \pes b \]`
p	Pes Auctus Descendens	`\[ g \pes \auctum \descendens b \]`
q	Pes Auctus Ascendens	`\[ g \pes \auctum \ascendens b \]`
r	Epiphonus	`\[ g \pes \deminutum b \]`
s	Pes Quassus	`\[ \oriscus g \pes \virga b \]`
t	Pes Quassus Auctus Descendens	`\[ \oriscus g \pes \auctum \descendens b \]`
u	Quilisma Pes	`\[ \quilisma g \pes b \]`
v	Quilisma Pes Auctus Descendens	`\[ \quilisma g \pes \auctum \descendens b \]`
w	Pes Initio Debilis	`\[ \deminutum g \pes b \]`
x	Pes Auctus Descendens Initio Debilis	`\[ \deminutum g \pes \auctum \descendens b \]`
y	Torculus	`\[ a \pes b \flexa g \]`
z	Torculus Auctus Descendens	`\[ a \pes b \flexa \auctum \descendens g \]`
A	Torculus Deminutus	`\[ a \pes b \flexa \deminutum g \]`
B	Torculus Initio Debilis	`\[ \deminutum a \pes b \flexa g \]`
C	Torculus Auctus Descendens Initio Debilis	`\[ \deminutum a \pes b \flexa \auctum \descendens g \]`
D	Torculus Deminutus Initio Debilis	`\[ \deminutum a \pes b \flexa \deminutum g \]`
E	Porrectus	`\[ a \flexa g \pes b \]`
F	Porrectus Auctus Descendens	`\[ a \flexa g \pes \auctum \descendens b \]`
G	Porrectus Deminutus	`\[ a \flexa g \pes \deminutum b \]`
H	Climacus	`\[ \virga b \inclinatum a \inclinatum g \]`
I	Climacus Auctus	`\[ \virga b \inclinatum a \inclinatum \auctum g \]`
J	Climacus Deminutus	`\[ \virga b \inclinatum a \inclinatum \deminutum g \]`
K	Scandicus	`\[ g \pes a \virga b \]`
L	Scandicus Auctus Descendens	`\[ g \pes a \pes \auctum \descendens b \]`

M	Scandicus Deminutus	`\[ g \pes a \pes \deminutum b \]`
N	Salicus	`\[ g \oriscus a \pes \virga b \]`
O	Salicus Auctus Descendens	`\[ g \oriscus a \pes \auctum \descendens b \]`
P	Trigonus	`\[ \stropha b \stropha b \stropha a \]`

Les ligatures que nous venons de voir, bien que rudimentaires, donnent un aperçu des possibilités de former des ligatures grégoriennes. En théorie, vous pouvez inclure entre les délimiteurs `\[` et `\]`, autant de sons que nécessaires à la ligature, ainsi que de préfixes tels que `\pes`, `\flexa`, `\virga`, `\inclinatum`, ... Bien sûr, les règles de construction présentées ci-dessus peuvent se combiner, ce qui permet la création d'une infinité de ligatures.

Les points d'*augmentum*, ou *morae*, s'obtiennent avec la fonction `\augmentum`. Notez que cette fonction `\augmentum` est implémentée en tant que fonction unaire plutôt que comme un préfixe de note. Par conséquent, `\augmentum \virga c` ne donnera rien de particulier. Il faut l'utiliser avec la syntaxe `\virga \augmentum c` ou `\augmentum {\virga c}`. Par ailleurs, l'expression `\augmentum {a g}` constitue une forme abrégée de `\augmentum a \augmentum g`.

```
\include "gregorian-init.ly"
\score {
  \new VaticanaVoice {
    \[ \augmentum a \flexa \augmentum g \]
    \augmentum g
  }
}
```



Commandes prédéfinies

LilyPond dispose des préfixes suivants :

`\virga`, `\stropha`, `\inclinatum`, `\auctum`, `\descendens`, `\ascendens`, `\oriscus`, `\quilisma`, `\deminutum`, `\cavum`, `\linea`.

Les préfixes de note peuvent s'agglutiner, modulo quelques restrictions. Par exemple, on peut appliquer un `\descendens` ou un `\ascendens` à une note, mais pas les deux simultanément à une même note.

Deux notes adjacentes peuvent être reliées grâce aux commandes `\pes` ou `\flexa` pour marquer une ligne mélodique respectivement ascendante ou descendante.

Utilisez la fonction musicale unaire `\augmentum` pour ajouter des points d'augmentum.

Problèmes connus et avertissements

Lorsqu'un `\augmentum` apparaît dans une ligature en fin de portée, son placement vertical peut être erroné. Pour y remédier, ajoutez un silence invisible, `s8` par exemple, comme dernière note de cette portée.

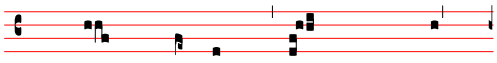
L'`\augmentum` devrait être implémenté en tant que préfixe plutôt qu'en tant que fonction unaire, afin qu'`\augmentum` puisse s'intégrer avec d'autres préfixes dans n'importe quel ordre.

2.8.4 Contextes prédéfinis

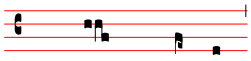
2.8.4.1 Contextes du chant grégorien

Les contextes `VaticanaVoiceContext` et `VaticanaStaffContext` permettent de graver le chant grégorien dans le style des éditions vaticanes. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant, comme ci-dessous :

```
\include "gregorian-init.ly"
\score {
  <<
    \new VaticanaVoice = "cantus" {
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \]
      f \divisioMinima
      \[ f\melisma \pes a c' c' \pes d'\melismaEnd \]
      c' \divisioMinima \break
      \[ c'\melisma c' \flexa a \]
      \[ a \flexa \deminutum g\melismaEnd \] f \divisioMinima
    }
    \new Lyrics \lyricsto "cantus" {
      San- ctus, San- ctus, San- ctus
    }
  >>
}
```



San- ctus, San- ctus,



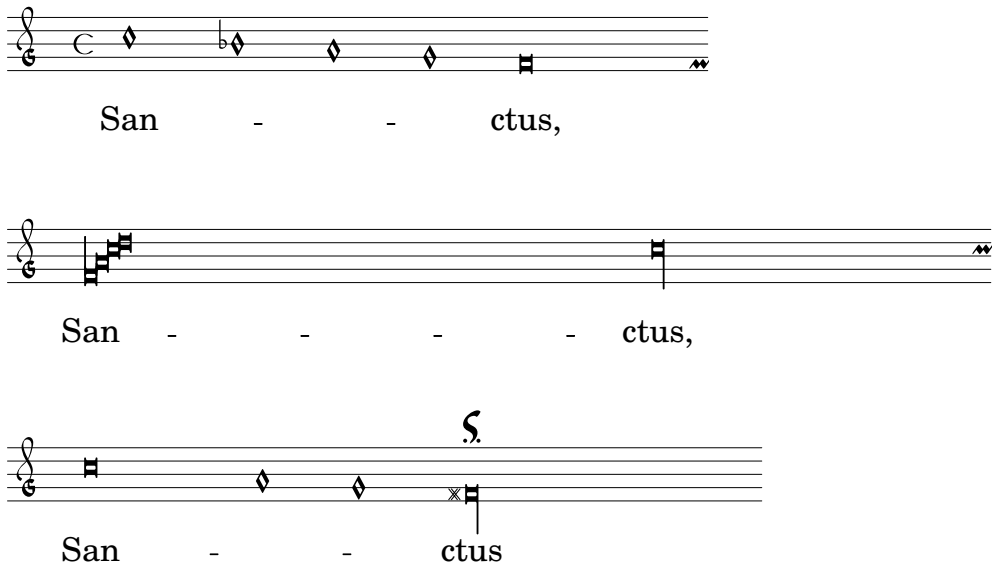
San- ctus

2.8.4.2 Le contexte mensural

Les contextes `MensuralVoiceContext` et `MensuralStaffContext` permettent de graver des chants dans le style mesuré. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant comme ci-après :

```
\score {
  <<
    \new MensuralVoice = "discantus" \transpose c c' {
      \override Score.BarNumber #'transparent = ##t {
        c'1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c'\breve d'\melismaEnd \]
        c'\longa
        c'\breve\melisma a1 g1\melismaEnd
        fis\longa^\signumcongruentiae
```

```
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```



2.8.5 Transcription de musique mensurale

2.8.5.1 Différentes éditions à partir d'une même source

2.8.5.2 Des incipits

2.8.5.3 Mise en forme de la musique mensurale

2.8.5.4 Transcription de chant grégorien

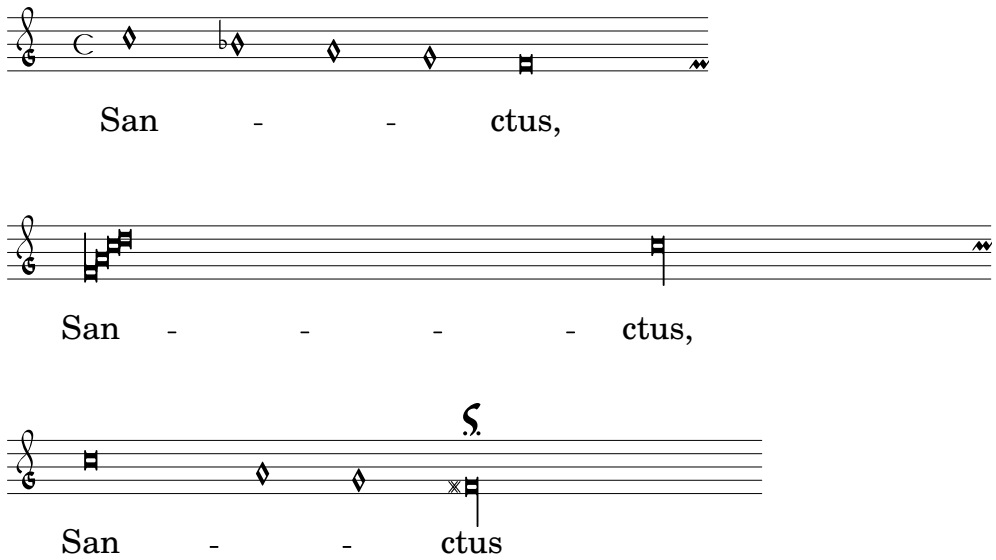
2.8.6 Notation éditoriale

2.8.6.1 Altérations accidentelles

Les contextes `MensuralVoiceContext` et `MensuralStaffContext` permettent de graver des chants dans le style mesuré. Ces contextes initialisent les propriétés de tous les autres contextes et objets graphiques à des valeurs adéquates, de telle sorte que vous pouvez tout de suite vous lancer dans la saisie de votre chant comme ci-après :

```
\score {
  <<
    \new MensuralVoice = "discantus" \transpose c c' {
      \override Score.BarNumber #'transparent = ##t {
        c'1\melisma bes a g\melismaEnd
        f\breve
        \[ f1\melisma a c'\breve d'\melismaEnd \]
        c'\longa
        c'\breve\melisma a1 g1\melismaEnd
```

```
        fis\longa^\signumcongruentiae
      }
    }
    \new Lyrics \lyricsto "discantus" {
      San -- ctus, San -- ctus, San -- ctus
    }
  >>
}
```



2.8.6.2 Notation du rythme dans la musique baroque

2.9 Musiques du monde

2.9.1 Musique arabe

References for Arabic music

Arabic note names

Arabic key signatures

Arabic time signatures

Arabic music example

Autres sources d'information

3 General input and output

3.1 Structure de fichier

3.1.1 Structure d'une partition

3.1.2 Plusieurs partitions dans un même ouvrage

3.1.3 Structure de fichier

3.2 Titres et entêtes

3.2.1 Création de titres

3.2.2 Titres personnalisés

3.2.3 Référence de numéro de page

3.2.4 Table des matières

3.3 Travail sur des fichiers texte

3.3.1 Insertion de fichiers LilyPond

3.3.2 Différentes éditions à partir d'une même source

Utilisation de variables

Using tags

La commande `\tag` affecte un nom à des expressions musicales. Les expressions ainsi balisées pourront être filtrées par la suite. Ce mécanisme permet d'obtenir différentes versions à partir d'une même source musicale.

Dans l'exemple qui suit, nous obtenons deux versions du même extrait, l'une pour le conducteur, l'autre pour l'instrumentiste, et qui comportera les ornements.

```
c1
<<
  \tag #'partie <<
    R1 \\
    {
      \set fontSize = #-1
      c4_"cue" f2 g4 }
  >>
  \tag #'conducteur R1
>>
c1
```

Ce principe peut s'appliquer aux articulations, textes, etc. Il suffit de positionner

`-\tag #`*`votre-balise`*

avant l'articulation, comme ici :

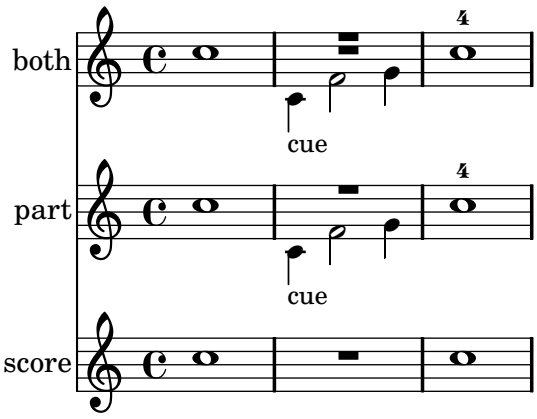
```
c1-\tag #'part ^4
```

Ceci définira une note avec une indication de doigté conditionnelle.

C'est grâce aux commandes `\keepWithTag` et `\removeWithTag` que vous filtrerez les expressions balisées. Par exemple :

```
<<
  de la musique
  \keepWithTag #'score de la musique
  \keepWithTag #'part de la musique
>>
```

donnerait :



Les arguments de la commande `\tag` doivent être un symbole (tel que `#'score` ou `#'part`), suivi d'une expression musicale. Vous pouvez utiliser de multiples balises dans un morceau en saisissant plusieurs `\tag`.

```
\tag #'original-part \tag #'transposed-part ...
```

Problèmes connus et avertissements

Lorsqu'elles comportent des silences, ceux-ci ne seront pas fusionnés si vous imprimez une partition avec les deux sections balisées.

3.3.3 Codage du texte

3.3.4 Affichage de notation au format LilyPond

3.4 Contrôle des sorties

3.4.1 Extraction de fragments musicaux

3.4.2 Ignorer des passages de la partition

3.5 Sortie MIDI

3.5.1 Création de fichiers MIDI

Noms d'instrument

3.5.2 Le bloc MIDI

3.5.3 Éléments pris en compte dans le MIDI

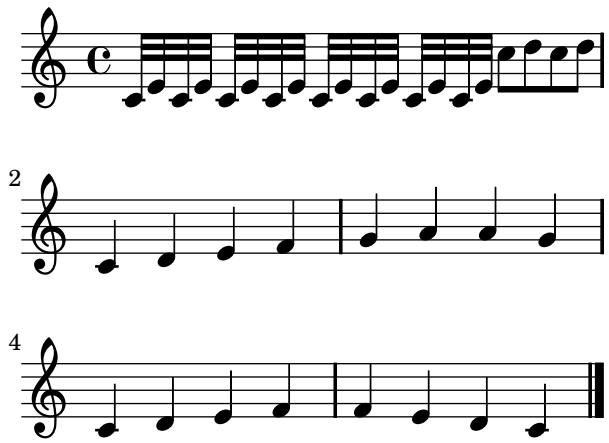
Supported in MIDI

Unsupported in MIDI

3.5.4 Répétitions et MIDI

Au prix de quelques réglages, les reprises de toutes sortes peuvent être rendues dans le fichier MIDI. Il suffit pour cela de recourir à la fonction `\unfoldRepeats`, qui développe toutes les reprises. En d'autre termes, `\unfoldRepeats` transforme toutes les reprises en reprises de type `unfold`.

```
\unfoldRepeats {
  \repeat tremolo 8 {c'32 e' }
  \repeat percent 2 { c''8 d'' }
  \repeat volta 2 {c'4 d' e' f'}
  \alternative {
    { g' a' a' g' }
    {f' e' d' c' }
  }
}
\bar "|."
```



Lorsque l'on veut utiliser `\unfoldRepeats` seulement pour le rendu MIDI, il faut établir deux blocs `\score` : un pour le MIDI, avec des reprises explicites, et l'autre pour la partition, avec des reprises notées sous forme de barres de reprise, de trémolo ou de symboles de pourcentage. Par exemple

```
\score {
 ..musique..
 \layout { ..  }
}
\score {
 \unfoldRepeats ..musique..
 \midi { ..  }
}
```

3.5.5 MIDI et nuances

Indications des nuances

Overall MIDI volume

Equalizing different instruments (i)

Equalizing different instruments (ii)

4 Gestion de l'espace

4.1 Du papier et des pages

4.1.1 Format du papier

4.1.2 Mise en forme de la page

4.2 Mise en forme de la musique

4.2.1 Définition de la taille de portée

4.2.2 Mise en forme de la partition

4.3 Sauts

4.3.1 Sauts de ligne

4.3.2 Sauts de page

4.3.3 Optimisation des sauts de page

4.3.4 Optimisation des tournes

4.3.5 Minimisation des sauts de page

4.3.6 Sauts explicites

4.3.7 Recours à une voix supplémentaire pour gérer les sauts

4.4 Espacement vertical

4.4.1 Espacement vertical au sein d'un système

4.4.2 Espacement vertical entre les systèmes

4.4.3 Positionnement explicite des portées et systèmes

4.4.4 Optimisation du remplissage avec un deuxième passage

4.4.5 Résolution des collisions verticales

4.5 Espacement horizontal

4.5.1 Généralités sur l'espacement horizontal

4.5.2 Changement d'espacement au cours de la partition

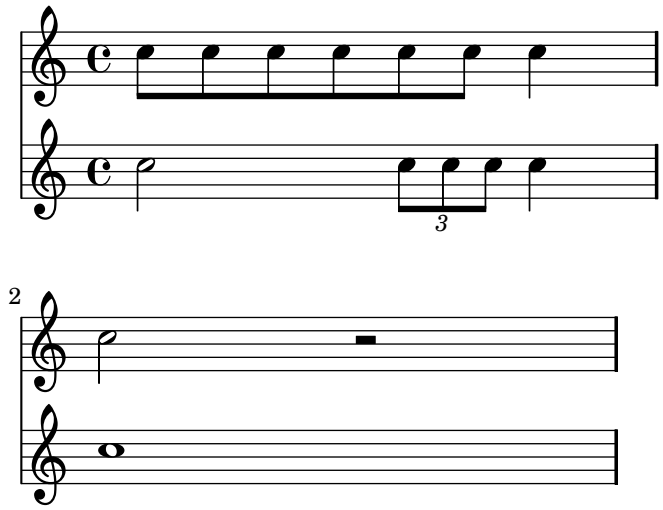
4.5.3 Modification de l'espacement horizontal

4.5.4 Longueur de ligne

4.5.5 Notation proportionnelle

Les notes peuvent s'espacer proportionnellement en assignant une durée à `proportionalNotationDuration`

```
<<
  \set Score.proportionalNotationDuration = #(ly:make-moment 1 16)
  \new Staff { c8[ c c c c c]  c4 c2 r2 }
  \new Staff { c2  \times 2/3 { c8 c c } c4 c1 }
>>
```



Manipuler cette propriété affectera l'espacement idéal uniquement pour des notes consécutives. Pour obtenir une véritable notation proportionnelle, vous devrez tenir compte des réglages suivants :

- La véritable notation proportionnelle exige que des symboles puissent en écraser d'autres. Pour y parvenir, il faut retirer le Section "Separating_line_group_engraver" dans *Référence des propriétés internes* du contexte Section "Staff" dans *Référence des propriétés internes*.
- L'influence en matière d'espacement induite par le formatage (clés, barres de mesure, etc) s'annule en assignant *vrai* (#t) à la propriété `strict-note-spacing` de l'objet Section "SpacingSpanner" dans *Référence des propriétés internes*.
- Les affinages optiques se règlent en assignant *vrai* à la propriété `uniform-stretching` du Section "SpacingSpanner" dans *Référence des propriétés internes*.

Voir aussi

Exemples : Section "Spacing" dans *Exemples de code*.

Le fichier '`input/proportional.ly`' illustre la notation proportionnelle stricte.

4.6 Réduction du nombre de pages de la partition

4.6.1 Mise en évidence de l'espacement

4.6.2 Modification de l'espacement

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 Section 4.6.1 [Mise en évidence de l'espacement], page 172 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
}
```

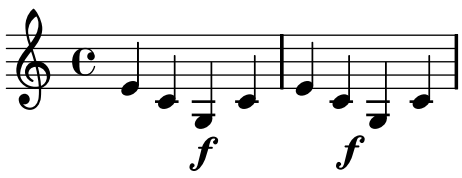
- Obliger LilyPond à mettre un certain nombre de systèmes par page. Par exemple, si LilyPond veut placer onze systèmes dans une page, vous pouvez l'obliger à n'en mettre que dix.

```
\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.

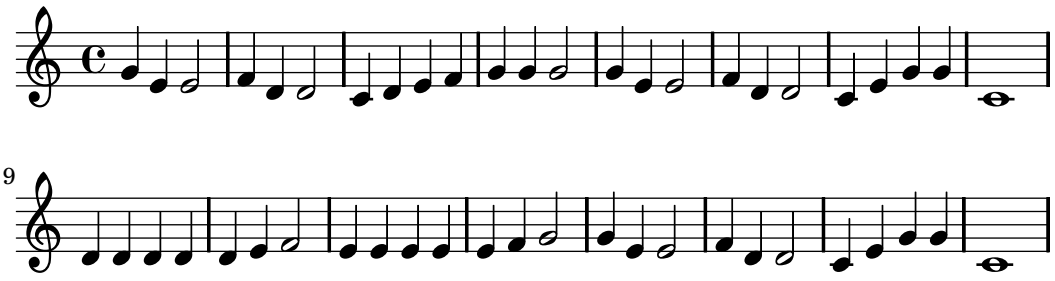
```
\relative c' {
  e4 c g\f c
  \override DynamicLineSpanner #'padding = #-1.8
  \override DynamicText #'extra-offset = #'( -2.1 . 0)
  e4 c g\f c
}
```



- Modifier l'espacement vertical avec `SpacingSpanner`. Reportez-vous à Section 4.5.3 [Modification de l'espacement horizontal], page 171 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)
    }
  }
}
```



5 Modification des réglages prédéfinis

LilyPond est conçu pour engendrer, par défaut, des partitions de la plus haute qualité. Cependant, on peut parfois avoir à modifier cette mise en page par défaut. Celle-ci est réglée par tout un ensemble de « leviers et manettes », dont ce chapitre ne cherche pas à faire l'inventaire exhaustif. Le propos est plutôt ici de mettre en évidence les différents groupes auxquels s'apparentent ces contrôles, et d'expliquer comment trouver le bon levier pour obtenir tel ou tel effet en particulier.

Les moyens de contrôle des différents réglages sont décrits dans un document séparé, la référence du programme Ce guide répertorie toutes les variables, fonctions et autres options que LilyPond met à votre disposition. Il est consultable en ligne, au format HTML, mais est également inclus dans la documentation fournie avec le logiciel.

Il est quatre domaines dans lesquels on peut modifier les réglages par défaut :

- La notation automatique, ce qui revient à modifier la manière dont les éléments de notation sont automatiquement créés – par exemple, les règles de ligatures.
- L'apparence, qui se rapporte aux objets pris individuellement – ainsi de la direction des hampes, du placement des indications textuelles.
- Les contextes, qui recouvrent la manière dont les évènements musicaux sont représentés sous forme de notation – par exemple, le fait d'attribuer un chiffre de mesure distinct à chaque portée.
- La mise en page, autrement dit les marges, l'espacement, la taille du papier ; tous ces facteurs font l'objet des chapitres Chapitre 3 [General input and output], page 167 et Chapitre 4 [Gestion de l'espace], page 171.

En sous-main, LilyPond se sert du langage Scheme (un dérivé du LISP) comme infrastructure. Modifier les choix de mise en page revient à pénétrer dans les entrailles du programme, et de ce fait requiert l'emploi du Scheme. Les fragments de Scheme, dans un fichier `.ly`, sont introduits par le caractère 'hash', (`#`, improprement surnommé 'dièse').[1]

5.1 Contextes d'interprétation

Cette section traite des contextes.

5.1.1 Tout savoir sur les contextes

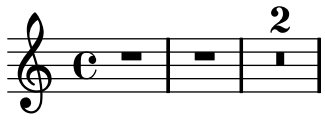
5.1.2 La commande `\set`

Chaque contexte peut avoir plusieurs *propriétés*, c'est-à-dire des variables qu'il inclut. Ces dernières peuvent être modifiées « à la volée », c'est-à-dire pendant que la compilation s'accomplit. C'est là le rôle de la commande `\set`.

```
\set contexte.propriété = #valeur
```

Ainsi :

```
R1*2
\set Score.skipBars = ##t
R1*2
```



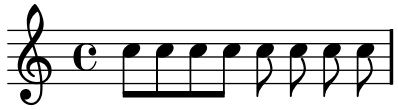
[1] Le Section "Scheme tutorial" dans *Manuel d'initiation* fournit quelques notions de base pour saisir des nombres, des listes, des chaînes de caractères ou des symboles, en Scheme.

Cette commande permet de condenser les mesures vides de notes, en des silences multi-mesures. Il s'agit d'un objet Scheme, auquel on attribue la valeur booléenne 'vrai', c'est-à-dire la lettre `#t` pour 'True' en anglais.

Ce changement étant appliqué 'à la volée', il n'affecte que le second groupe de notes.

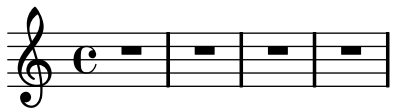
Si l'argument *contexte* n'est pas spécifié, alors la propriété cherchera à s'appliquer dans le contexte le plus restreint où elle est employée : le plus souvent `ChordNames`, `Voice`, ou `Lyrics`. Dans l'exemple suivant,

```
c8 c c c
\set autoBeaming = ##f
c8 c c c
```



aucun argument *contexte* n'a été donné à la commande `\set`. De ce fait, les ligatures automatiques sont désactivées dans le contexte actuel, c'est-à-dire Section "Voice" dans *Référence des propriétés internes*. Notez que le contexte le plus restreint n'est pas toujours le bon, et peut ne pas contenir la propriété qui vous intéresse : ainsi, la propriété `skipBars`, évoquée plus haut, ne relève pas du contexte `Voice`, et le code suivant ne fonctionnera pas.

```
R1*2
\set skipBars = ##t
R1*2
```



Les contextes s'organisent de façon hiérarchique : aussi, lorsqu'un contexte de niveau supérieur est spécifié (par exemple `Staff`), la propriété sera modifée dans tous les contextes inférieurs (tous les contextes `Voice`, par exemple) qu'il contient.

La commande `\unset` permet d'annuler la définition d'une propriété :

`\unset` *`contexte.propriété`*

si et seulement si cette propriété à été définie dans ce *contexte* précis ; ainsi,

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

même s'il s'applique à tous les contextes `Voice` contenus dans le contexte `Staff`, ne peut être annulé au niveau `Voice`. Le code suivant sera sans effet.

```
\unset Voice.autoBeaming
```

En d'autres termes, la commande `\unset` doit impérativement être accompagnée du même contexte que la commande `\set` d'origine. Pour annuler l'effet, dans notre exemple, de `Staff.autoBeaming = ##f`, il faut donc entrer :

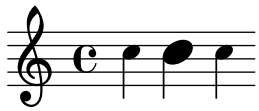
```
\unset Staff.autoBeaming
```

Si l'on se trouve dans le contexte le plus restreint, il n'est pas obligatoire, là encore, de spécifier le *contexte*. Ainsi, les deux lignes suivantes sont équivalentes.

```
\set Voice.autoBeaming = ##t
\set autoBeaming = ##t
```

Pour modifier une propriété de façon à ce qu'elle ne s'applique qu'une seule fois, il convient d'employer la commande `\once` :

```
c4
\once \set fontSize = #4.7
c4
c4
```



Ici le changement de taille est annulé aussitôt après la note concernée.

La référence du programme contient une description exhaustive de toutes les propriétés contexte par contexte : voir Translation ↦ Tunable context properties.

5.1.3 Modification des greffons de contexte

Les contextes, tels que `Score` ou `Staff`, ne contiennent pas que des propriétés ; ils mettent également en œuvre certains sous-programmes (‘plug-ins’, pour employer le terme consacré) nommés ‘graveurs’ (‘engravers’, pour reprendre le terme anglais). Ces sous-programmes sont chargés de créer les différents éléments de notation : On trouve ainsi dans le contexte `Voice`, un graveur `Note_head_engraver`, chargé des têtes de notes, et dans le contexte `Staff`, un graveur `Key_signature_engraver`, chargé de l’armure.

Vous trouverez une description exhaustive de chaque graveur dans Program reference ↦Translation ↦ Engravers. Chaque contexte mentionné dans Program reference ↦Translation ↦ Context. répertorie les graveurs mis en œuvre.

On peut faire, au moyen de ces graveurs, sa propre « cuisine », en modifiant les contextes à volonté.

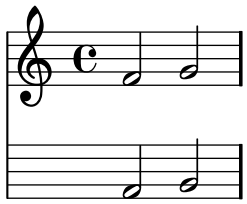
Lorsque un contexte est créé, par la commande `\new` ou `\context`, on peut y adjoindre un bloc `\with` (en anglais ‘avec’), dans lequel il est possible d’ajouter (commande `\consists`) ou d’enlever (commande `\remove`) des graveurs :

```
\new contexte \with {
  \consists ...
  \consists ...
  \remove ...
  \remove ...
  etc.
}
{
  ..musique..
}
```

Ici les points de suspension ... devront être remplacés par les noms des graveurs désirés. Dans l’exemple suivant, on enlève du contexte `Staff`, le chiffre de mesure (graveur `Time_signature_engraver`) et la clé (graveur `Clef_engraver`).

```
<<
  \new Staff {
    f2 g
  }
  \new Staff \with {
     \remove "Time_signature_engraver"
     \remove "Clef_engraver"
  } {
    f2 g2
```

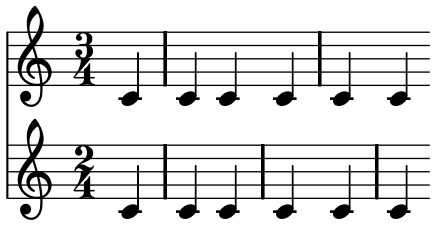
```
  }
>>
```



La clé et le chiffre de mesure ont disparu de la deuxième portée. C'est une méthode quelque peu radicale, puisqu'elle affectera toute la portée jusqu'à la fin de la partition. L'espacement s'en trouve également affecté, ce qui peut être ou non l'effet recherché. Une méthode plus sophistiquée aurait été de rendre ces objets transparents (voir Section "Common tweaks" dans *Manuel d'initiation*).

Dans l'exemple suivant, voici une mise en pratique plus utile. En temps normal, les barres de mesure et la métrique sont synchronisées verticalement dans toute la partition. Les graveurs qui en sont responsables se nomment `Timing_translator` et `Default_bar_line_engraver`. En les enlevant du contexte `Score` pour les attribuer au contexte `Staff`, chaque portée peut désormais avoir sa propre métrique.

```
\new Score \with {
  \remove "Timing_translator"
  \remove "Default_bar_line_engraver"
} <<
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  } {
      \time 3/4
      c4 c c c c c
  }
  \new Staff \with {
    \consists "Timing_translator"
    \consists "Default_bar_line_engraver"
  } {
       \time 2/4
       c4 c c c c c
  }
>>
```



5.1.4 Retouches de mise en forme au sein des contextes

Chaque contexte est chargé de créer plusieurs types d'objets graphiques. Il contient également les réglages nécessaires pour chacun de ces objets. Si l'on modifie ces réglages, les objets n'auront plus la même apparence.

La syntaxe employée pour ce faire est

```
\override contexte.objet #'propriété = #valeur
```

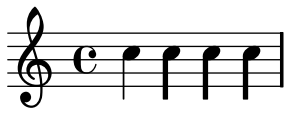
Ici *objet* est un objet graphique, tel que `Stem` (les hampes) ou `NoteHead` (les têtes de note) ; *propriété* est une variable (désignée par un symbole, ce qui explique l'apostrophe) employée par le système de mise en page. La sous-section Section 5.2.1 [Élaboration d'une retouche], page 183 vous aidera à savoir quoi mettre à la place de *objet*, *propriété* et *valeur* ; notre propos n'est ici que d'examiner l'emploi de cette commande.

La commande suivante :

```
\override Staff.Stem #'thickness = #4.0
```

rend les hampes plus épaisses (la valeur par défaut est 1.3, ce qui signifie qu'elles sont 1,3 fois plus épaisses que les lignes de la portée). Dans la mesure où nous avons indiqué `Staff` comme contexte, ce réglage ne s'appliquera qu'à la portée courante ; les autres portées demeureront intactes.

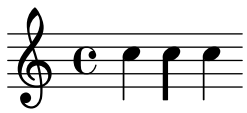
```
c4
\override Staff.Stem #'thickness = #4.0
c4
c4
c4
```



La commande `\override` modifie donc la définition de l'objet `Stem` dans le contexte `Staff` ; toutes les hampes qui suivent seront affectées.

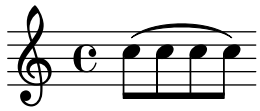
Tout comme avec la commande `\set`, l'argument *contexte* peut être omis, auquel cas le contexte par défaut (ici, `Voice`) sera employé. La commande `\once` permet de n'appliquer la modification qu'une seule fois.

```
c4
\once \override Stem #'thickness = #4.0
c4
c4
```



La commande `\override` doit être entrée *avant* l'objet concerné. Ainsi, lorsque l'on veut altérer un objet qui se prolonge, tel qu'une liaison, une ligature ou tout autre objet dit *Spanner*, la commande `\override` doit être saisie avant que l'objet soit créé.

```
\override Slur #'thickness = #3.0
c8[( c
\override Beam #'thickness = #0.6
c8 c])
```



Dans cet exemple, la liaison (*Slur*) est épaissie, mais non la ligature (*Beam*). En effet, le code qui lui est relatif n'a pas été inséré avant le début de la ligature, et demeure donc sans effet.

De même que la commande `\unset`, la commande `\revert` défait ce qui a été fait par une commande `\override`. Tout comme avec `\unset`, elle ne peut annuler que les réglages effectués dans le même contexte. Ainsi dans l'exemple suivant, la commande `\revert` est sans effet.

```
\override Voice.Stem #'thickness = #4.0
\revert Staff.Stem #'thickness
```

Il existe, à l'intérieur même de certaines propriétés, des options que l'on nomme 'sous-propriétés'. La syntaxe est alors

```
\override contexte.objet #'propriété #'sous-propriété = #valeur
```

Ainsi, par exemple :

```
\override Stem #'details #'beamed-lengths = #'(4 4 3)
```

Voir aussi

Référence du programme : Section "OverrideProperty" dans *Référence des propriétés internes*, Section "RevertProperty" dans *Référence des propriétés internes*, Section "PropertySet" dans *Référence des propriétés internes*, Section "Backend" dans *Référence des propriétés internes*, et Section "All layout objects" dans *Référence des propriétés internes*.

Problèmes connus et avertissements

La sous-couche Scheme ne vérifie pas la saisie des propriétés de façon très stricte. Des références cycliques dans des valeurs Scheme peuvent de ce fait interrompre, ou faire planter le programme – ou bien les deux.

5.1.5 Modification des réglages par défaut d'un contexte

Les réglages montrés dans les sous-sections Section 5.1.2 [La commande set], page 175, Section 5.1.3 [Modification des greffons de contexte], page 177 et Section 5.1.4 [Retouches de mise en forme au sein des contextes], page 178 peuvent également être saisis indépendamment de la musique dans le bloc `\layout`, au moyen de la commande `\context` :

```
\layout {
  ...
  \context {
    \Staff

    \set fontSize = #-2
    \override Stem #'thickness = #4.0
    \remove "Time_signature_engraver"
  }
}
```

Le raccourci `\Staff` invoque les définitions inhérentes au contexte `Staff`, de façon à ce qu'elles puissent être modifiées.

Les lignes suivantes affecteront toutes les portées (tous les contextes `Staff`) dans la partition.

```
\set fontSize = #-2
\override Stem #'thickness = #4.0
\remove "Time_signature_engraver"
```

Les autres contextes peuvent être modifiés de même manière.

La commande `\set`, dans le bloc `\layout`, est facultative ; aussi les lignes suivantes produiront-elles le même effet.

```
\context {
  ...
  fontSize = #-2
}
```

Problèmes connus et avertissements

Il est impossible de stocker des modifications de contexte dans un identificateur.

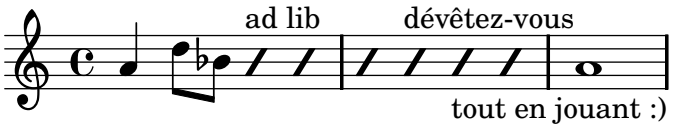
Le raccourci `\RemoveEmptyStaffContext` détruit tous les réglages du contexte `\Staff`. Pour modifier des propriétés de portées gouvernées par `\RemoveEmptyStaffContext`, il faut le faire impérativement *après* avoir invoqué `\RemoveEmptyStaffContext` :

```
\layout {
  \context {
    \RemoveEmptyStaffContext

    \override Stem #'thickness = #4.0
  }
}
```

5.1.6 Définition de nouveaux contextes

Les contextes tels que `Staff` ou `Voice` sont faits de briques de constructions empilées. En combinant divers graveurs, il est possible de créer de nouveaux types de contextes.

Dans l'exemple suivant, on construit, de zéro, un nouveau contexte très semblable à `Voice`, mais qui n'imprime que des têtes de notes en forme de barres obliques au centre de la portée. Un tel contexte, par exemple, peut servir à indiquer un passage improvisé dans un morceau de jazz.



On a rassemblé les réglages dans un bloc `\context`, lui-même dans le bloc `\layout` :

```
\layout {
  \context {
    ...
  }
}
```

En lieu et place des points (. . .), voici les éléments à saisir :

- Tout d'abord, il convient de donner un nom `\name` à notre nouveau contexte :

  ```
  \name ImproVoice
  ```

- Comme il est très semblable à `Voice`, nous souhaitons que toutes les commandes associées au `Voice` déjà existant, restent valables. D'où nécessité de la commande `\alias`, qui va l'associer au contexte `Voice` :

  ```
  \alias Voice
  ```

- Ce contexte doit pouvoir imprimer des notes, et des indications textuelles ; on ajoute donc les graveurs appropriés.

```
\consists Note_heads_engraver
\consists Text_engraver
```

- Cependant, on veut que les notes s'affichent toutes au centre de la portée :

  ```
  \consists Pitch_squash_engraver
  squashedPosition = #0
  ```

  Le graveur Section "Pitch_squash_engraver" dans *Référence des propriétés internes* intercepte les notes créées par Section "Note_heads_engraver" dans *Référence des propriétés internes*, et les 'écrase' pour qu'elles aient toutes la même position verticale, définie par `squashedPosition` : ici il s'agit de la valeur `0`, c'est-à-dire la ligne du milieu.

- On veut que les notes aient la forme d'une barre oblique, sans aucune hampe :

  ```
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
  ```

- Afin que tous ces graveurs puissent travailler de concert, on leur adjoint un sous-programme spécial, introduit par la commande `\type` : il s'agit de `Engraver_group`,

  ```
  \type "Engraver_group"
  ```

Récapitulons – on se retrouve avec le bloc suivant :

```
\context {
  \name ImproVoice
  \type "Engraver_group"
  \consists "Note_heads_engraver"
  \consists "Text_engraver"
  \consists Pitch_squash_engraver
  squashedPosition = #0
  \override NoteHead #'style = #'slash
  \override Stem #'transparent = ##t
  \alias Voice
}
```

Ce n'est pas tout. En effet, on veut intégrer le nouveau contexte `ImproVoice` dans la hiérarchie des contextes. Tout comme le contexte `Voice`, sa place est au sein du contexte `Staff`. Nous allons donc modifier le contexte `Staff`, au moyen de la commande `\accepts` :

```
\context {
  \Staff
  \accepts ImproVoice
}
```

Le contraire de `\accepts` est `\denies`, qui est parfois utile lorsque l'on recycle des définitions de contextes déjà existantes.

Enfin, tout cela doit prendre place dans le bloc `\layout`, comme suit :

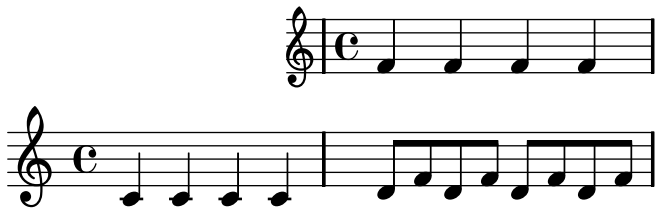
```
\layout {
  \context {
    \name ImproVoice
    ...
  }
  \context {
    \Staff
    \accepts "ImproVoice"
  }
}
```

On peut alors saisir la musique, comme dans l'exemple plus haut :

```
\relative c'' {
  a4 d8 bes8
  \new ImproVoice {
    c4^"ad lib" c
    c4 c^"dévêtez-vous"
    c c_"tout en jouant :)"
  }
  a1
}
```

5.1.7 Alignement des contextes

Il est possible d'aligner verticalement chaque nouveau contexte, en-dessous ou au-dessus, par exemple dans le cas de musique vocale (Section "Vocal ensembles" dans *Manuel d'initiation*) ou d'« ossias ».



5.1.8 Groupement vertical d'objets graphiques

Les objets `VerticalAlignment` et `VerticalAxisGroup` travaillent de concert. Comme leurs noms anglais l'indiquent, `VerticalAxisGroup` regroupe différents objets tels que les portées (`Staff`), les paroles (`Lyrics`) et ainsi de suite ; puis `VerticalAlignment` synchronise verticalement ces différents groupes. En général, il n'y a qu'un seul `VerticalAlignment` pour l'ensemble de la partition, mais chaque contexte `Staff`, `Lyrics`, etc. possède son propre `VerticalAxisGroup`.

5.2 La commande `\override`

La commande `\override` permet de modifier la mise en page en détail. Examinons son utilisation concrètementet dans les détails. La syntaxe de cette commande ressemble généralement à :

`\override` *`contexte.objet #'propriété = #valeur`*

La propriété *propriété* de l'objet *objet*, appartenant au contexte *contexte*, se voit ainsi attribuer la valeur *valeur*.

5.2.1 Élaboration d'une retouche

Les commandes permettant de modifier l'apparence de la partition ressemblent en général à

```
\override Voice.Stem #'thickness = #3.0
```

Pour élaborer un réglage de ce type, on a besoin de connaître précisément :

- le contexte : ici `Voice` (la voix).
- l'objet à affecter : ici `Stem` (les hampes).
- la propriété à modifier : ici `thickness` (l'épaisseur du trait).
- la valeur désirée : ici `3.0` (par défaut, elle est de 1.3).

Certaines 'sous-propriétés' sont parfois contenues dans une propriété. La commande devient alors :

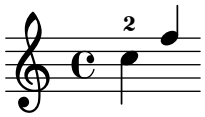
```
\override Stem #'details #'beamed-lengths = #'(4 4 3)
```

Pour bien des propriétés, quel que soit le type de valeur requise, attribuer la valeur ‘faux’ (`##f` en Scheme) reviendra à désactiver complètement l’action de la propriété qui se trouve ainsi purement ignorée par LilyPond. Cela peut s’avérer fort utile pour des propriétés causant des désagréments.

5.2.2 Navigation dans la référence du programme

Comment, par exemple, déplacer le doigté dans le fragment suivant ?

```
c-2
\stemUp
f
```



Sur la page de la documentation relative aux doigtés, c’est-à-dire [Doigtés], page 135, se trouve l’indication suivante :

Voir aussi

Référence du programme : Section “Fingering” dans *Référence des propriétés internes*.

Ladite référence est disponible au format HTML, ce qui rend la navigation bien plus aisée. Il est possible soit de la lire en ligne, soit de la télécharger dans ce format. La démarche présentée ici sera plus difficile à comprendre dans un document au format PDF.

Suivons le lien Section “Fingering” dans *Référence des propriétés internes*. En haut de la nouvelle page, on peut lire

Fingering objects are created by: Section “Fingering_engraver” dans *Référence des propriétés internes* and Section “New_fingering_engraver” dans *Référence des propriétés internes*.

En d’autres termes, *Les indications de doigtés* (`Fingering` en anglais) *sont créées par les graveurs Section “Fingering_engraver” dans Référence des propriétés internes et Section “New_fingering_engraver” dans Référence des propriétés internes.*

En suivant derechef les liens propres à la référence du programme, on suit en fait le cheminement qui aboutit à la création de la partition :

- Section “Fingering” dans *Référence des propriétés internes*: Section “Fingering” dans *Référence des propriétés internes* objects are created by: Section “Fingering_engraver” dans *Référence des propriétés internes*
- Section “Fingering_engraver” dans *Référence des propriétés internes*: Music types accepted: Section “fingering-event” dans *Référence des propriétés internes*
- Section “fingering-event” dans *Référence des propriétés internes*: Music event type `fingering-event` is in Music expressions named Section “FingeringEvent” dans *Référence des propriétés internes*

Ce cheminement se produit, bien sûr, en sens inverse : nous sommes ici partis du résultat, et avons abouti aux évènements (en anglais ‘Events’) engendrés par le fichier d’entrée. L’inverse est également possible : on peut partir d’un évènement, et suivre le cheminement de LilyPond qui aboutit à la création d’un ou plusieurs objets graphiques.

La référence du programme peut également se parcourir comme un document normal. On y trouve des chapitres tels que `Music definitions` Section “Translation” dans *Référence des propriétés internes*, ou encore Section “Backend” dans *Référence des propriétés internes*. Chaque chapitre recense toutes les définitions employées, et les propriétés sujettes à ajustements.

5.2.3 Interfaces de rendu

Tous les éléments de notation sont considérés comme des objets graphiques (en anglais 'Graphical Object', d'où le diminutif *Grob*). Chaque objet est doté d'un certain nombre de propriétés (l'épaisseur du trait, l'orientation, etc.), et lié à d'autres objets. Le fonctionnement de ces objets est décrit en détail dans Section "grob-interface" dans *Référence des propriétés internes*.

Prenons l'exemple des doigtés (en anglais 'Fingering'). La page `Fingering` de la Référence du programme établit une liste de définitions propres à ce type d'objets :

`padding` (dimension, in staff space):

`0.5`

Ce qui signifie que les doigtés doivent être maintenus à une certaine distance (*padding*) des notes : 0,5 unités *staff-space* (espace de portée).

Chaque objet peut avoir plusieurs attributs, en tant qu'élément typographique ou musical. Ainsi, un doigté (objet 'Fingering') possède les attributs suivants :

- Sa taille ne dépend pas de l'espacement horizontal, contrairement aux liaisons ou ligatures.
- C'est du texte – un texte vraiment court, certes.
- Ce texte est imprimé au moyen d'une fonte, contrairement aux liaisons ou ligatures.
- Sur l'axe horizontal, le centre de ce symbole doit être aligné avec le centre de la note.
- Sur l'axe vertical, le symbole doit être proche de la note et de la portée.
- Sur l'axe vertical encore, il doit également s'ordonner avec les éventuels autres symboles, ponctuations, ou éléments textuels.

Faire appliquer ces différents attributs est le rôle des *interfaces*, que l'on trouve en bas de la page Section "Fingering" dans *Référence des propriétés internes*.

This object supports the following interfaces: Section "item-interface" dans *Référence des propriétés internes*, Section "self-alignment-interface" dans *Référence des propriétés internes*, Section "side-position-interface" dans *Référence des propriétés internes*, Section "text-interface" dans *Référence des propriétés internes*, Section "text-script-interface" dans *Référence des propriétés internes*, Section "font-interface" dans *Référence des propriétés internes*, Section "finger-interface" dans *Référence des propriétés internes*, and Section "grob-interface" dans *Référence des propriétés internes*.

En français,

Cet objet admet les interfaces suivantes :

Suit la liste des interfaces en question, présentées comme autant de liens, qui conduisent sur les pages dédiées à chacune d'entre elles. Chaque interface est dotée d'un certain nombre de propriétés, dont certaines peuvent être modifiées, et d'autres non (les 'Internal properties', ou propriétés internes).

Pour aller encore plus loin, plutôt que de simplement parler de l'objet `Fingering`, ce qui ne nous avance pas à grand chose, on peut aller explorer son âme même, dans les fichiers source de LilyPond (voir Section "Other sources of information" dans *Manuel d'initiation*), en l'occurence le fichier '`scm/define-grobs.scm`' :

```
(Fingering
  . ((padding . 0.5)
     (avoid-slur . around)
     (slur-padding . 0.2)
     (staff-padding . 0.5)
     (self-alignment-X . 0)
     (self-alignment-Y . 0)
     (script-priority . 100)
```

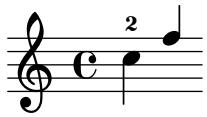
```
(stencil . ,ly:text-interface::print)
(direction . ,ly:script-interface::calc-direction)
(font-encoding . fetaNumber)
(font-size . -5)  ; don't overlap when next to heads.
(meta . ((class . Item)
(interfaces . (finger-interface
               font-interface
               text-script-interface
               text-interface
               side-position-interface
               self-alignment-interface
               item-interface))))))
```

. . .où l'on découvre que l'objet `Fingering` n'est rien de plus qu'un amas de variables et de réglages. La page de la Référence du programme est en fait directement engendrée par cette définition.

5.2.4 Détermination de la propriété de l'objet graphique (grob)

Nous voulions changer la position du chiffre **2** dans le fragment suivant :

```
c-2
\stemUp
f
```



Dans la mesure où le **2** est placé, verticalement, à proximité de la note qui lui correspond, nous allons devoir trouver l'interface en charge de ce placement, qui se trouve être `side-position-interface`. Sur la page de cette interface, on peut lire :

`side-position-interface`

Position a victim object (this one) next to other objects (the support). The property `direction` signifies where to put the victim object relative to the support (left or right, up or down?)

Ce qui signifie

`side-position-interface`

Placer l'objet affecté à proximité d'autres objets. La propriété `direction` indique où placer l'objet (à droite ou à gauche, en haut ou en bas).

En-dessous de cette description se trouve décrite la variable `padding` :

`padding` (dimension, in staff space)
Add this much extra space between objects that are next to each other.

Ce qui signifie

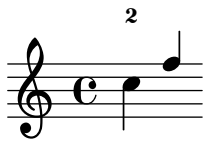
Ajouter tel espace supplémentaire entre des objets proches les uns des autres.

En augmentant la valeur de `padding`, on peut donc éloigner le doigté de la note. La commande suivante insère trois unités d'espace vide entre la note et le doigté :

```
\once \override Voice.Fingering #'padding = #3
```

En ajoutant cette commande avant la création du doigté (de l'objet 'Fingering'), donc avant `c2`, on obtient le résultat suivant :

```
\once \override Voice.Fingering #'padding = #3
c-2
\stemUp
f
```



Dans le cas présent, le réglage intervient dans le contexte `Voice`, ce qui pouvait également se déduire de la Référence du programme, où la page du graveur Section “Fingering_engraver” dans *Référence des propriétés internes* indique :

Fingering_engraver is part of contexts: . . . Section “Voice” dans *Référence des propriétés internes*

Ce qui signifie

Le graveur Fingering_engraver fait partie des contextes : . . . Section “Voice” dans *Référence des propriétés internes*

5.2.5 La commande `\set`

Dans certains cas, on peut passer par un raccourci pour arranger les objets graphiques. Lorsqu'un objet est directement engendré par un élément distinct du fichier source, on peut utiliser la commande `\tweak`.

Dans l'accord suivant, les notes sont modifiées une par une :

```
<
  c
  \tweak #'color #red d
  g
  \tweak #'duration-log #1  a
>4-\tweak #'padding #10 -.
```

Comme on peut le voir, les propriétés sont ici modifiées directement en même temps que les objets sont saisis. Il n'est plus besoin de spécifier ni le nom de l'objet (*grob*), ni le contexte dans lequel cela doit s'appliquer.

Ce procédé ne marche que pour des objets directement liés aux évènements (Section “Event” dans *Référence des propriétés internes*) du fichier source. Par exemple :

- Les têtes de notes au sein d'un accord, qui sont directement engendrées par les hauteurs indiquées
- Les signes d'articulation, engendrés par les indications de ponctuation.

En revanche, les hampes ou les altérations sont engendrées par les têtes de notes, et non par des évènements dans le fichier source. De même pour les clés, qui ne sont pas directement engendrées par le fichier source, mais plutôt par le changement d'une propriété interne.

En fait, très peu d'objets passent *directement* du code source à la partition. Une note toute simple, par exemple `c4`, fait l'objet d'un traitement et n'est donc pas directement rendue ; c'est pourquoi le code suivant ne sera d'aucun effet :

```
\tweak #'color #red c4
```

Voir pour plus de détails Section 6.3.1 [Afficher des expressions musicales], page 194.

5.2.6 Utilisation de code Scheme au lieu de `\tweak`

L'inconvénient principal de la commande `\tweak` est la rigidité de sa syntaxe. Par exemple, le code suivant produit une erreur.

```
F = \tweak #'font-size #-3 -\flageolet

\relative c'' {
  c4^\F c4_\F
}
```

En d'autres termes, `\tweak` ne se comporte pas comme une articulation : il ne peut notamment pas être accolé avec les symboles '`^`' ou '`_`'.

C'est en se servant du langage Scheme que l'on peut résoudre ce problème. Dans cet exemple, on a recours aux méthodes décrites dans Section 6.3.4 [Exemple : ajouter une articulation à plusieurs notes], page 194, en particulier quant à l'emploi de `\displayMusic`.

```
F = #(let ((m (make-music 'ArticulationEvent
                          'articulation-type "flageolet")))
       (set! (ly:music-property m 'tweaks)
             (acons 'font-size -3
                    (ly:music-property m 'tweaks)))
       m)

\relative c'' {
  c4^\F c4_\F
}
```

Ici les propriétés `tweak` de l'objet flageolet nommé '`m`' (créé au moyen de `make-music`) sont extraites par `ly:music-property`, une nouvelle valeur de la taille de fonte est ajoutée à la liste de ses propriétés (grâce à la fonction Scheme `acons`), et le résultat de cette opération est renvoyé par `set!`. Le dernier élément, dans ce bloc `let`, est la valeur finale de '`m`' lui-même.

5.2.7 `\set` ou `\override`

Si les propriétés peuvent être modifiées de deux façons, par les commandes `\set` et `\override`, c'est qu'il y a deux types de propriétés.

Les contextes peuvent avoir des propriétés, dont les noms commencent par une minuscule puis comprennent une ou plusieurs majuscules (de style `totoTutu`). Elles ont surtout trait à la notation des éléments musicaux : par exemple, `localKeySignature` permet de choisir s'il faut ou non imprimer une altération, ou `measurePosition` permet de choisir quand il faut imprimer une barre de mesure. Ces propriétés de contextes sont appelées à changer au long de l'interprétation de la partition : `measurePosition` en est un exemple évident. Ces propriétés doivent être modifiées avec la commande `\set`.

Il existe un type particulier de propriétés : les descriptions d'éléments. Ces propriétés, dont les noms commencent par une majuscule, puis comprennent une ou plusieurs majuscules (de style `TotoTata`), contiennent les réglages 'par défaut' pour les objets graphiques. Ces réglages sont sous forme de liste Scheme ; on peut les consulter dans le fichier '`scm/define-grobs.scm`'. Les descriptions d'éléments doivent être modifiées avec la commande `\override`.

`\override` est en fait un raccourci :

`\override` *`contexte.objet #'propriété = #valeur`*

est plus ou moins l'équivalent de

`\set` *`contexte.objet #'propriété = #(cons (cons 'propriété valeur) <valeur précédente de contexte)`*

La valeur de `context` (la liste Scheme, ou 'alist') sert à initialiser les propriétés des objets un par un. Les objets eux-même ont leurs propriétés, dont les noms, dans la tradition du langage Scheme, comprennent un trait d'union (`toto-titi`). Ces propriétés internes changent constamment au cours de la mise en page : en fait, la gravure d'une page n'est autre que le calcul de toutes ces propriétés, au moyen de fonctions de rappel.

La propriété `fontSize` est une exception : c'est un raccourci, qui équivaudrait à saisir `\override ... #'font-size` pour tous les objets textuels. Dans la mesure où il s'agit d'une manipulation très courante, une propriété spéciale a été créée. Elle doit être modifiée avec la commande `\set`.

5.2.8 Retouches complexes

Certains réglages sont plus délicats que d'autres.

- L'un d'entre eux est l'apparence des objets dits 'spanner', qui s'étendent horizontalement, tels que les liaisons. Si, en principe, un seul de ces objets est créé à la fois et peut donc être modifié de façon habituelle, lorsque ces objets doivent enjamber un changement de ligne, ils sont dupliqués au début du ou des systèmes suivants. Comme ces objets sont des clones de l'objet d'origine, ils en héritent toutes les propriétés, y compris les éventuelles commandes `\override`.

  En d'autres termes, une commande `\override` affecte toujours les deux extrémités d'un objet 'spanner'. Pour ne modifier que la partie précédant ou suivant le changement de ligne, il faut intervenir directement dans le processus de mise en page. La fonction de rappel `after-line-breaking` contient toute l'opération Scheme effectuée lorsque les sauts de lignes ont été déterminés, et que des objets graphiques ont été divisés sur des systèmes différents.

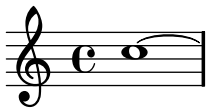
  Dans l'exemple suivant, on définit une nouvelle opération nommée `my-callback`. Cette opération

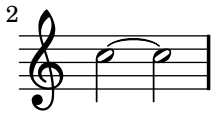
  - détermine si l'objet a été divisé à l'occasion d'un changement de ligne
  - si oui, recherche les différents morceaux de l'objet
  - vérifie si l'objet considéré est bien la deuxième moitié d'un objet divisé
  - si oui, applique un espacement supplémentaire (`extra-offset`).

  On ajoute cette procédure à l'objet Section "Tie" dans *Référence des propriétés internes* (liaison de tenue), de façon à ce que le deuxième morceau d'une liaison divisée soit rehaussé.

```
#(define (my-callback grob)
  (let* (
         ; l'objet a-t-il été divisé ?
         (orig (ly:grob-original grob))

         ; si oui, rechercher les morceaux frères (siblings)
         (siblings (if (ly:grob? orig)
                       (ly:spanner-broken-into orig) '() )))

    (if (and (>= (length siblings) 2)
             (eq? (car (last-pair siblings)) grob))
      (ly:grob-set-property! grob 'extra-offset '(-2 . 5)))))

\relative c'' {
```

```
  \override Tie #'after-line-breaking =
  #my-callback
  c1 ~ \break c2 ~ c
}
```





Lorsque cette astuce va être appliquée, notre nouvelle fonction de rappel `after-line-breaking` devra également appeler celle d'origine (`after-line-breaking`), si elle existe. Ainsi, pour l'utiliser dans le cas d'un crescendo (objet `Hairpin`), il faudra appeler également `ly:hairpin::after-line-breaking`.

- Pour des raisons d'ordre technique, certains objets ne peuvent être modifiés par `\override`. Parmi ceux-là, les objets `NonMusicalPaperColumn` et `PaperColumn`. La commande `\overrideProperty` sert à les modifier, de façon similaire à `\once \override`, mais avec une syntaxe différente :

```
\overrideProperty
#"Score.NonMusicalPaperColumn"  % Nom de l'objet
#'line-break-system-details     % Nom de la propriété
#'((next-padding . 20))         % valeur
```

Notez cependant que la commande `\override` peut tout de même être appliquée à `NoteMusicalPaperColumn` et `PaperColumn` dans un bloc `\context`.

6 Interfaces pour les programmeurs

6.1 Fonctions musicales

6.1.1 Aperçu des fonctions musicales

6.1.2 Fonctions de substitution simple

6.1.3 Fonctions de substitution par paire

6.1.4 De l'usage des mathématiques dans les fonctions

6.1.5 Fonctions fantômes

6.1.6 Fonctions dépourvues d'argument

6.1.7 Liste des fonctions musicales prédéfinies

`acciaccatura` - *music* (music)
(undocumented; fixme)

`addChordShape` - *key-symbol* (symbol) *shape-string* (string)
(undocumented; fixme)

`addInstrumentDefinition` - *name* (string) *lst* (list)
(undocumented; fixme)

`addQuote` - *name* (string) *music* (music)
(undocumented; fixme)

`afterGrace` - *main* (music) *grace* (music)
(undocumented; fixme)

`allowPageTurn`
(undocumented; fixme)

`applyContext` - *proc* (procedure)
(undocumented; fixme)

`applyMusic` - *func* (procedure) *music* (music)
(undocumented; fixme)

`applyOutput` - *ctx* (symbol) *proc* (procedure)
(undocumented; fixme)

`appoggiatura` - *music* (music)
(undocumented; fixme)

`assertBeamQuant` - *l* (pair) *r* (pair)
(undocumented; fixme)

`assertBeamSlope` - *comp* (procedure)
(undocumented; fixme)

`autochange` - *music* (music)
(undocumented; fixme)

`balloonGrobText` - *grob-name* (symbol) *offset* (pair of numbers) *text* (markup)
(undocumented; fixme)

`balloonText` - *offset* (pair of numbers) *text* (markup)
(undocumented; fixme)

`bar` - *type* (string)
(undocumented; fixme)

`barNumberCheck` - *n* (integer)
(undocumented; fixme)

`bendAfter` - *delta* (unknown)
(undocumented; fixme)

`breathe` (undocumented; fixme)

`clef` - *type* (string)
(undocumented; fixme)

`cueDuring` - *what* (string) *dir* (direction) *main-music* (music)
(undocumented; fixme)

`displayLilyMusic` - *music* (music)
(undocumented; fixme)

`displayMusic` - *music* (music)
(undocumented; fixme)

`endSpanners` - *music* (music)
(undocumented; fixme)

`featherDurations` - *factor* (moment) *argument* (music)
(undocumented; fixme)

`grace` - *music* (music)
(undocumented; fixme)

`includePageLayoutFile`
(undocumented; fixme)

`instrumentSwitch` - *name* (string)
(undocumented; fixme)

`keepWithTag` - *tag* (symbol) *music* (music)
(undocumented; fixme)

`killCues` - *music* (music)
(undocumented; fixme)

`label` - *label* (symbol)
(undocumented; fixme)

`makeClusters` - *arg* (music)
(undocumented; fixme)

`musicMap` - *proc* (procedure) *mus* (music)
(undocumented; fixme)

`noPageBreak`
(undocumented; fixme)

`noPageTurn`
(undocumented; fixme)

`octaveCheck` - *pitch-note* (music)
(undocumented; fixme)

`oldaddlyrics` - *music* (music) *lyrics* (music)
(undocumented; fixme)

`ottava` - *octave* (number)
(undocumented; fixme)

`overrideProperty` - *name* (string) *property* (symbol) *value* (any type)
(undocumented; fixme)

`pageBreak`
(undocumented; fixme)

`pageTurn` (undocumented; fixme)

`parallelMusic` - *voice-ids* (list) *music* (music)
(undocumented; fixme)

`parenthesize` - *arg* (music)
(undocumented; fixme)

`partcombine` - *part1* (music) *part2* (music)
(undocumented; fixme)

`pitchedTrill` - *main-note* (music) *secondary-note* (music)
(undocumented; fixme)

`pointAndClickOff`
(undocumented; fixme)

`pointAndClickOn`
(undocumented; fixme)

`quoteDuring` - *what* (string) *main-music* (music)
(undocumented; fixme)

`removeWithTag` - *tag* (symbol) *music* (music)
(undocumented; fixme)

`resetRelativeOctave` - *reference-note* (music)
(undocumented; fixme)

`rightHandFinger` - *finger* (number or string)
(undocumented; fixme)

`scaleDurations` - *fraction* (pair of numbers) *music* (music)
(undocumented; fixme)

`scoreTweak` - *name* (string)
(undocumented; fixme)

`shiftDurations` - *dur* (integer) *dots* (integer) *arg* (music)
(undocumented; fixme)

`spacingTweaks` - *parameters* (list)
(undocumented; fixme)

`storePredefinedDiagram` - *chord* (music) *tuning* (list) *terse-definition* (string)
(undocumented; fixme)

`tag` - *tag* (symbol) *arg* (music)
(undocumented; fixme)

`tocItem` - *text* (markup)
Add a line to the table of content, using the `tocItemMarkup` paper variable markup

`transposedCueDuring` - *what* (string) *dir* (direction) *pitch-note* (music) *main-music* (music)
(undocumented; fixme)

`transposition` - *pitch-note* (music)
(undocumented; fixme)

`tweak` - *sym* (symbol) *val* (any type) *arg* (music)
(undocumented; fixme)

`unfoldRepeats` - *music* (music)
(undocumented; fixme)

`withMusicProperty` - *sym* (symbol) *val* (any type) *music* (music)
(undocumented; fixme)

6.2 Interfaces de programmation

6.2.1 Variables d'entrée et Scheme

6.2.2 Représentation interne de la musique

6.3 Construction de fonctions complexes

6.3.1 Afficher des expressions musicales

6.3.2 Propriétés de la musique

6.3.3 Exemple : redoubler une note avec liaison

6.3.4 Exemple : ajouter une articulation à plusieurs notes

6.4 Interface de programmation des marqueurs de texte

6.4.1 Construction Scheme d'un marqueur

6.4.2 Fonctionnement interne des marqueurs

6.4.3 Définition d'une nouvelle commande de marqueur

6.4.4 Définition d'une nouvelle commande de liste de marqueurs

6.5 Contextes pour programmeurs

6.5.1 Évaluation d'un contexte

6.5.2 Application d'une fonction à tous les objets de mise en forme

6.6 Utilisation de procédures Scheme comme propriétés

Annexe A Bibliographie

Annexe B Tables du manuel de notation

B.1 Table des noms d'accord

B.2 Instruments MIDI

B.3 Liste des couleurs

Couleurs de base

Noms de couleur X

Noms de couleur sans suffixe numérique

Noms de couleur avec suffixe numérique

Échelle de gris

B.4 La fonte Feta

B.5 Styles de tête de note

B.6 Text markup commands

The following commands can all be used inside `\markup { }`.

B.6.1 Font

`\abs-fontsize` *size* (number) *arg* (markup)

Use *size* as the absolute font size to display *arg*. Adjust baseline skip and word space accordingly.

```
\markup {
  default text font size
  \hspace #2
  \abs-fontsize #16 { text font size 16 }
  \hspace #2
  \abs-fontsize #12 { text font size 12 }
}
```

default text font size text font size 16 text font size 12

`\bigger` *arg* (markup)

Increase the font size relative to current setting.

```
\markup {
  \huge {
    huge
    \hspace #2
    \bigger {
      bigger
    }
```

```
    \hspace #2
    huge
  }
}
```

huge bigger huge

`\bold` *arg* (markup)
: Switch to bold font-series.

```
\markup {
  default
  \hspace #2
  \bold
  bold
}
```

default **bold**

`\box` *arg* (markup)
: Draw a box round *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \override #'(box-padding . 0.5)
  \box
  \line { V. S. }
}
```

V. S.

Used properties:

- `box-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\caps` *arg* (markup)
: Copy of the `\smallCaps` command.

```
\markup {
  default
  \hspace #2
  \caps {
    Text in small caps
  }
}
```

default TEXT IN SMALL CAPS

`\dynamic` *arg* (markup)
: Use the dynamic font. This font only contains **s**, **f**, **m**, **z**, **p**, and **r**. When producing phrases, like 'più **f**', the normal words (like 'più') should be done in a different font. The recommended font for this is bold and italic.

```
\markup {
  \dynamic {
    sfzp
  }
}
```

sfzp

`\finger` *arg* (markup)
Set the argument as small numbers.

```
\markup {
  \finger {
    1 2 3 4 5
  }
}
```

1 2 3 4 5

`\fontCaps` *arg* (markup)
Set `font-shape` to `caps`

Note: `\fontCaps` requires the installation and selection of fonts which support the `caps` font shape.

`\fontsize` *increment* (number) *arg* (markup)
Add *increment* to the font-size. Adjust baseline skip accordingly.

```
\markup {
  default
  \hspace #2
  \fontsize #-1.5
  smaller
}
```

default smaller

Used properties:

- `baseline-skip` (2)
- `word-space` (1)
- `font-size` (0)

`\huge` *arg* (markup)
Set font size to +2.

```
\markup {
  default
  \hspace #2
  \huge
  huge
}
```

default huge

`\italic` *arg* (markup)
Use italic `font-shape` for *arg*.

```
\markup {
  default
  \hspace #2
  \italic
  italic
}
```

default *italic*

`\large` *arg* (markup)
Set font size to +1.

```
\markup {
  default
  \hspace #2
  \large
  large
}
```

default large

`\larger` *arg* (markup)
Copy of the `\bigger` command.

```
\markup {
  default
  \hspace #2
  \larger
  larger
}
```

default larger

`\magnify` *sz* (number) *arg* (markup)
Set the font magnification for its argument. In the following example, the middle A is 10% larger:

```
A \magnify #1.1 { A } A
```

Note: Magnification only works if a font name is explicitly selected. Use `\fontsize` otherwise.

```
\markup {
  default
  \hspace #2
  \magnify #1.5 {
    50% larger
  }
}
```

default 50% larger

`\medium` *arg* (markup)
Switch to medium font series (in contrast to bold).

```
\markup {
  \bold {
    some bold text
    \hspace #2
    \medium {
      medium font series
    }
    \hspace #2
    bold again
  }
}
```

some bold text medium font series **bold again**

`\normal-size-sub` *arg* (markup)
Set *arg* in subscript, in a normal font size.

```
\markup {
  default
  \normal-size-sub {
    subscript in standard size
  }
}
```

default $_{\text{subscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-size-super` *arg* (markup)
Set *arg* in superscript with a normal font size.

```
\markup {
  default
  \normal-size-super {
    superscript in standard size
  }
}
```

default $^{\text{superscript in standard size}}$

Used properties:

- `baseline-skip`

`\normal-text` *arg* (markup)
Set all font related properties (except the size) to get the default normal text font, no matter what font was used earlier.

```
\markup {
  \huge \bold \sans \caps {
    Some text with font overrides
    \hspace #2
    \normal-text {
      Default text, same font-size
```

```
    }
    \hspace #2
    More text as before
  }
}
```

SOME TEXT WITH FONT OVERRIDES Default text, same font-size **MORE**

`\normalsize` *arg* (markup)
Set font size to default.

```
\markup {
  \teeny {
    this is very small
    \hspace #2
    \normalsize {
      normal size
    }
    \hspace #2
    teeny again
  }
}
```

this is very small normal size teeny again

`\number` *arg* (markup)
Set font family to `number`, which yields the font used for time signatures and fingerings. This font only contains numbers and some punctuation. It doesn't have any letters.

```
\markup {
  \number {
    0 1 2 3 4 5 6 7 8 9 . ,
  }
}
```

0123456789.,

`\roman` *arg* (markup)
Set font family to `roman`.

```
\markup {
  \sans \bold {
    sans serif, bold
    \hspace #2
    \roman {
      text in roman font family
    }
    \hspace #2
    return to sans
  }
}
```

sans serif, bold **text in roman font family** **return to sans**

`\sans` *arg* (markup)
Switch to the sans serif family.

```
\markup {
  default
  \hspace #2
  \sans {
    sans serif
  }
}
```

default sans serif

`\simple` *str* (string)
A simple text string; `\markup { foo }` is equivalent with `\markup { \simple #"foo" }`.

Note: for creating standard text markup or defining new markup commands, the use of `\simple` is unnecessary.

```
\markup {
  \simple #"simple"
  \simple #"text"
  \simple #"strings"
}
```

simple text strings

`\small` *arg* (markup)
Set font size to -1.

```
\markup {
  default
  \hspace #2
  \small
  small
}
```

default small

`\smallCaps` *text* (markup)
Emit *arg* as small caps.

Note: `\smallCaps` does not support accented characters.

```
\markup {
  default
  \hspace #2
  \smallCaps {
    Text in small caps
  }
}
```

default Text in small caps

`\smaller` *arg* (markup)
Decrease the font size relative to current setting.

```
\markup {
  \fontsize #3.5 {
    some large text
    \hspace #2
    \smaller {
      a bit smaller
    }
    \hspace #2
    more large text
  }
}
```

some large text a bit smaller more large text

`\sub` *arg* (markup)
Set *arg* in subscript.

```
\markup {
  \concat {
    H
    \sub {
      2
    }
    O
  }
}
```

H_2O

Used properties:

- `baseline-skip`
- `font-size` (0)

`\super` *arg* (markup)
Raising and lowering texts can be done with `\super` and `\sub`:

```
\markup {
  E =
  \concat {
    mc
    \super
    2
  }
}
```

$E = mc^2$

Used properties:

- `baseline-skip`
- `font-size` (0)

`\teeny` *arg* (markup)
Set font size to -3.

```
\markup {
  default
  \hspace #2
  \teeny
  teeny
}
```

default teeny

`\text` *arg* (markup)
Use a text font instead of music symbol or music alphabet font.

```
\markup {
  \number {
    1, 2,
    \text {
      three, four,
    }
    5
  }
}
```

1, 2, three, four, **5**

`\tiny` *arg* (markup)
Set font size to -2.

```
\markup {
  default
  \hspace #2
  \tiny
  tiny
}
```

default tiny

`\typewriter` *arg* (markup)
Use `font-family` typewriter for *arg*.

```
\markup {
  default
  \hspace #2
  \typewriter
  typewriter
}
```

default `typewriter`

`\underline` *arg* (markup)
Underline *arg*. Looks at `thickness` to determine line thickness and y offset.

```
\markup {
  default
  \hspace #2
  \override #'(thickness . 2)
```

```
  \underline {
    underline
  }
}
```

default underline

Used properties:

- `thickness` (1)

`\upright` *arg* (markup)
Set font shape to `upright`. This is the opposite of `italic`.

```
\markup {
  \italic {
    italic text
    \hspace #2
    \upright {
      upright text
    }
    \hspace #2
    italic again
  }
}
```

italic text upright text *italic again*

B.6.2 Align

`\center-align` *arg* (markup)
Align `arg` to its X center.

```
\markup {
  \column {
    one
    \center-align
    two
    three
  }
}
```

one
two
three

`\center-column` *args* (list of markups)
Put `args` in a centered column.

```
\markup {
  \center-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\column` *args* (list of markups)

Stack the markups in *args* vertically. The property `baseline-skip` determines the space between each markup in *args*.

```
\markup {
  \column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\combine` *m1* (markup) *m2* (markup)

Print two markups on top of each other.

Note: `\combine` cannot take a list of markups enclosed in curly braces as an argument; the follow example will not compile:

```
\combine { a list }
```

```
\markup {
  \fontsize #5
  \override #'(thickness . 2)
  \combine
    \draw-line #'(0 . 4)
    \arrow-head #Y #DOWN ##f
}
```

`\concat` *args* (list of markups)

Concatenate *args* in a horizontal line, without spaces inbetween. Strings and simple markups are concatenated on the input level, allowing ligatures. For example, `\concat { "f" \simple #"i" }` is equivalent to `"fi"`.

```
\markup {
  \concat {
    one
    two
    three
  }
}
```

onetwothree

`\dir-column` *args* (list of markups)
Make a column of args, going up or down, depending on the setting of the `#'direction` layout property.

```
\markup {
  \override #'(direction . 1) {
    \dir-column {
      going up
    }
  }
  \dir-column {
    going down
  }
}
```

up
going going
down

Used properties:

- `baseline-skip`
- `direction`

`\fill-line` *markups* (list of markups)
Put *markups* in a horizontal line of width *line-width*. The markups are spaced or flushed to fill the entire line. If there are no arguments, return an empty stencil.

```
\markup {
  \column {
    \fill-line {
      Words evenly spaced across the page
    }
    \null
    \fill-line {
      \line { Text markups }
      \line {
        \italic { evenly spaced }
      }
      \line { across the page }
    }
  }
}
```

Words evenly spaced across the page

Text markups *evenly spaced* across the page

Used properties:

- `line-width` (#f)
- `word-space` (1)
- `text-direction` (1)

`\general-align` *axis* (integer) *dir* (number) *arg* (markup)
Align *arg* in *axis* direction to the *dir* side.

```
\markup {
  \column {
    one
    \general-align #X #LEFT
    two
    three
    \null
    one
    \general-align #X #CENTER
    two
    three
    \null
    \line {
      one
      \general-align #Y #UP
      two
      three
    }
    \null
    \line {
      one
      \general-align #Y #3.2
      two
      three
    }
  }
}
```

one
two
three

one
two
three

one two three

one three
two

\halign *dir* (number) *arg* (markup)
Set horizontal alignment. If *dir* is `-1`, then it is left-aligned, while `+1` is right. Values in between interpolate alignment accordingly.

```
\markup {
  \column {
    one
    \halign #LEFT
    two
    three
    \null
```

```
    one
    \halign #CENTER
    two
    three
    \null
    one
    \halign #RIGHT
    two
    three
    \null
    one
    \halign #-5
    two
    three
  }
}
```

one
two
three

one
two
three

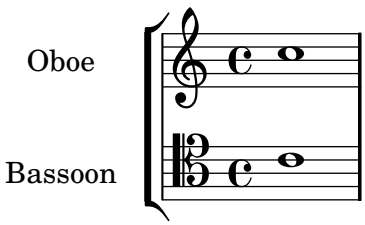
one
two
three

one
two
three

`\hcenter-in` *length* (number) *arg* (markup)
: Center *arg* horizontally within a box of extending *length*/2 to the left and right.

```
\new StaffGroup <<
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Oboe
    }
    c''1
  }
  \new Staff {
    \set Staff.instrumentName = \markup {
      \hcenter-in #12
      Bassoon
    }
    \clef tenor
    c'1
```

```
  }
>>
```



`\hspace` *amount* (number)

This produces an invisible object taking horizontal space. For example,

```
\markup { A \hspace #2.0 B }
```

puts extra space between A and B, on top of the space that is normally inserted before elements on a line.

```
\markup {
  one
  \hspace #2
  two
  \hspace #8
  three
}
```

one two three

`\justify-field` *symbol* (symbol)

Justify the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  descr = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \justify-field #'header:descr
    }
  }
}

\markup {
  \null
}
```

My title

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

`\justify` *args* (list of markups)

Like wordwrap, but with lines stretched to justify the margins. Use `\override #'(line-width . X)` to set the line width; X is the number of staff spaces.

```
\markup {
  \justify {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\justify-string` *arg* (string)

Justify a string. Paragraphs may be separated with double newlines

```
\markup {
  \override #'(line-width . 40)
  \justify-string #"Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut labore
    et dolore magna aliqua.


    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.


    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

`\left-align` *arg* (markup)
Align *arg* on its left edge.

```
\markup {
  \column {
    one
    \left-align
    two
    three
  }
}
```

one
two
three

`\left-column` *args* (list of markups)
Put `args` in a left-aligned column.

```
\markup {
  \left-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\line` *args* (list of markups)
Put *args* in a horizontal line. The property `word-space` determines the space between each markup in *args*.

```
\markup {
  \line {
    one two three
  }
}
```

one two three

Used properties:

- `text-direction` (1)
- `word-space`

`\lower` *amount* (number) *arg* (markup)
: Lower *arg* by the distance *amount*. A negative *amount* indicates raising; see also `\raise`.

```
\markup {
  one
  \lower #3
  two
  three
}
```

one three
two

`\pad-around` *amount* (number) *arg* (markup)
: Add padding *amount* all around *arg*.

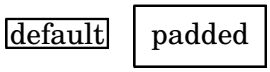
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #0.5 {
      padded
    }
  }
}
```

default padded

`\pad-markup` *padding* (number) *arg* (markup)
: Add space around a markup object.

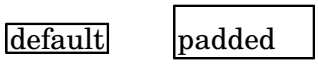
```
\markup {
  \box {
    default
  }
  \hspace #2
  \box {
    \pad-around #1 {
      padded
```

```
    }
  }
}
```



`\pad-to-box` *x-ext* (pair of numbers) *y-ext* (pair of numbers) *arg* (markup)
Make *arg* take at least *x-ext*, *y-ext* space.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-to-box #'(0 . 10) #'(0 . 3) {
      padded
    }
  }
}
```



`\pad-x` *amount* (number) *arg* (markup)
Add padding *amount* around *arg* in the X direction.

```
\markup {
  \box {
    default
  }
  \hspace #4
  \box {
    \pad-x #2 {
      padded
    }
  }
}
```

`\put-adjacent` *axis* (integer) *dir* (direction) *arg1* (markup) *arg2* (markup)
Put *arg2* next to *arg1*, without moving *arg1*.

`\raise` *amount* (number) *arg* (markup)
Raise *arg* by the distance *amount*. A negative *amount* indicates lowering, see also `\lower`.

The argument to `\raise` is the vertical displacement amount, measured in (global) staff spaces. `\raise` and `\super` raise objects in relation to their surrounding markups.

If the text object itself is positioned above or below the staff, then `\raise` cannot be used to move it, since the mechanism that positions it next to the staff cancels any shift made with `\raise`. For vertical positioning, use the `padding` and/or `extra-offset` properties.

```
\markup {
  C
  \small
  \bold
  \raise #1.0
  9/7+
}
```

C **9/7+**

`\right-align` *arg* (markup)
Align *arg* on its right edge.

```
\markup {
  \column {
    one
    \right-align
    two
    three
  }
}
```

one
two
three

`\right-column` *args* (list of markups)
Put `args` in a right-aligned column.

```
\markup {
  \right-column {
    one
    two
    three
  }
}
```

one
two
three

Used properties:

- `baseline-skip`

`\rotate` *ang* (number) *arg* (markup)
Rotate object with *ang* degrees around its center.

```
\markup {
  default
  \hspace #2
  \rotate #45
  \line {
    rotated 45˚
  }
}
```

default rotated 45°

`\translate` *offset* (pair of numbers) *arg* (markup)
: This translates an object. Its first argument is a cons of numbers.

```
A \translate #(cons 2 -3) { B C } D
```

This moves 'B C' 2 spaces to the right, and 3 down, relative to its surroundings. This command cannot be used to move isolated scripts vertically, for the same reason that `\raise` cannot be used for that.

```
\markup {
  *
  \translate #'(2 . 3)
  \line { translated two spaces right, three up }
}
```

translated two spaces right, three up
*

`\translate-scaled` *offset* (pair of numbers) *arg* (markup)
: Translate *arg* by *offset*, scaling the offset by the `font-size`.

```
\markup {
  \fontsize #5 {
    * \translate #'(2 . 3) translate
    \hspace #2
    * \translate-scaled #'(2 . 3) translate-scaled
  }
}
```

* translate * translate-scaled

Used properties:

- `font-size` (0)

`\vcenter` *arg* (markup)
: Align `arg` to its Y center.

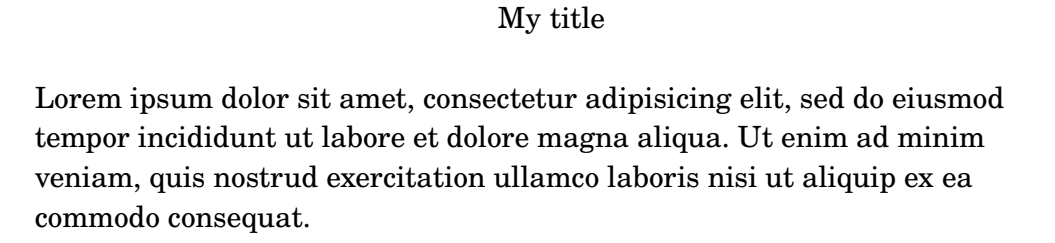
```
\markup {
  one
  \vcenter
  two
  three
}
```

one two three

`\wordwrap-field` *symbol* (symbol)
: Wordwrap the data which has been assigned to *symbol*.

```
\header {
  title = "My title"
  descr = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

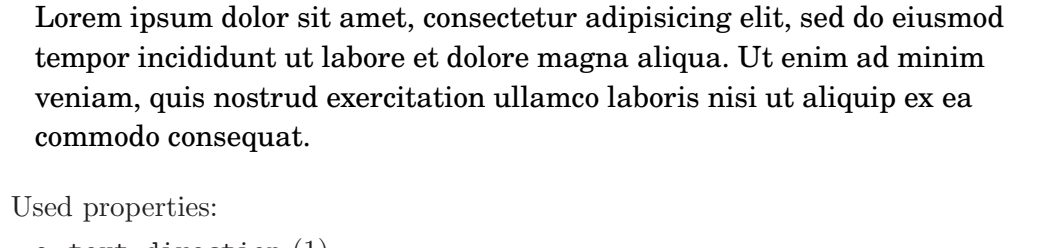
```
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat."
}

\paper {
  bookTitleMarkup = \markup {
    \column {
      \fill-line { \fromproperty #'header:title }
      \null
      \wordwrap-field #'header:descr
    }
  }
}

\markup {
  \null
}
```



`\wordwrap` *args* (list of markups)

Simple wordwrap. Use `\override #'(line-width . X)` to set the line width, where X is the number of staff spaces.

```
\markup {
  \wordwrap {
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  }
}
```



Used properties:

- `text-direction` (1)
- `word-space`
- `line-width` (#f)
- `baseline-skip`

`\wordwrap-string` *arg* (string)

Wordwrap a string. Paragraphs may be separated with double newlines.

```
\markup {
  \override #'(line-width . 40)
  \wordwrap-string #"Lorem ipsum dolor sit amet, consectetur
    adipisicing elit, sed do eiusmod tempor incididunt ut labore
    et dolore magna aliqua.


    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.


    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum"
}
```

Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore
et dolore magna aliqua.
Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo
consequat.
Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia
deserunt mollit anim id est laborum

Used properties:

- `text-direction` (1)
- `word-space`
- `line-width`
- `baseline-skip`

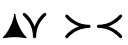
B.6.3 Graphic

`\arrow-head` *axis* (integer) *direction* (direction) *filled* (boolean)

Produce an arrow head in specified direction and axis. Use the filled head if *filled* is specified.

```
\markup {
  \fontsize #5 {
    \general-align #Y #DOWN {
      \arrow-head #Y #UP ##t
      \arrow-head #Y #DOWN ##f
      \hspace #2
      \arrow-head #X #RIGHT ##f
      \arrow-head #X #LEFT ##f
    }
  }
```

```
}
```



`\beam` *width* (number) *slope* (number) *thickness* (number)
Create a beam with the specified parameters.

```
\markup {
  \beam #5 #1 #2
}
```

`\bracket` *arg* (markup)
Draw vertical brackets around *arg*.

```
\markup {
  \bracket {
    \note #"2." #UP
  }
}
```

`\circle` *arg* (markup)
Draw a circle around *arg*. Use `thickness`, `circle-padding` and `font-size` properties to determine line thickness and padding around the markup.

```
\markup {
  \circle {
    Hi
  }
}
```

Used properties:

- `circle-padding` (0.2)
- `font-size` (0)
- `thickness` (1)

`\draw-circle` *radius* (number) *thickness* (number) *fill* (boolean)
A circle of radius *radius*, thickness *thickness* and optionally filled.

```
\markup {
  \draw-circle #2 #0.5 ##f
  \hspace #2
  \draw-circle #2 #0 ##t
}
```

`\draw-line` *dest* (pair of numbers)
A simple line.

```
\markup {
  \draw-line #'(4 . 4)
  \override #'(thickness . 5)
  \draw-line #'(-3 . 0)
}
```

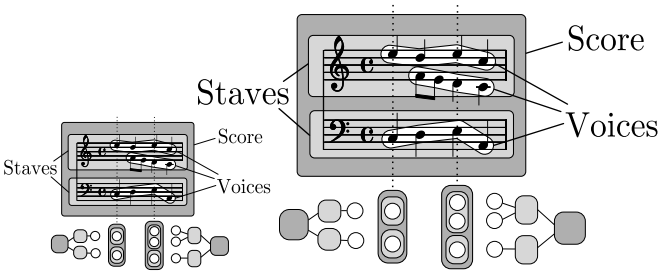
Used properties:

- `thickness` (1)

`\epsfile` *axis* (number) *size* (number) *file-name* (string)

Inline an EPS image. The image is scaled along *axis* to *size*.

```
\markup {
  \general-align #Y #DOWN {
    \epsfile #X #20 #"context-example.eps"
    \epsfile #Y #20 #"context-example.eps"
  }
}
```



`\filled-box` *xext* (pair of numbers) *yext* (pair of numbers) *blot* (number)

Draw a box with rounded corners of dimensions *xext* and *yext*. For example,

```
\filled-box #'(-.3 . 1.8) #'(-.3 . 1.8) #0
```

creates a box extending horizontally from -0.3 to 1.8 and vertically from -0.3 up to 1.8, with corners formed from a circle of diameter 0 (i.e. sharp corners).

```
\markup {
  \filled-box #'(0 . 4) #'(0 . 4) #0
  \filled-box #'(0 . 2) #'(-4 . 2) #0.4
  \filled-box #'(1 . 8) #'(0 . 7) #0.2
  \with-color #white
  \filled-box #'(-4.5 . -2.5) #'(3.5 . 5.5) #0.7
}
```

`\hbracket` *arg* (markup)

Draw horizontal brackets around *arg*.

```
\markup {
  \hbracket {
```

```
    \line {
      one two three
    }
  }
}
```

one two three

`\postscript` *str* (string)

This inserts *str* directly into the output as a PostScript command string. Due to technicalities of the output backends, different scales should be used for the TEX and PostScript backend, selected with `-f`.

For the TEX backend, the following string prints a rotated text

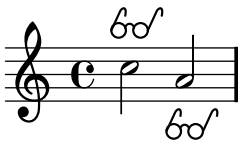
```
0 0 moveto /ecrm10 findfont
1.75 scalefont setfont 90 rotate (hello) show
```

The magical constant 1.75 scales from LilyPond units (staff spaces) to TEX dimensions.

For the postscript backend, use the following

```
gsave /ecrm10 findfont
 10.0 output-scale div
 scalefont setfont 90 rotate (hello) show grestore
```

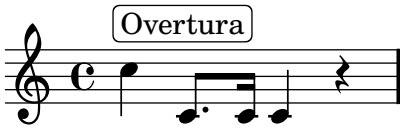
```
eyeglassesps = #"
  0.15 setlinewidth
  -0.9 0 translate
  1.1 1.1 scale
  1.2 0.7 moveto
  0.7 0.7 0.5 0 361 arc
  stroke
  2.20 0.70 0.50 0 361 arc
  stroke
  1.45 0.85 0.30 0 180 arc
  stroke
  0.20 0.70 moveto
  0.80 2.00 lineto
  0.92 2.26 1.30 2.40 1.15 1.70 curveto
  stroke
  2.70 0.70 moveto
  3.30 2.00 lineto
  3.42 2.26 3.80 2.40 3.65 1.70 curveto
  stroke"

eyeglasses = \markup {
  \with-dimensions #'(0 . 4.4) #'(0 . 2.5)
  \postscript #eyeglassesps
}

\relative c'' {
  c2^\eyeglasses
  a2_\eyeglasses
}
```



`\rounded-box` *arg* (markup)

Draw a box with rounded corners around *arg*. Looks at `thickness`, `box-padding` and `font-size` properties to determine line thickness and padding around the markup; the `corner-radius` property makes it possible to define another shape for the corners (default is 1).

```
c4^\markup {
  \rounded-box {
    Overtura
  }
}
c,8. c16 c4 r
```



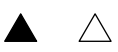
Used properties:

- `box-padding` (0.5)
- `font-size` (0)
- `corner-radius` (1)
- `thickness` (1)

`\triangle` *filled* (boolean)

A triangle, either filled or empty.

```
\markup {
  \triangle ##t
  \hspace #2
  \triangle ##f
}
```



Used properties:

- `baseline-skip` (2)
- `font-size` (0)
- `thickness` (0.1)

`\with-url` *url* (string) *arg* (markup)

Add a link to URL *url* around *arg*. This only works in the PDF backend.

```
\markup {
  \with-url #"http://lilypond.org/web/" {
    LilyPond ... \italic {
      music notation for everyone
    }
  }
}
```

LilyPond ... *music notation for everyone*

B.6.4 Music

`\doubleflat`
: Draw a double flat symbol.

```
\markup {
  \doubleflat
}
```

`\doublesharp`
: Draw a double sharp symbol.

```
\markup {
  \doublesharp
}
```

`\flat`
: Draw a flat symbol.

```
\markup {
  \flat
}
```

`\musicglyph` *glyph-name* (string)
: *glyph-name* is converted to a musical symbol; for example, `\musicglyph #"accidentals.natural"` selects the natural sign from the music font. See Section "La fonte Feta" dans *Manuel de notation* for a complete listing of the possible glyphs.

```
\markup {
  \musicglyph #"f"
  \musicglyph #"rests.2"
  \musicglyph #"clefs.G_change"
}
```

`\natural`
: Draw a natural symbol.

```
\markup {
  \natural
}
```

`\note-by-number` *log* (number) *dot-count* (number) *dir* (number)
: Construct a note symbol, with stem. By using fractional values for *dir*, you can obtain longer or shorter stems.

```
\markup {
  \note-by-number #3 #0 #DOWN
  \hspace #2
  \note-by-number #1 #2 #0.8
}
```

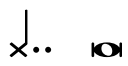
Used properties:

- `style` ('())
- `font-size` (0)

`\note` *duration* (string) *dir* (number)

This produces a note with a stem pointing in *dir* direction, with the *duration* for the note head type and augmentation dots. For example, `\note #"4." #-0.75` creates a dotted quarter note, with a shortened down stem.

```
\markup {
  \override #'(style . cross) {
    \note #"4.." #UP
  }
  \hspace #2
  \note #"breve" #0
}
```



Used properties:

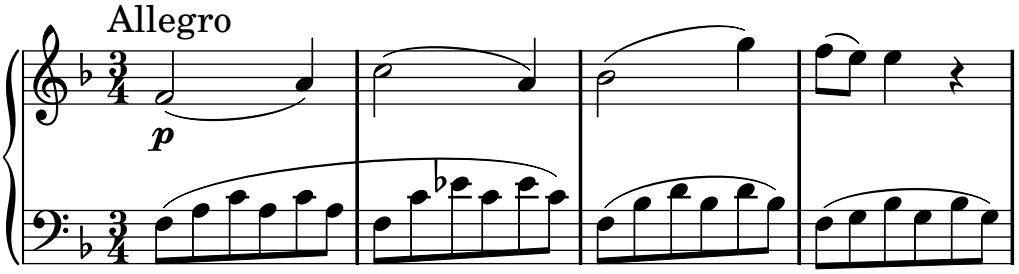
- `style` ('())
- `font-size` (0)

`\score` *score* (unknown)

Inline an image of music.

```
\markup {
  \score {
    \new PianoStaff <<
      \new Staff \relative c' {
        \key f \major
        \time 3/4
        \mark \markup { Allegro }
        f2\p( a4)
        c2( a4)
        bes2( g'4)
        f8( e) e4 r
      }
      \new Staff \relative c {
        \clef bass
        \key f \major
        \time 3/4
        f8( a c a c a
        f c' es c es c)
```

```
        f,( bes d bes d bes)
        f( g bes g bes g)
      }
    >>
    \layout {
      indent = 0.0\cm
      \context {
        \Score
        \override RehearsalMark #'break-align-symbols =
          #'(time-signature key-signature)
        \override RehearsalMark #'self-alignment-X = #LEFT
      }
      \context {
        \Staff
        \override TimeSignature #'break-align-anchor-alignment = #LEFT
      }
    }
  }
}
```



`\semiflat`

Draw a semiflat symbol.

```
\markup {
  \semiflat
}
```

`\semisharp`

Draw a semi sharp symbol.

```
\markup {
  \semisharp
}
```

`\sesquiflat`

Draw a 3/2 flat symbol.

```
\markup {
  \sesquiflat
}
```

`\sesquisharp`
: Draw a 3/2 sharp symbol.

```
\markup {
  \sesquisharp
}
```

`\sharp`
: Draw a sharp symbol.

```
\markup {
  \sharp
}
```

`\tied-lyric` *str* (string)
: Like simple-markup, but use tie characters for '`~`' tilde symbols.

```
\markup {
  \tied-lyric #"Lasciate~i monti"
}
```

Lasciate i monti

B.6.5 Instrument Specific Markup

`\fret-diagram` *definition-string* (string)
: Make a (guitar) fret diagram. For example, say

`\markup \fret-diagram #"s:0.75;6-x;5-x;4-o;3-2;2-3;1-2;"`

for fret spacing 3/4 of staff space, D chord diagram

Syntax rules for *definition-string*:

- Diagram items are separated by semicolons.
- Possible items:
  - `s:`*number* – Set the fret spacing of the diagram (in staff spaces). Default: 1.
  - `t:`*number* – Set the line thickness (in staff spaces). Default: 0.05.
  - `h:`*number* – Set the height of the diagram in frets. Default: 4.
  - `w:`*number* – Set the width of the diagram in strings. Default: 6.
  - `f:`*number* – Set fingering label type (0 = none, 1 = in circle on string, 2 = below string). Default: 0.
  - `d:`*number* – Set radius of dot, in terms of fret spacing. Default: 0.25.
  - `p:`*number* – Set the position of the dot in the fret space. 0.5 is centered; 1 is on lower fret bar, 0 is on upper fret bar. Default: 0.6.
  - `c:`*string1*`-`*string2*`-`*fret* – Include a barre mark from *string1* to *string2* on *fret*.
  - *string*`-`*fret* – Place a dot on *string* at *fret*. If *fret* is '`o`', *string* is identified as open. If *fret* is '`x`', *string* is identified as muted.

- *string-fret-fingering* – Place a dot on *string* at *fret*, and label with *fingering* as defined by the `f:` code.

– Note: There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-terse` *definition-string* (string)

Make a fret diagram markup using terse string-based syntax.

Here is an example

```
\markup \fret-diagram-terse #"x;x;o;2;3;2;"
```

for a D chord diagram.

Syntax rules for *definition-string*:

- Strings are terminated by semicolons; the number of semicolons is the number of strings in the diagram.
- Mute strings are indicated by '`x`'.
- Open strings are indicated by '`o`'.
- A number indicates a fret indication at that fret.
- If there are multiple fret indicators desired on a string, they should be separated by spaces.
- Fingerings are given by following the fret number with a `-`, followed by the finger indicator, e.g. '`3-2`' for playing the third fret with the second finger.
- Where a barre indicator is desired, follow the fret (or fingering) symbol with `-(` to start a barre and `-)` to end the barre.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\fret-diagram-verbose` *marking-list* (list)

Make a fret diagram containing the symbols indicated in *marking-list*.

For example,

```
\markup \fret-diagram-verbose
  #'((mute 6) (mute 5) (open 4)
     (place-fret 3 2) (place-fret 2 3) (place-fret 1 2))
```

produces a standard D chord diagram without fingering indications.

Possible elements in *marking-list*:

`(mute` *`string-number`*`)`

Place a small 'x' at the top of string *string-number*.

`(open` *`string-number`*`)`

Place a small 'o' at the top of string *string-number*.

`(barre` *`start-string end-string fret-number`*`)`

Place a barre indicator (much like a tie) from string *start-string* to string *end-string* at fret *fret-number*.

`(place-fret` *`string-number fret-number finger-value`*`)`

Place a fret playing indication on string *string-number* at fret *fret-number* with an optional fingering label *finger-value*. By default, the fret playing indicator is a solid dot. This can be changed by setting the value of the variable *dot-color*. If the *finger* part of the `place-fret` element is present, *finger-value* will be displayed according to the setting of the variable *finger-code*. There is no limit to the number of fret indications per string.

Used properties:

- `thickness` (0.5)
- `fret-diagram-details`
- `size` (1.0)
- `align-dir` (-0.4)

`\harp-pedal` *definition-string* (string)

Make a harp pedal diagram.

Possible elements in *definition-string*:

`^`	pedal is up
`-`	pedal is neutral
`v`	pedal is down
`|`	vertical divider line
`o`	the following pedal should be circled (indicating a change)

The function also checks if the string has the typical form of three pedals, then the divider and then the remaining four pedals. If not it prints out a warning. However, in any case, it will also print each symbol in the order as given. This means you can place the divider (even multiple dividers) anywhere you want, but you'll have to live with the warnings.

The appearance of the diagram can be tweaked inter alia using the size property of the TextScript grob (`\override Voice.TextScript #'size = #0.3`) for the overall, the thickness property (`\override Voice.TextScript #'thickness = #3`) for the line thickness of the horizontal line and the divider. The remaining configuration (box sizes, offsets and spaces) is done by the harp-pedal-details list of properties (`\override Voice.TextScript #'harp-pedal-details #'box-width = #1`). It contains the following settings: `box-offset` (vertical shift of the box center for up/down pedals), `box-width`, `box-height`, `space-before-divider` (the spacing between two boxes before the divider) and `space-after-divider` (box spacing after the divider).

```
\markup \harp-pedal #"^-v|--ov^"
```

Used properties:

- `thickness` (0.5)
- `harp-pedal-details`
- `size` (1.0)

B.6.6 Other

`\backslashed-digit` *num* (integer)
A feta number, with backslash. This is for use in the context of figured bass notation.

```
\markup {
  \backslashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \backslashed-digit #7
}
```

Used properties:

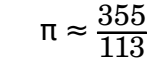
- `thickness` (1.6)
- `font-size` (0)

`\char` *num* (integer)
Produce a single character. For example, `\char #65` produces the letter 'A'.

```
\markup {
  \char #65
}
```

`\fraction` *arg1* (markup) *arg2* (markup)
Make a fraction of two markups.

```
\markup {

  \fraction 355 113
}
```



Used properties:

- `font-size` (0)

`\fromproperty` *symbol* (symbol)
Read the *symbol* from property settings, and produce a stencil from the markup contained within. If *symbol* is not defined, it returns an empty markup.

```
\header {
  myTitle = "myTitle"
  title = \markup {
    from
    \italic
    \fromproperty #'header:myTitle
  }
}
\markup {
  \null
}
```

from *myTitle*

`\lookup` *glyph-name* (string)
: Lookup a glyph by name.

```
\markup {
  \override #'(font-encoding . fetaBraces) {
    \lookup #"brace200"
    \hspace #2
    \rotate #180
    \lookup #"brace180"
  }
}
```

`\markalphabet` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z and continue with double letters.

```
\markup {
  \markalphabet #8
  \hspace #2
  \markalphabet #26
}
```

I AA

`\markletter` *num* (integer)
: Make a markup letter for *num*. The letters start with A to Z (skipping letter I), and continue with double letters.

```
\markup {
  \markletter #8
  \hspace #2
  \markletter #26
}
```

J AB

`\null`
: An empty markup with extents of a single point.

```
\markup {
  \null
}
```

`\on-the-fly` *procedure* (symbol) *arg* (markup)
: Apply the *procedure* markup command to *arg*. *procedure* should take a single argument.

`\override` *new-prop* (pair) *arg* (markup)
: Add the first argument in to the property list. Properties may be any sort of property supported by Section “font-interface” dans *Référence des propriétés internes* and Section “text-interface” dans *Référence des propriétés internes*, for example

```
\override #'(font-family . married) "bla"
```

```
\markup {
  \line {
    \column {
      default
      baseline-skip
    }
    \hspace #2
    \override #'(baseline-skip . 4) {
      \column {
        increased
        baseline-skip
      }
    }
  }
}
```

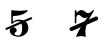
default baseline-skip

increased baseline-skip

`\page-ref` *label* (symbol) *gauge* (markup) *default* (markup)
: Reference to a page number. *label* is the label set on the referenced page (using the `\label` command), *gauge* a markup used to estimate the maximum width of the page number, and *default* the value to display when *label* is not found.

`\slashed-digit` *num* (integer)
: A feta number, with slash. This is for use in the context of figured bass notation.

```
\markup {
  \slashed-digit #5
  \hspace #2
  \override #'(thickness . 3)
  \slashed-digit #7
}
```



Used properties:

- `thickness` (1.6)
- `font-size` (0)

`\stencil` *stil* (unknown)
: Use a stencil as markup.

```
\markup {
  \stencil #(make-circle-stencil 2 0 #t)
}
```

`\strut`

Create a box of the same height as the space in the current font.

`\transparent` *arg* (markup)

Make the argument transparent.

```
\markup {
  \transparent {
    invisible text
  }
}
```

`\verbatim-file` *name* (string)

Read the contents of a file, and include it verbatim.

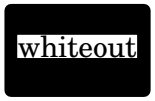
```
\markup {
  \verbatim-file #"simple.ly"
}
```

```
%% A simple piece in LilyPond, a scale.
\relative c' {
  c d e f g a b c
}
%% Optional helper for automatic updating by convert-ly.  May be omitte
\version "2.11.51"
```

`\whiteout` *arg* (markup)

Provide a white background for *arg*.

```
\markup {
  \combine
    \filled-box #'(-1 . 10) #'(-3 . 4) #1
    \whiteout whiteout
}
```



`\with-color` *color* (list) *arg* (markup)

Draw *arg* in color specified by *color*.

```
\markup {
  \with-color #red
  red
  \hspace #2
  \with-color #green
  green
  \hspace #2
  \with-color #blue
  blue
}
```

red green blue

`\with-dimensions` *x* (pair of numbers) *y* (pair of numbers) *arg* (markup)
: Set the dimensions of *arg* to *x* and *y*.

B.7 Text markup list commands

The following commands can all be used with `\markuplines`.

`\column-lines` *args* (list of markups)
: Like `\column`, but return a list of lines instead of a single markup. `baseline-skip` determines the space between each markup in *args*.

  Used properties:

  - `baseline-skip`

`\justified-lines` *args* (list of markups)
: Like `\justify`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width; *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

`\override-lines` *new-prop* (pair) *args* (list of markups)
: Like `\override`, for markup lists.

`\wordwrap-internal` *justify* (boolean) *args* (list of markups)
: Internal markup list command used to define `\justify` and `\wordwrap`.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)

`\wordwrap-lines` *args* (list of markups)
: Like `\wordwrap`, but return a list of lines instead of a single markup. Use `\override-lines #'(line-width . X)` to set the line width, where *X* is the number of staff spaces.

  Used properties:

  - `text-direction` (1)
  - `word-space`
  - `line-width` (#f)
  - `baseline-skip`

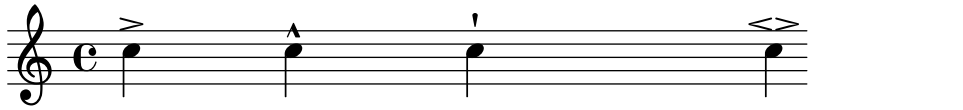
`\wordwrap-string-internal` *justify* (boolean) *arg* (string)
: Internal markup list command used to define `\justify-string` and `\wordwrap-string`.

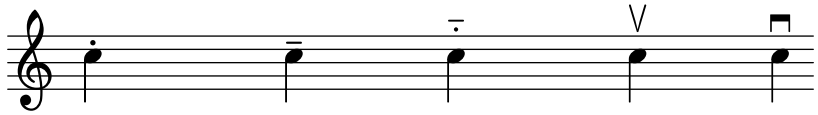
  Used properties:

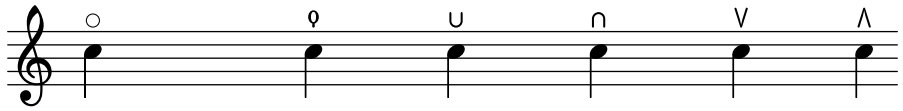
  - `text-direction` (1)
  - `word-space`
  - `line-width`

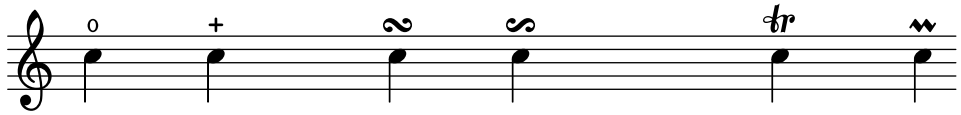
B.8 Liste des signes d'articulation

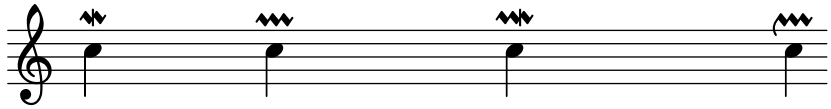
Voici la liste exhaustive des symboles :

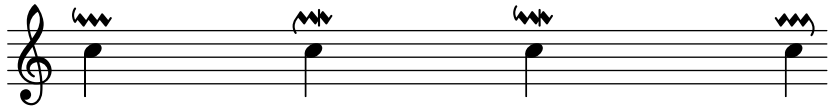


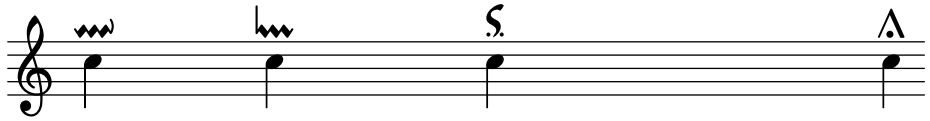


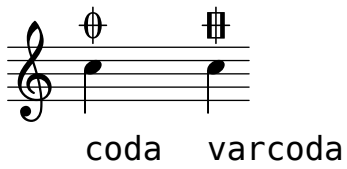












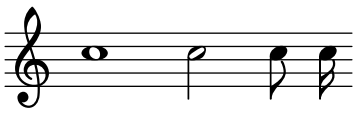
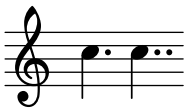
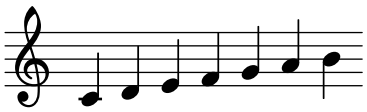
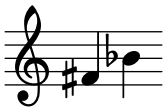
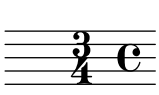
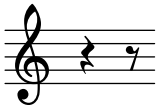
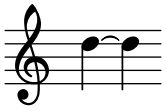
B.9 Liste des propriétés de contexte

B.10 Propriétés de mise en forme

B.11 Variables

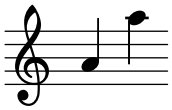
B.12 Fonctions Scheme

Annexe C Aide-mémoire

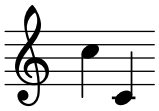
Syntaxe	Description	Exemple
`1 2 8 16`	valeurs rythmiques	
`c4. c4..`	notes pointées	
`c d e f g a b`	gamme	
`fis bes`	altérations	
`\clef treble \clef bass`	clés	
`\time 3/4 \time 4/4`	chiffre de mesure	
`r4 r8`	silences	
`d ~ d`	liaison de tenue	

`\key es \major`	armure

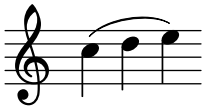
note`'`	monter d'une octave



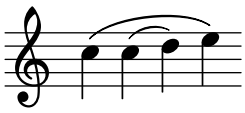
note`,`	baisser d'une octave



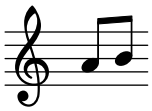
`c( d e)`	liaisons



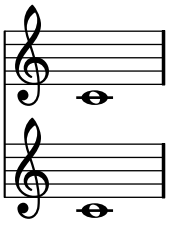
`c\( c( d) e\)`	liaisons de phrasé



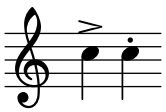
`a8[ b]`	ligatures



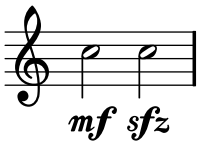
`<< \new Staff ... >>`	ajouter des portées



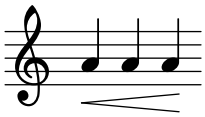
`c-> c-.`	indications d'articulation



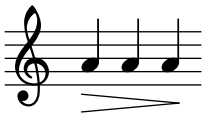
`c\mf c\sfz` — nuances



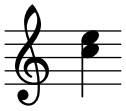
`a\< a a\!` — crescendo



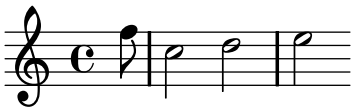
`a\> a a\!` — decrescendo



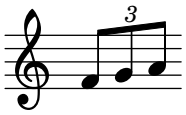
`< >` — accords



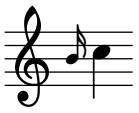
`\partial 8` — levées



`\times 2/3 {f g a}` — triolets



`\grace` — appogiatures

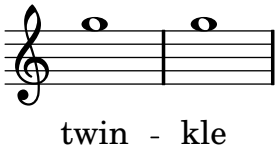


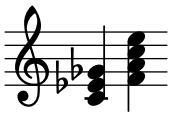
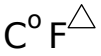
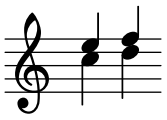
`\lyricmode { twinkle }` — ajouter des paroles — twinkle

`\new Lyrics` — imprimer les paroles

twinkle

`twin -- kle` — diviser un mot en plusieurs syllabes



`\chordmode { c:dim f:maj7 }`	accords chiffrés	
`\context ChordNames`	imprimer les chiffrages d'accords	
`<<{e f} \\ {c d}>>`	polyphonie	
`s4 s8 s16`	silences invisibles	

Annexe D 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.

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 E Index des commandes LilyPond

|

~

A

B

C

D

F

M

N

P

R

S

T

W

Annexe F Index de LilyPond

|

~

1

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W