‘`+.ly`’

Introduction

This document presents proofs for LilyPond 2.10.1. When the text corresponds with the shown notation, we consider LilyPond Officially BugFree (tm). This document is intended for finding bugs and for documenting bugfixes.

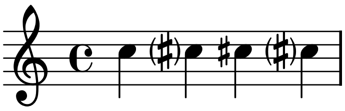
In the web version of this document, you can click on the file name or figure for each example to see the corresponding input file.

TODO: order of tests (file names!), test only one feature per test. Smaller and neater tests.

(left blank intentionally)

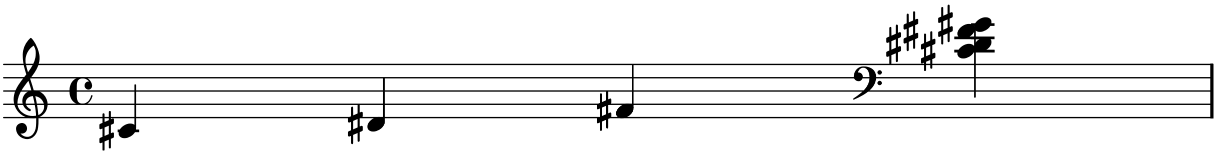
‘`accidental-cautionary.ly`’

Cautionary accidentals are indicated using either parentheses (default) or smaller accidentals.



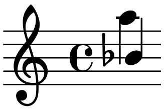
‘`accidental-clef-change.ly`’

Accidentals are reset for clef changes.



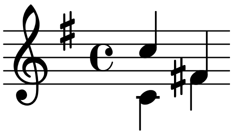
‘`accidental-collision.ly`’

accidentals avoid stems of other notes too.



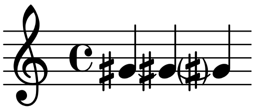
‘`accidental-double.ly`’

If two forced accidentals happen at the same time, only one sharp sign is printed.



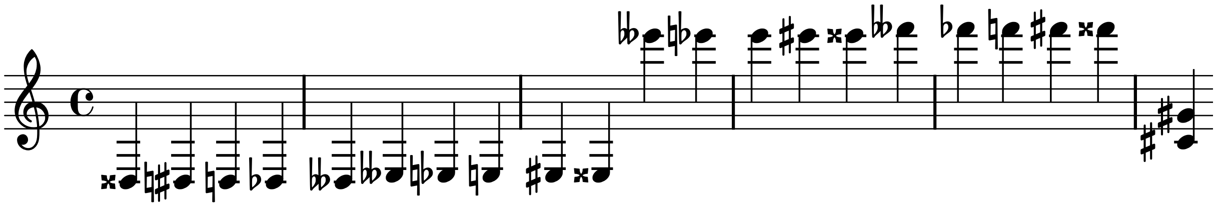
‘`accidental-forced-tie.ly`’

Accidentals can be forced with ! and ? even if the notes are tied.



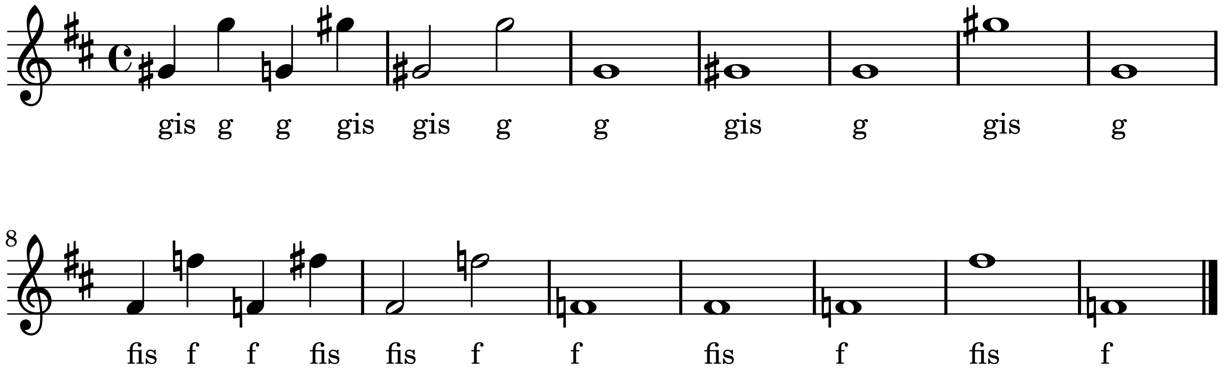
`accidental-ledger.ly`

Ledger lines are shortened when there are accidentals. This happens only for the single ledger line close to the note head, and only if the accidental is horizontally close to the head.



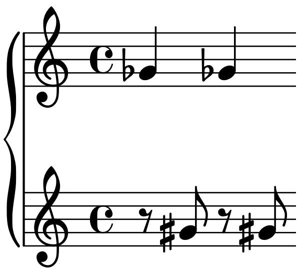
`accidental-octave.ly`

This shows how accidentals in different octaves are handled. The note names are also automatically printed but the octavation has been dropped out.


gis g g gis gis g g gis g gis g
8
fis f f fis fis f f fis f fis f

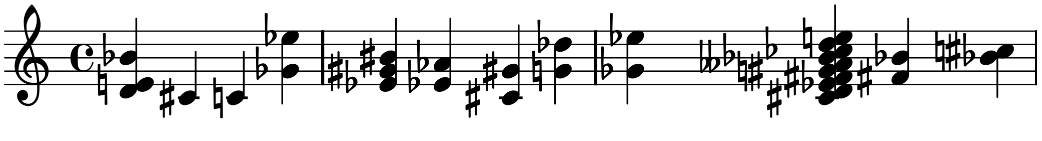
`accidental-piano.ly`

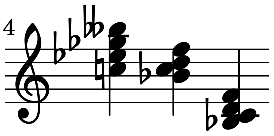
In piano accidental style, notes in both staves influence each other. In this example, each note should have an accidental.



`accidental-placement.ly`

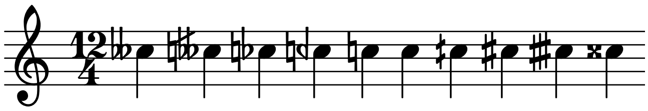
Accidentals are placed as closely as possible. Accidentals in corresponding octaves are aligned. The top accidental should be nearest to the chord. The flats in a sixth should be staggered.





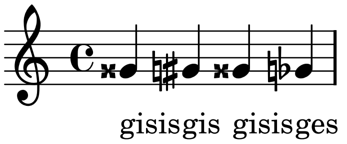
‘`accidental-quarter.ly`’

Quarter tone notation is supported, including threequarters flat.



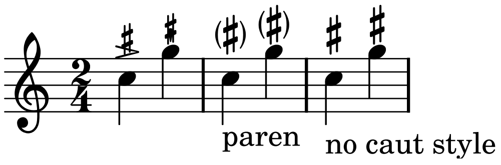
‘`accidental-single-double.ly`’

A sharp sign after a double sharp sign, as well as a flat sign after a double flat sign is automatically prepended with a natural sign.



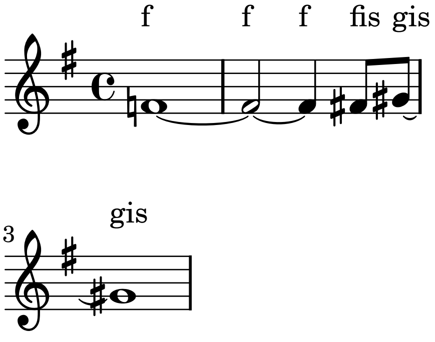
‘`accidental-suggestions.ly`’

setting the `suggestAccidentals` will print accidentals vertically relative to the note. This is useful for denoting Musica Ficta.



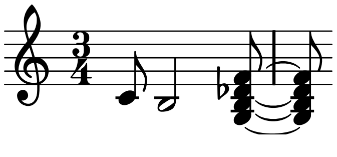
‘`accidental-tie.ly`’

The second and third notes should not get accidentals, 6 because they are tied to a note. However, an accidental is present if the line is broken at the tie, which happens for the G sharp.



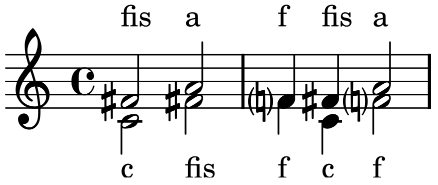
‘`accidental-unbroken-tie-spacing.ly`’

Tied accidentaled notes (which cause reminder accidentals) do not wreak havoc in the spacing when unbroken.



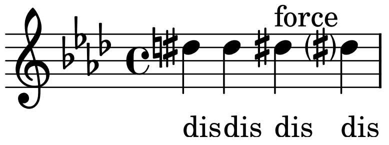
‘`accidental-voice.ly`’

This shows how modern cross voice auto cautionary accidentals are handled. The first two fisses get accidentals because they belong to different voices. The first f gets cautionary natural because of previous measure. The last f gets cautionary natural because fis was only in the other voice.



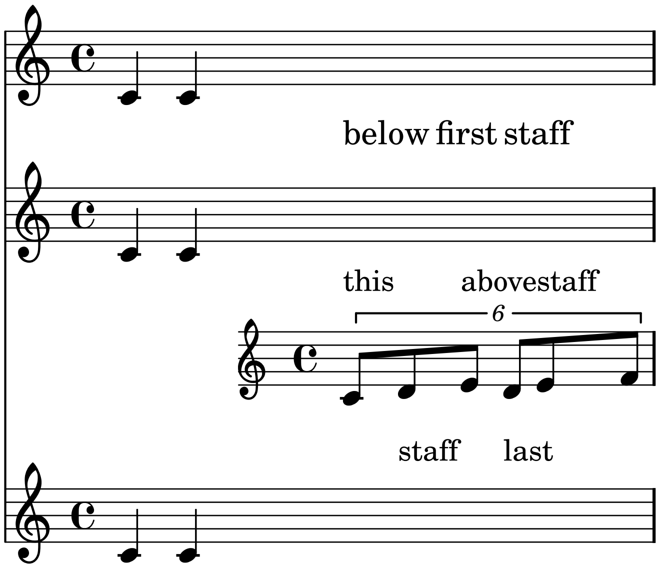
‘`accidental.ly`’

Accidentals work: the second note does not get a sharp. The third and fourth show forced and courtesy accidentals.



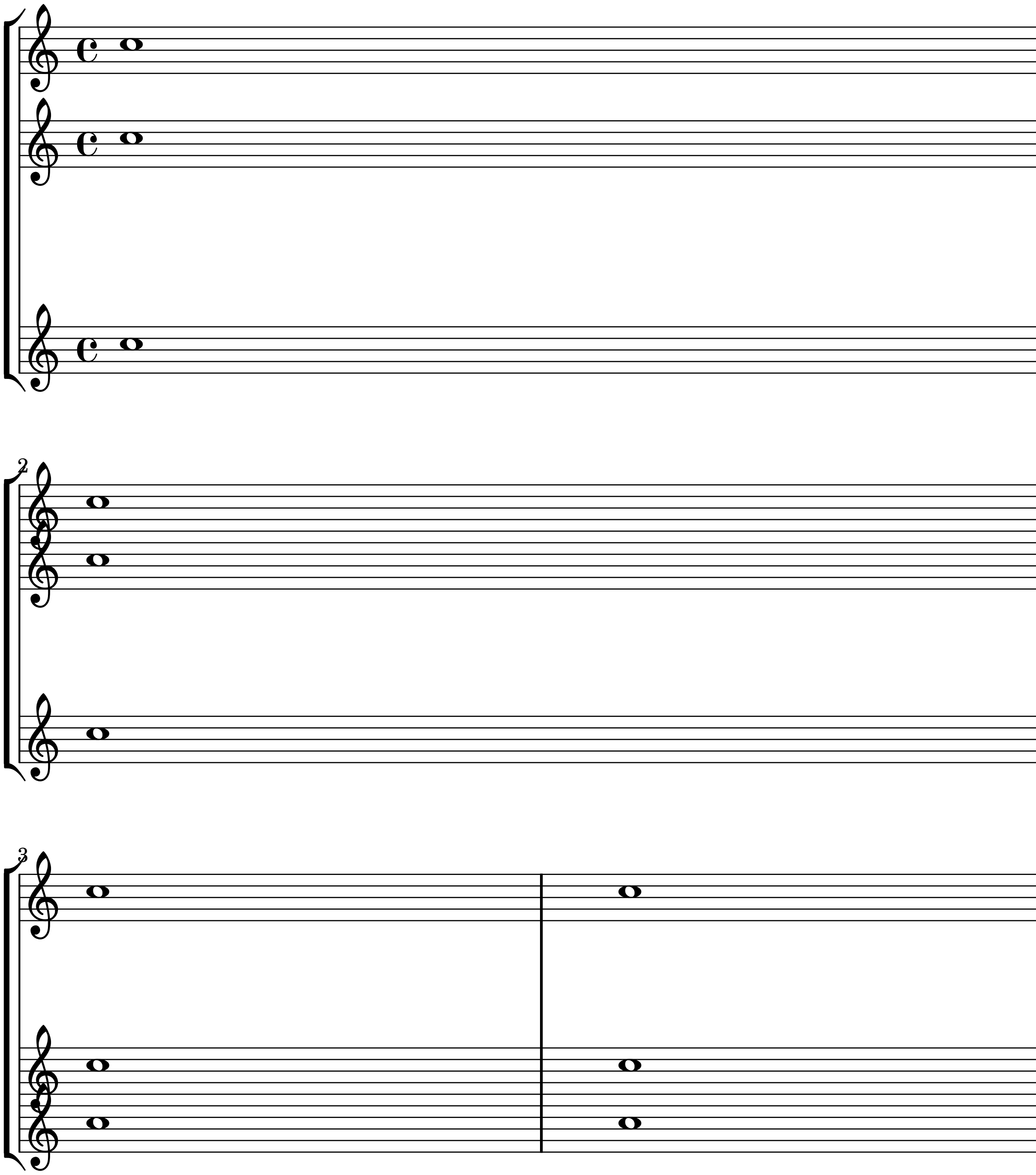
‘`alignment-order.ly`’

Newly created contexts can be inserted anywhere in the vertical alignment.



‘`alignment-vertical-manual-setting.ly`’

Alignments may be changed pre system by setting `alignment-offsets` in the `line-break-system-details` property

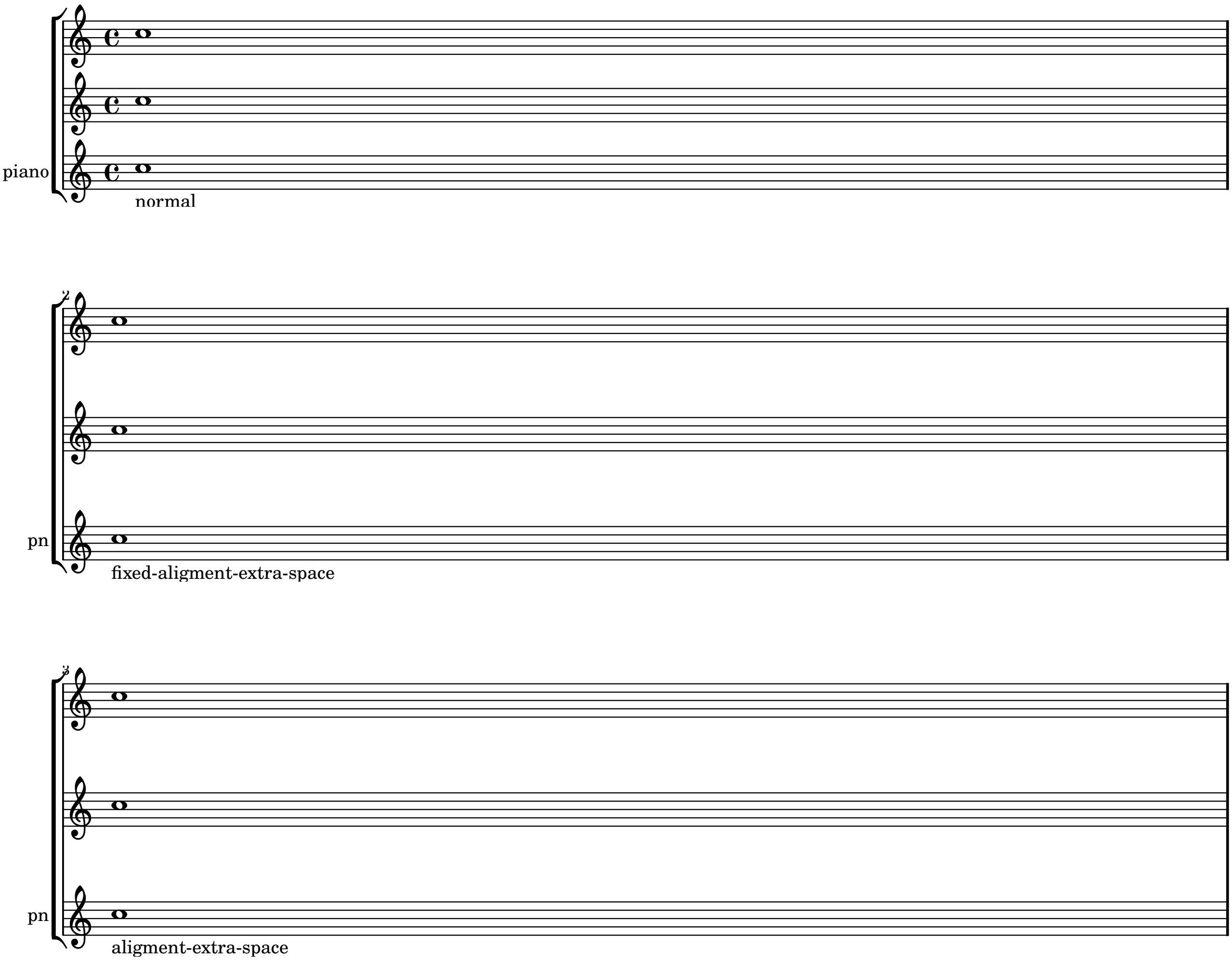

2
3

‘`alignment-vertical-spacing.ly`’

By setting properties in `NonMusicalPaperColumn`, vertical spacing of alignments can be adjusted per system.

By setting `alignment-extra-space` or `fixed-alignment-extra-space` an individual system may be stretched vertically.

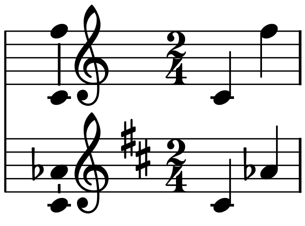
For technical reasons, `overrideProperty` has to be used for setting properties on individual object. `\override` in a `\context` block may still be used for global overrides.



‘`ambitus.ly`’

Ambituses indicate pitch ranges for voices.

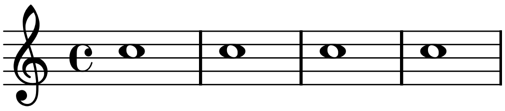
Accidentals only show up if they’re not part of key signature. `AmbitusNoteHead` grobs also have ledger lines.



'`apply-context.ly`'

With `\applyContext`, `\properties` can be modified procedurally. Applications include: checking bar numbers, smart octavation.

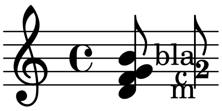
This example prints a bar-number during processing on stdout.



'`apply-output.ly`'

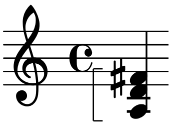
The `\applyOutput` expression is the most flexible way to tune properties for individual grobs.

Here, the layout of a note head is changed depending on its vertical position.



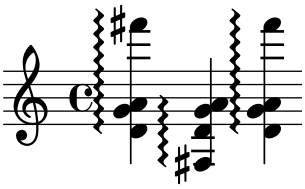
'`arpeggio-bracket.ly`'

A square bracket on the left indicates that the player should not arpeggiate the chord.



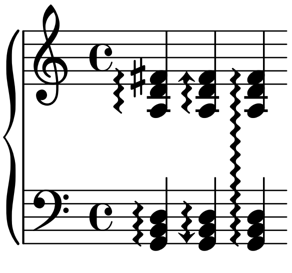
'`arpeggio-collision.ly`'

Arpeggio stays clear of accidentals and flipped note heads.



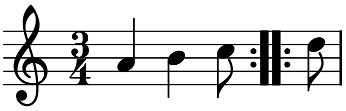
'`arpeggio.ly`'

Arpeggios are supported, both cross-staff and broken single staff.



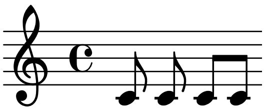
`‘auto-beam-bar.ly’`

No auto beams will be put over (manual) repeat bars.



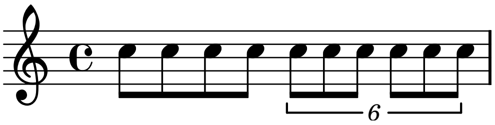
`‘auto-beam-no-beam.ly’`

The autobeamer may be switched off for a single note with `\noBeam`.



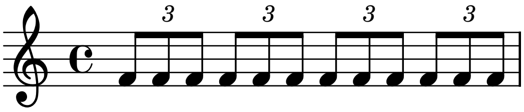
`‘auto-beam-triplet.ly’`

Automatic beaming is also done on tuplets.



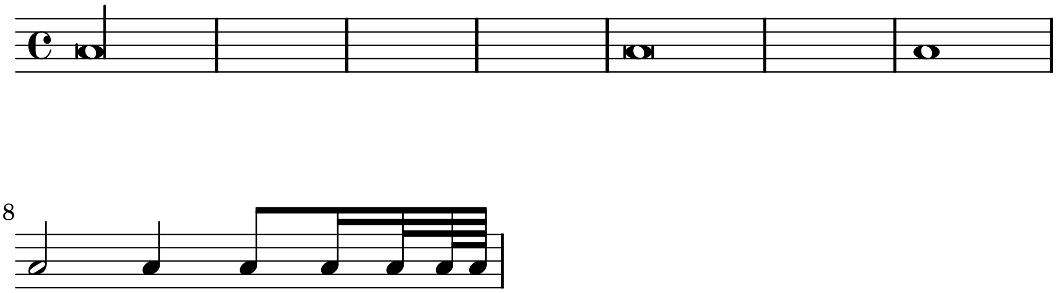
`‘auto-beam-tuplets.ly’`

Tuplet-spanner should not put (visible) brackets on beams even if they’re auto generated.



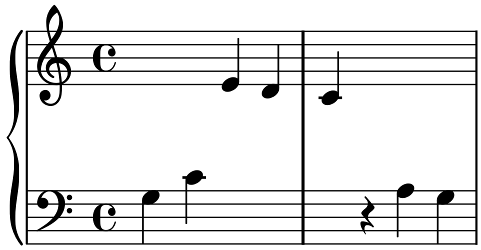
`‘auto-beam.ly’`

Beams are place automatically; the last measure should have a single beam.



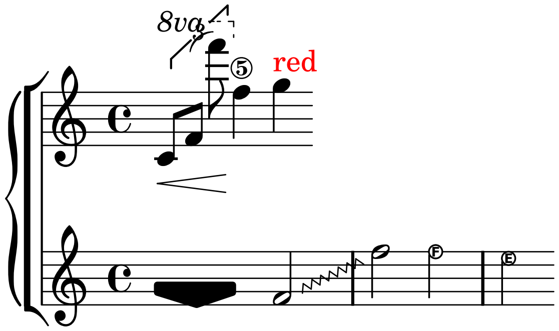
‘`auto-change.ly`’

Auto change piano staff switches voices between up and down staves automatically rests are switched along with the coming note. When central C is reached, staff is not yet switched (by default).



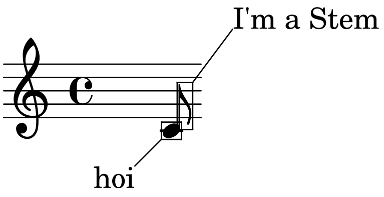
‘`backend-excercise.ly`’

Excercise all output functions



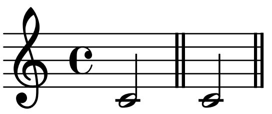
‘`balloon.ly`’

With balloon texts, objects in the output can be marked, with lines and explanatory text added.



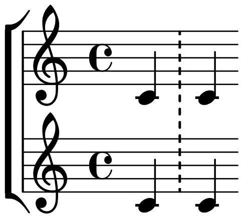
‘`bar-check-redefine.ly`’

The meaning of `|` is stored in the identifier `pipeSymbol`.



‘`bar-line-dashed.ly`’

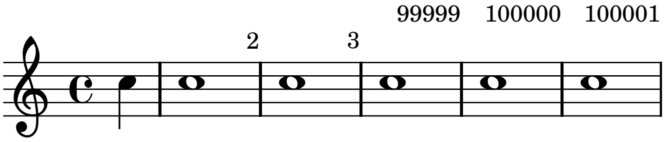
The dashes in a dashed bar line covers staff lines exactly. Dashed barlines between staves start and end on a half dash precisely.



‘`bar-number.ly`’

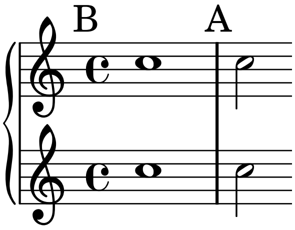
Bar number may be set and their padding adjusted individually. The counting of bar numbers is started after the anacrusis.

To prevent clashes at the beginning of a line, the padding may have to be increased.



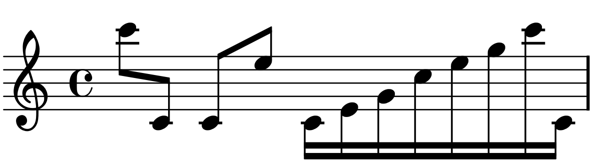
‘`bar-scripts.ly`’

Markings can be attached to (invisible) barlines.



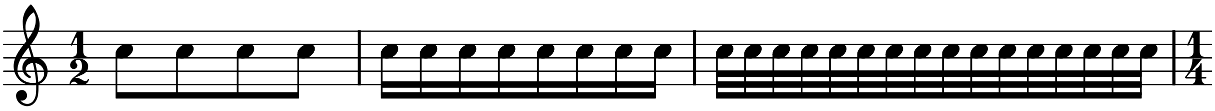
‘`beam-auto-knee.ly`’

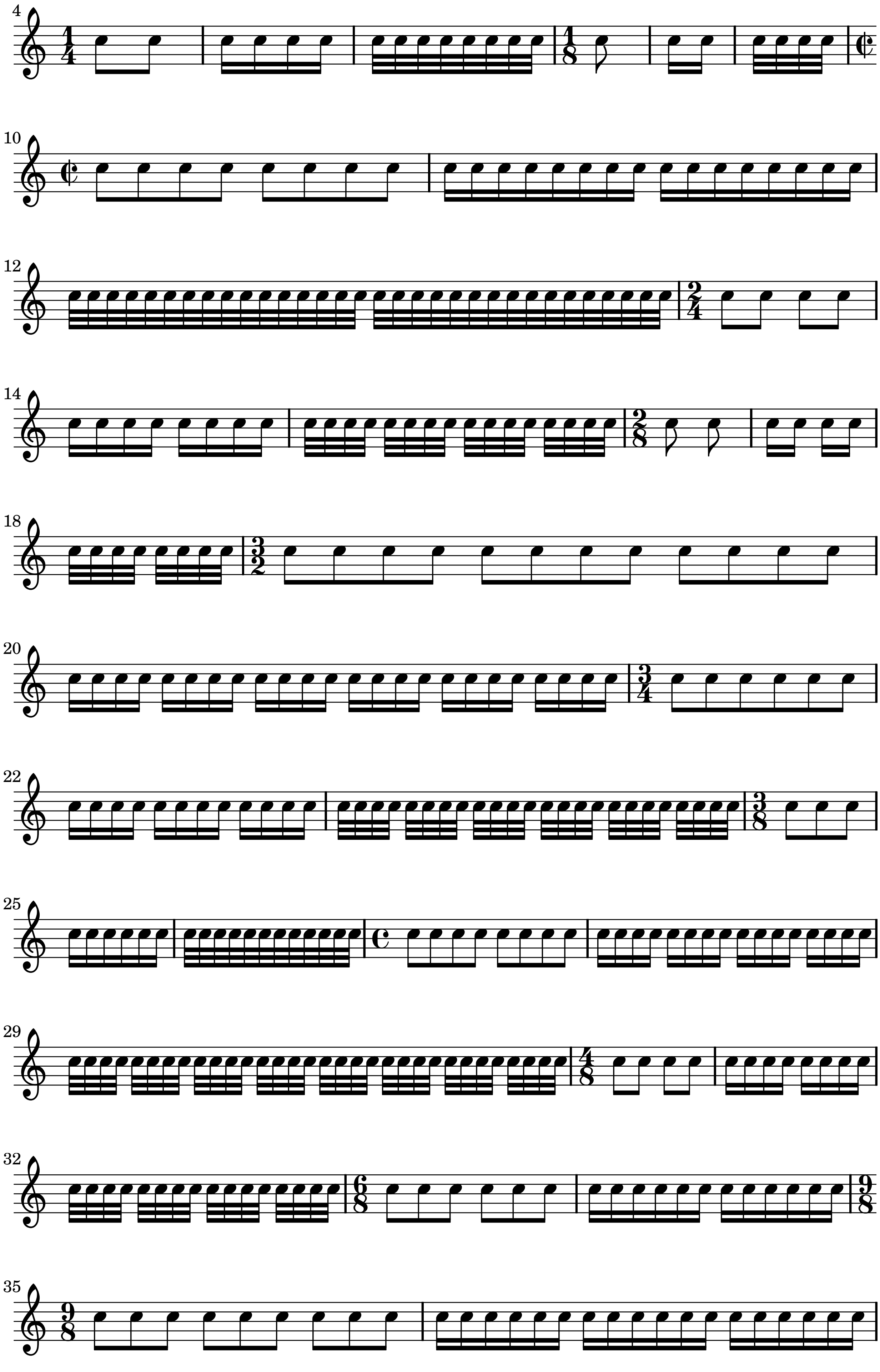
A knee is made automatically when a horizontal beam fits in a gap between note heads that is larger than a predefined threshold.



‘`beam-auto.ly`’

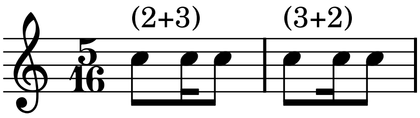
There are presets for the `auto-beam` engraver in the case of common time signatures.




4
10
12
14
18
20
22
25
29
32
35

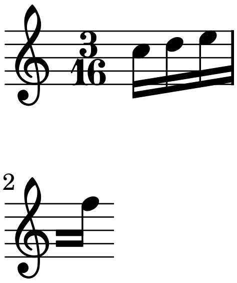
‘`beam-beat-grouping.ly`’

Beaming patterns obey the `beatGrouping` property.



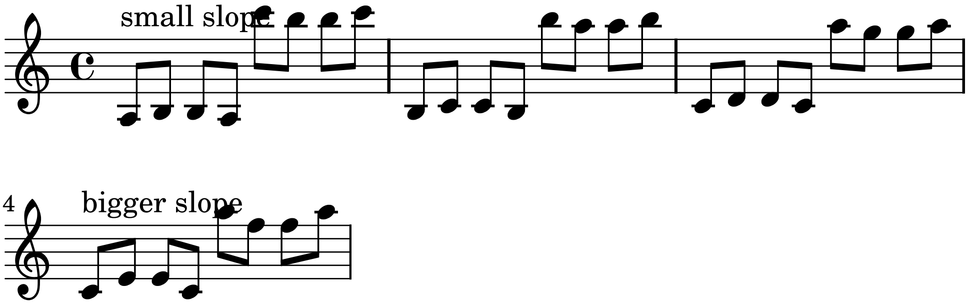
‘`beam-break.ly`’

Beams can be printed across line breaks, if forced.



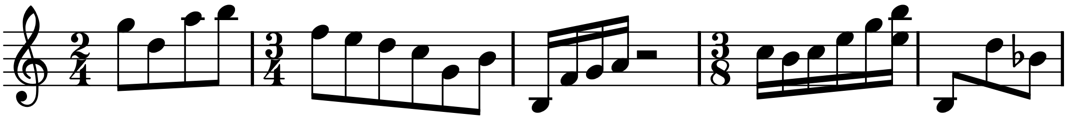
‘`beam-center-slope.ly`’

Simple beams on middle staffline are allowed to be slightly sloped, even if the notes have ledgers. Beams reaching beyond middle line can have bigger slope.



‘`beam-concave-damped.ly`’

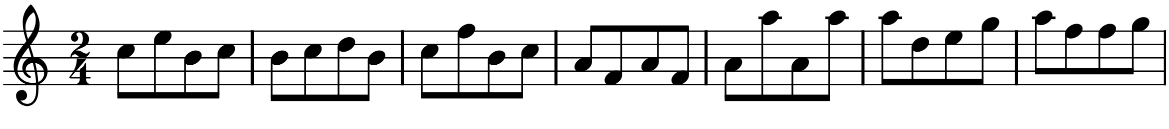
Beams that are not strictly concave are damped according to their concaveness.

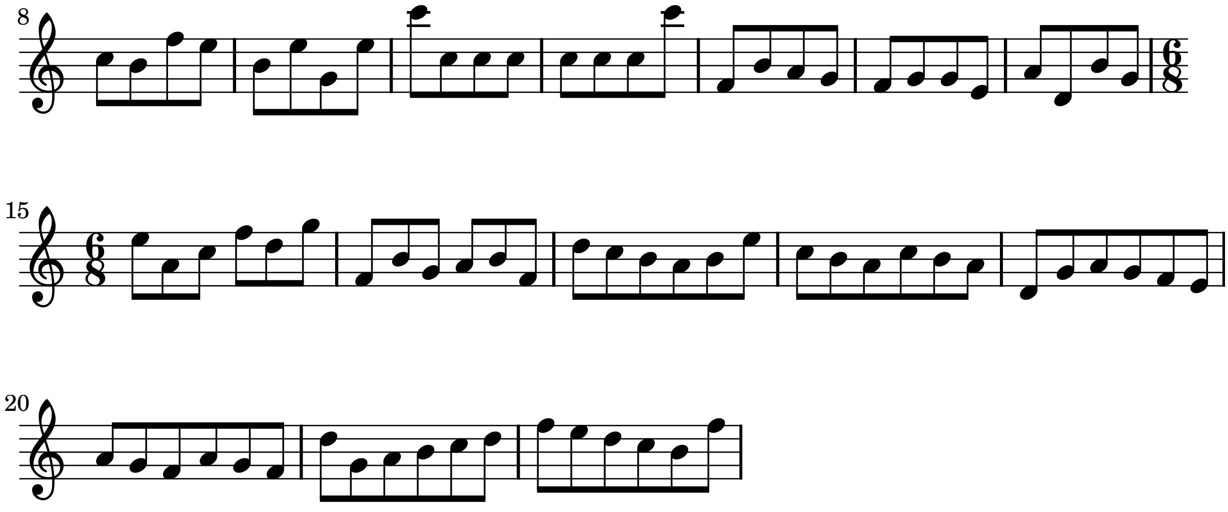


‘`beam-concave.ly`’

Fully concave beams should be horizontal. Informally spoken, concave refers to the shape of the notes that are opposite a beam. If an up-beam has high notes on its center stems, then we call it concave.

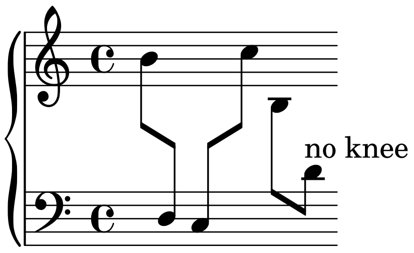
If a beam is fails a test, the desired slope is printed next to it.





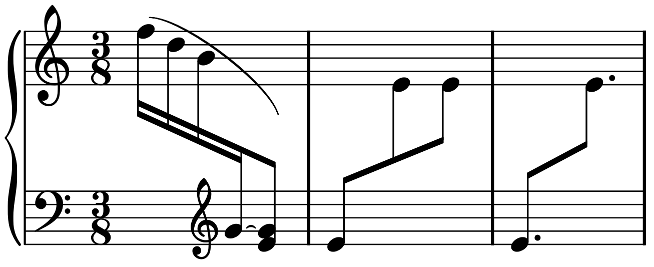
‘`beam-cross-staff-auto-knee.ly`’

Automatic cross-staff knees work also (here they were produced with explicit staff switches).



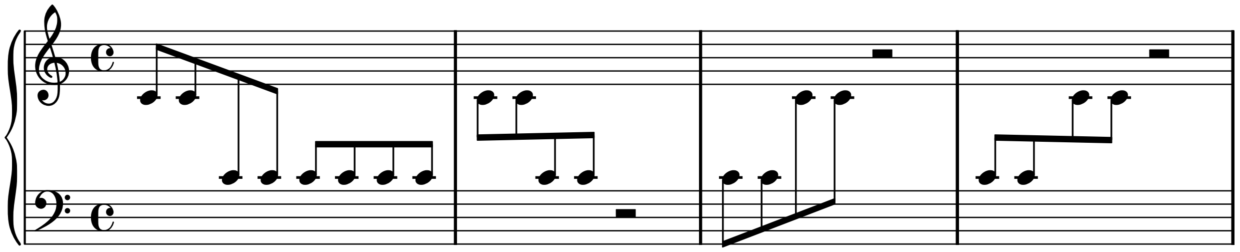
‘`beam-cross-staff-slope.ly`’

Cross staff (kneed) beams do not cause extreme slopes.



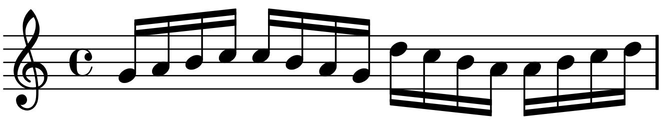
‘`beam-cross-staff.ly`’

Beams can be typeset over fixed distance aligned staves, beam beautification does not really work, but knees do. Beams should be behave well, wherever the switching point is.



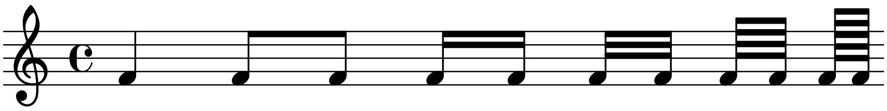
'`beam-damp.ly`'

Beams are less steep than the notes they encompass.



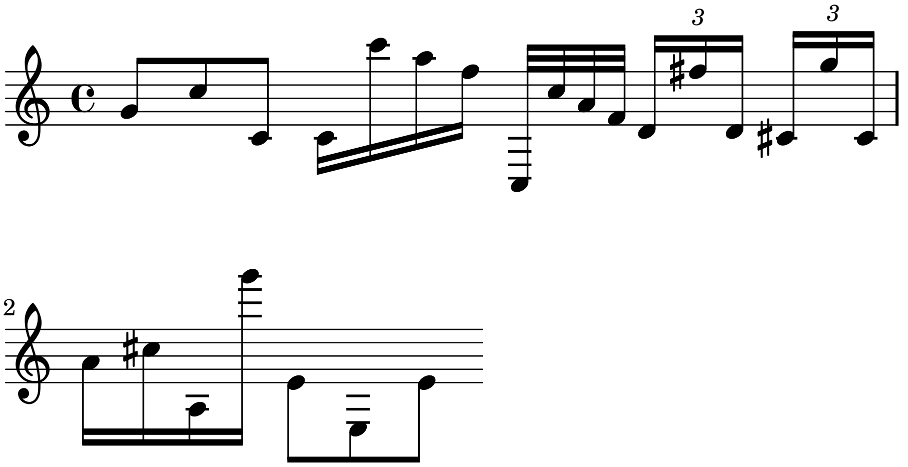
'`beam-default-lengths.ly`'

Beamed stems have standard lengths if possible. Quantization is switched off in this example.



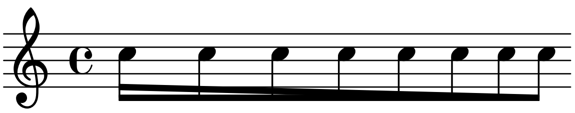
'`beam-extreme.ly`'

Beams should behave reasonably well, even under extreme circumstances. Stems may be short, but noteheads should never touch the beam. Note that under normal circumstances, these beams would get knees here `Beam.auto-knee-gap` was set to false.



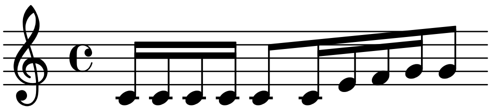
'`beam-feather.ly`'

Specifying `grow-direction` on a beam, will cause feathered beaming. The `\featherDurations` function can be used to adjust note durations.



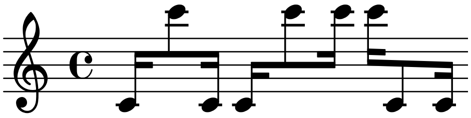
'`beam-french.ly`'

In french style beaming, the stems do not go between beams.



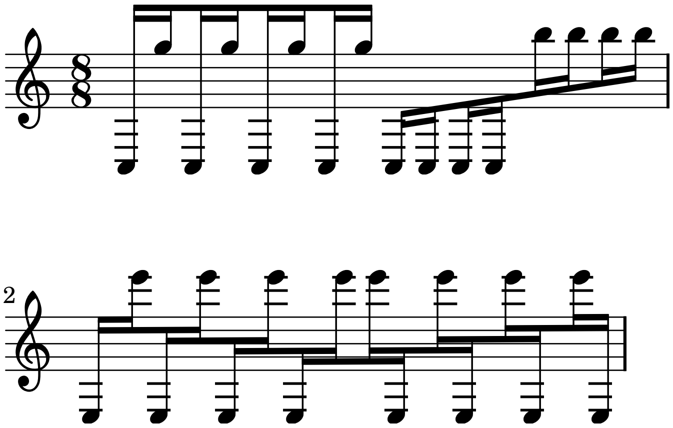
‘`beam-funky-beamlet.ly`’

Funky kneed beams with beamlets also work. The beamlets should be pointing to the note head.



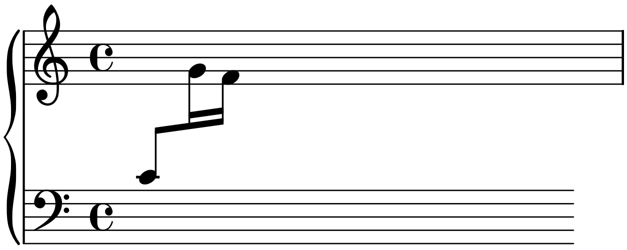
‘`beam-funky.ly`’

In complex configurations of knee beaming, according to Paul Roberts, the first stem of a beam determines the direction of the beam, and as such the way that following (kneed) stems attach to the beam. This is in disagreement with the current algorithm.



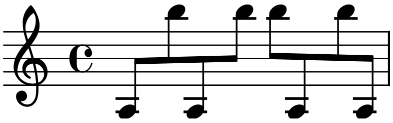
‘`beam-isknee.ly`’

Beams can be placed across a `PianoStaff`.



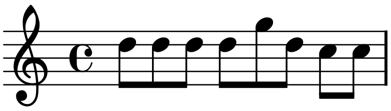
‘`beam-knee-symmetry.ly`’

Point-symmetric beams should receive the same quanting. There is no up/down bias in the quanting code.



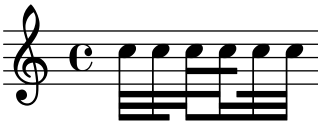
'`beam-length.ly`'

Beams should look the same.



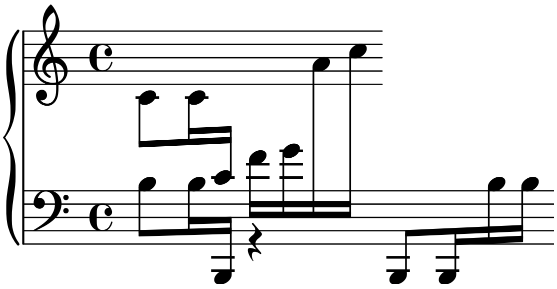
'`beam-manual-beaming.ly`'

Beaming can be overidden for individual stems.



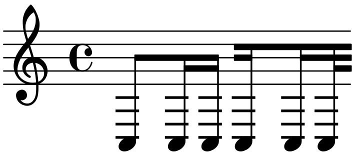
'`beam-multiple-cross-staff.ly`'

Kneed beams (often happens with cross-staff beams) should look good when there are multiple beams: all the beams should go on continuously at the staff change. Stems in both staves reach up to the last beam.



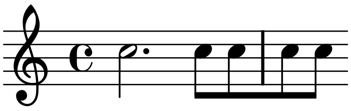
'`beam-outside-beamlets.ly`'

Beams may overshoot stems. This is also controlled with `break-overshoot`.



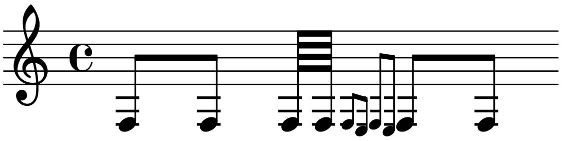
'`beam-over-barline.ly`'

Explicit beams may cross barlines.



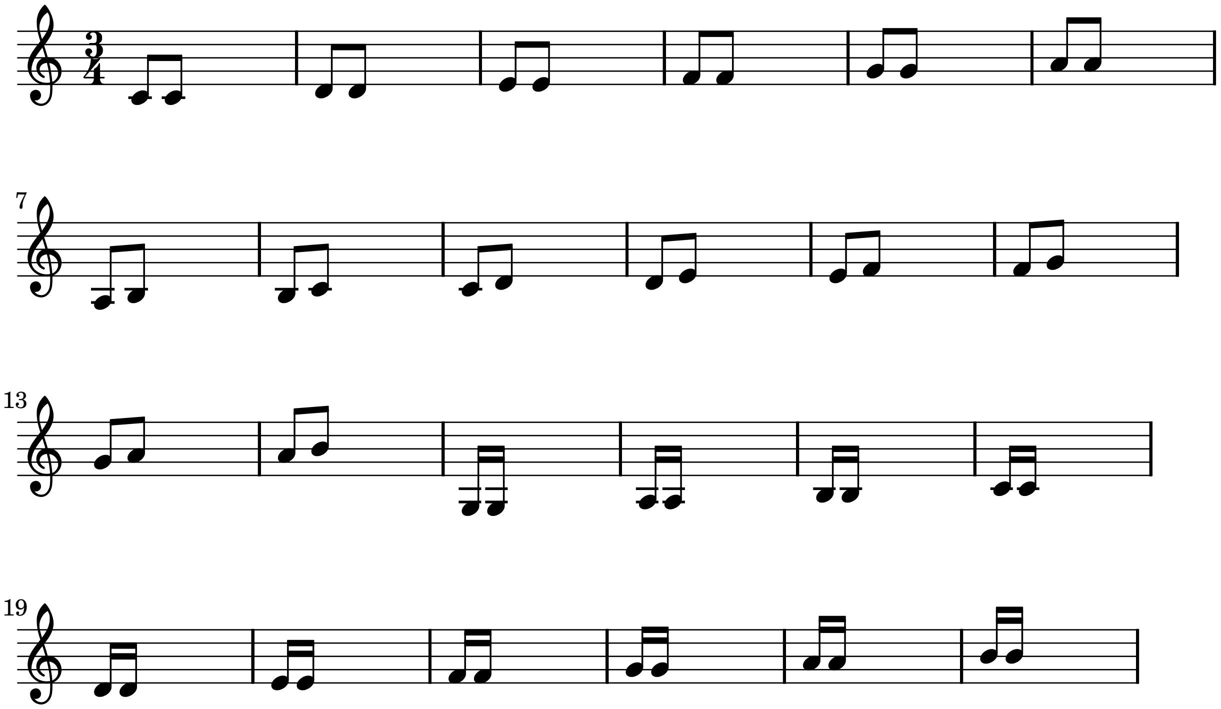
‘`beam-position.ly`’

Beams on ledgered notes should always reach the middle staff line. The second beam counting from the note head side, should never be lower than the second staff line. This does not hold for grace note beams. Override with `no-stem-extend`.



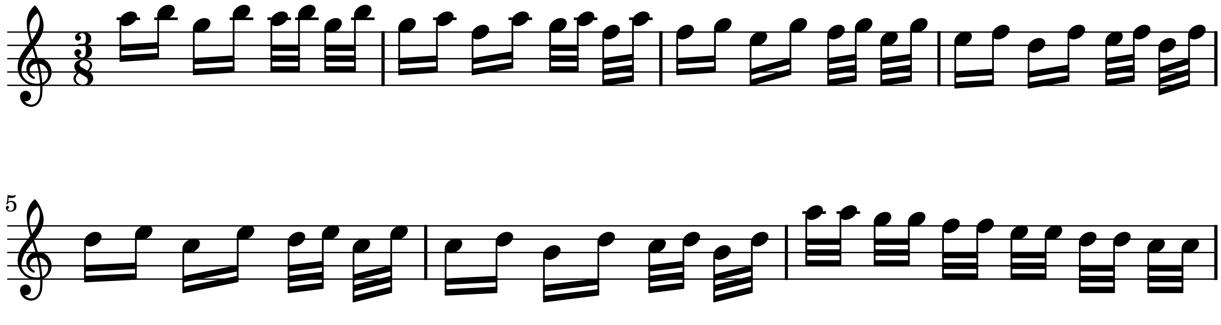
‘`beam-quant-standard.ly`’

This file tests a few standard beam quants, taken from Ted Ross’ book. If LilyPond finds another quant, the correct quant is printed over the beam.



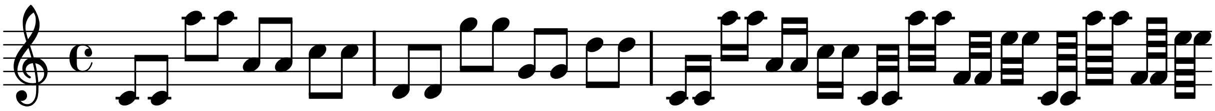
‘`beam-quanting-32nd.ly`’

Stem lengths take precedence over beam quants: ‘forbidden’ quants are only avoided for 32nd beams when they are outside of the staff. However, that leads to very long stems, which is even worse.



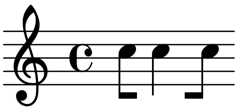
‘`beam-quanting-horizontal.ly`’

In this test for beam quant positions for horizontal beams, staff lines should be covered in all cases. For 32nd beams, the free stem lengths are between 2 and 1.5.



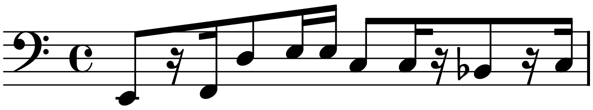
‘`beam-quarter.ly`’

Quarter notes may be beamed: the beam is halted momentarily.



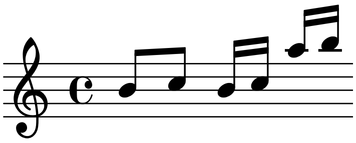
‘`beam-rest.ly`’

The number of beams does not change on a rest.



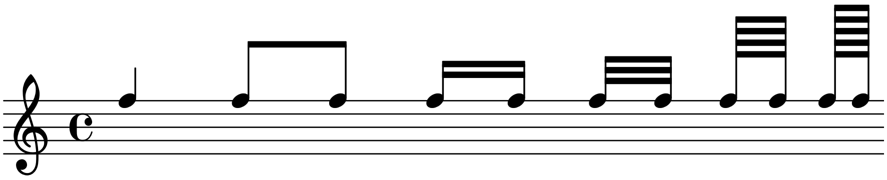
‘`beam-second.ly`’

Engraving second intervals is tricky. We used to have problems with seconds being too steep, or getting too long stems. In a file like this, showing seconds, you’ll spot something fishy very quickly.



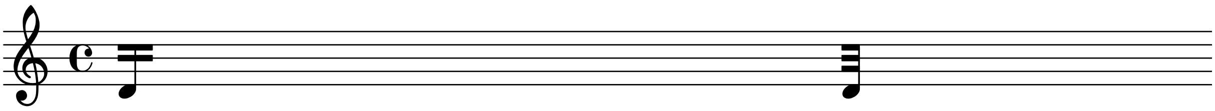
‘`beam-shortened-lengths.ly`’

Beams in unnatural direction, have shortened stems, but do not look too short.



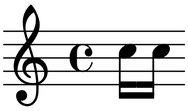
‘`beam-single-stem.ly`’

Single stem beams are also allowed. For such beams, clip-edges is switched off automatically.



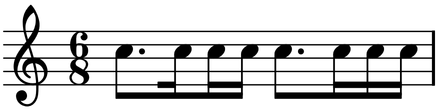
‘`beam-unconnected-beamlets.ly`’

By setting `max-beam-connect`, it is possible to create pairs of unconnected beamlets.



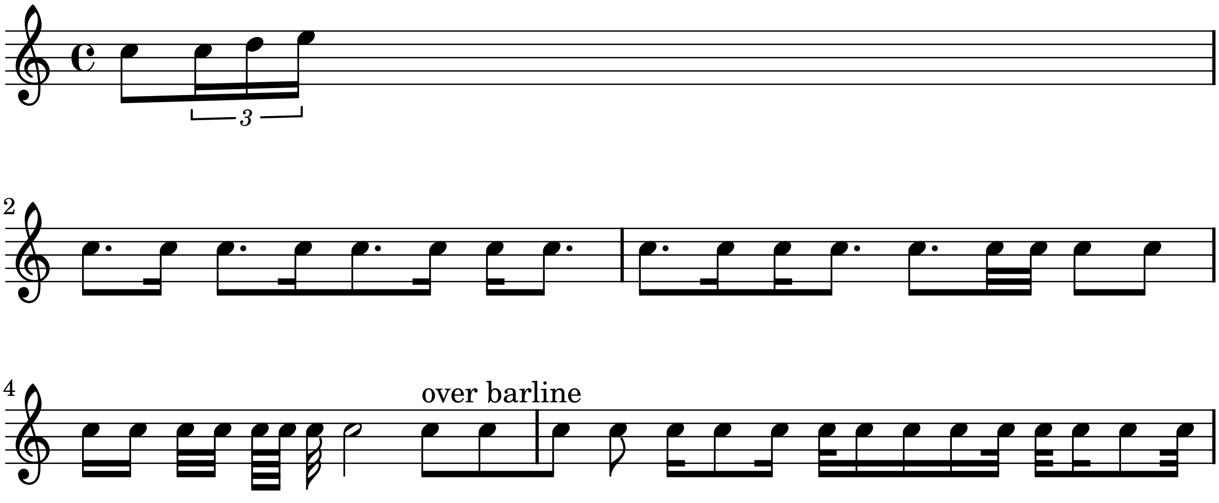
‘`beaming-ternary-metrum.ly`’

Automatic beaming works also in ternary time sigs. In this case, the 8th is a beat, so the 16ths are split into two groups. This can be avoided by overriding `beatLength` to be 3 8th notes.



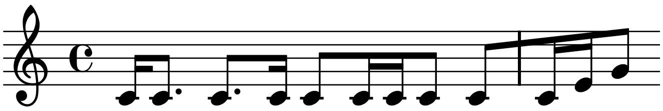
‘`beaming.ly`’

Beaming is generated automatically. Beams may cross bar lines. In that case, line breaks are forbidden.



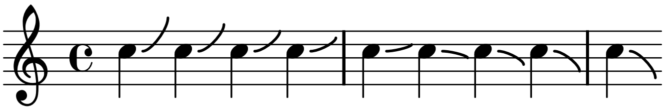
‘`beams.ly`’

Beaming can be also given explicitly.



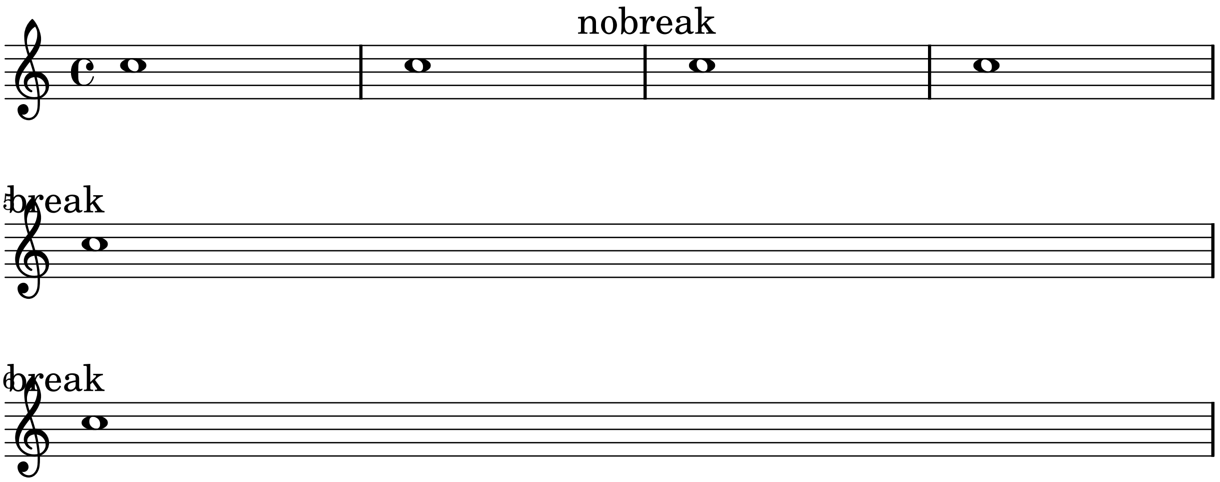
‘`bend-after.ly`’

Falls and doits can be created with bendAfter. They run to the next note, or to the next barline.



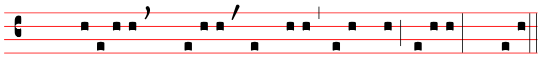
‘`break.ly`’

Breaks can be encouraged and discouraged using `\break` and `\noBreak`.



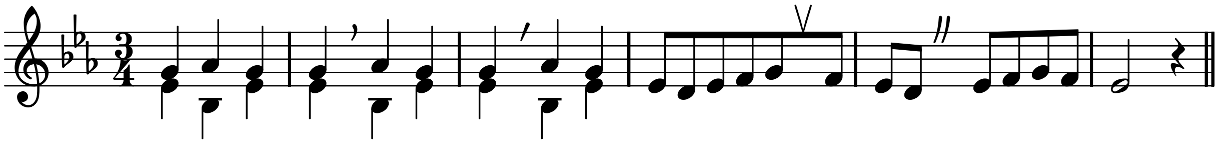
‘`breathing-sign-ancient.ly`’

Gregorian chant notation sometimes also uses commas and ticks, but in smaller font size (we call it ’virgula’ and ’caesura’). However, the most common breathing signs are divisio minima/maior/maxima and finalis, the latter three looking similar to bar glyphs.



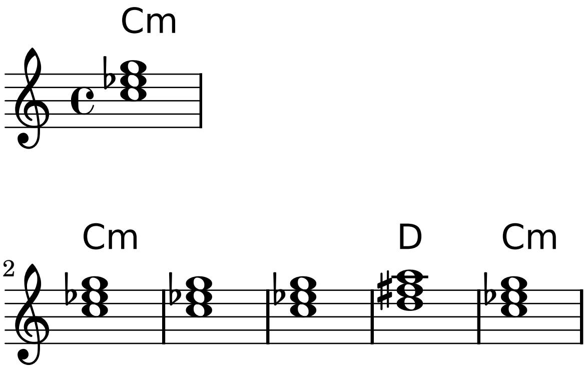
‘`breathing-sign.ly`’

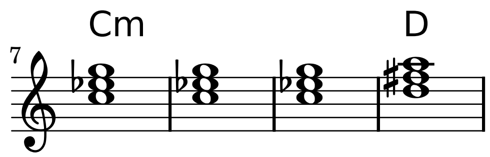
Breathing signs are available in different tastes: commas (default), ticks, vees and ‘railroad tracks’ (caesura).



‘`chord-changes.ly`’

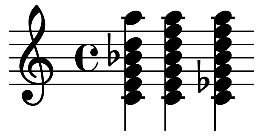
Property chordChanges: display chord names only when there’s a change in the chords scheme, but always display the chord name after a line break.





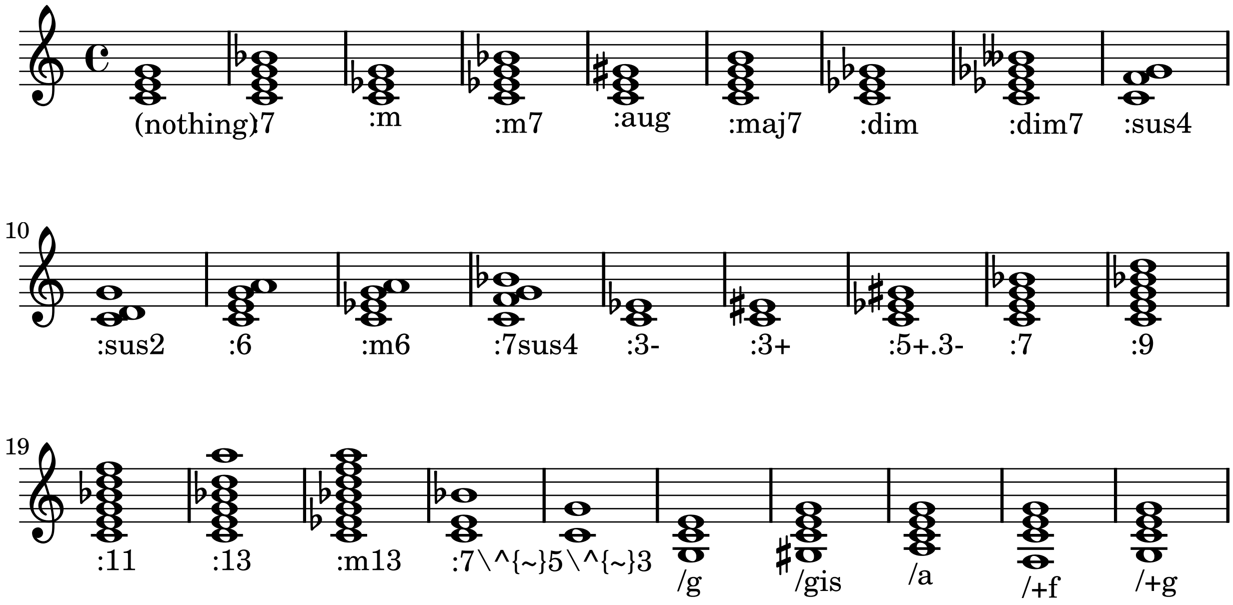
'`chord-name-entry-11.ly`'

The 11 is only added to major-13 if it is mentioned explicitly.



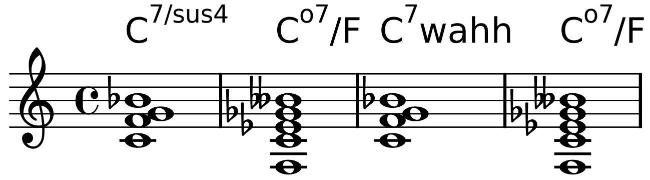
'`chord-name-entry.ly`'

Chords can be produced with the new chordname entry code (`\chordmode` mode), using a pitch and a suffix. Here, the suffixes are printed below pitches.



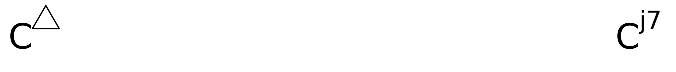
'`chord-name-exceptions.ly`'

The property `chordNameExceptions` can used to store a list of special notations for specific chords.



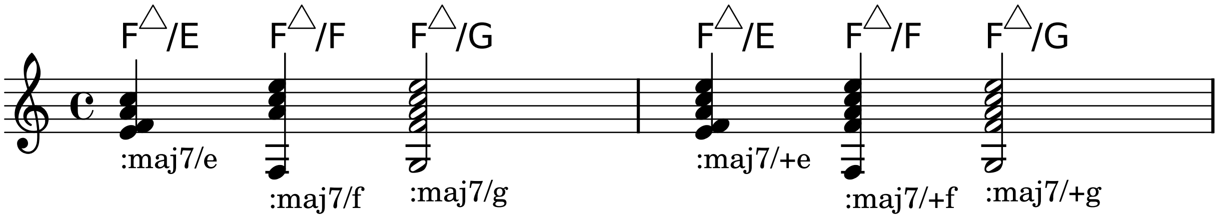
'`chord-name-major7.ly`'

The layout of the major 7 can be tuned with `majorSevenSymbol`.



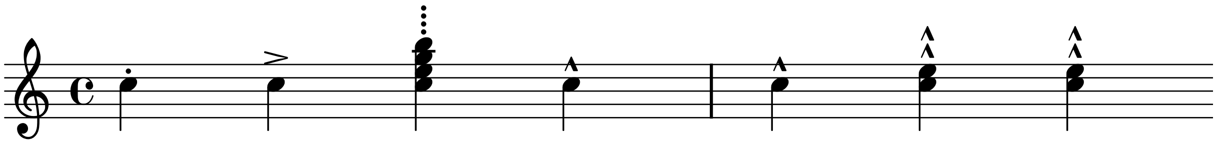
‘`chord-names-bass.ly`’

In ignatzek inversions, a note is dropped down to act as the bass note of the chord. Bass note may be also added explicitly. Above the staff: computed chord names. Below staff: entered chord name.



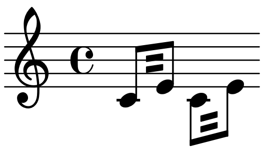
‘`chord-scripts.ly`’

Scripts can also be attached to chord elements.



‘`chord-tremolo-short.ly`’

Tremolo repeats can be constructed for short tremolos (total duration smaller than 1/4) too. Only some of the beams are connected to the stems.

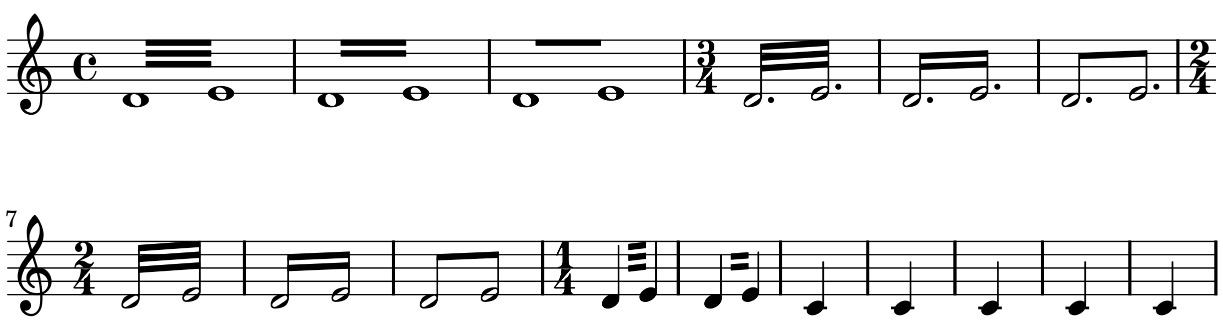


‘`chord-tremolo.ly`’

Chord tremolos look like beams, but are a kind of repeat symbol. To avoid confusion, chord tremolo beams do not reach the stems, but leave a gap. Chord tremolo beams on half notes are not ambiguous, as half notes cannot appear in a regular beam, and should reach the stems.

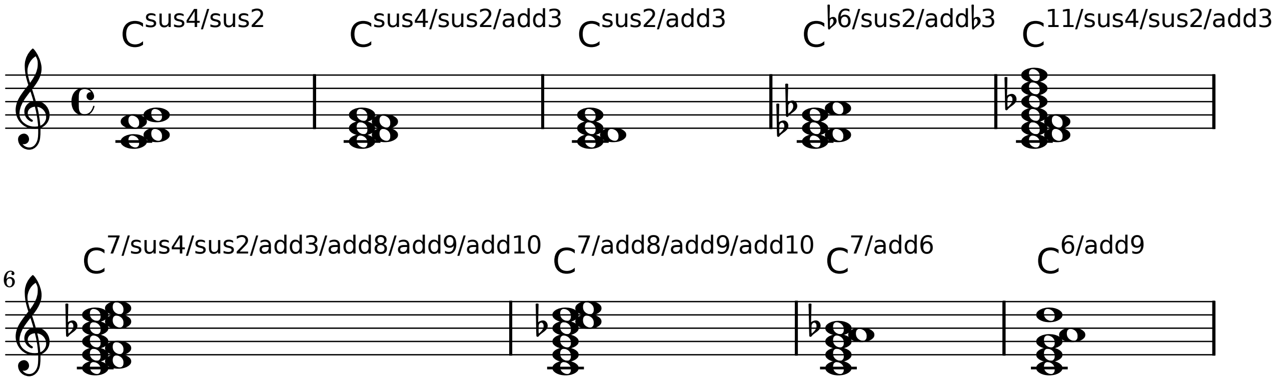
In this example, each tremolo lasts exactly one measure.

(To ensure that the spacing engine is not confused we add some regular notes as well.)



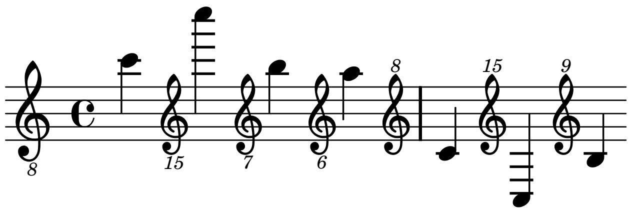
‘`chords-funky-ignatzek.ly`’

Jazz chords may have unusual combinations.



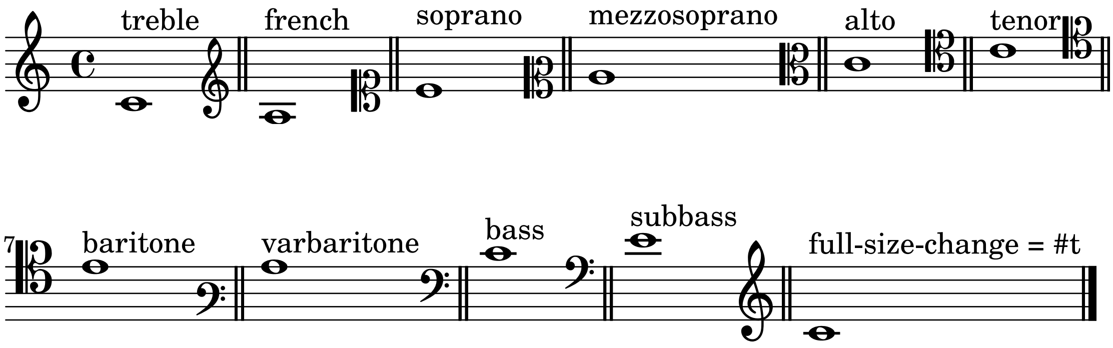
‘`clef-oct.ly`’

Octavation signs may be added to clefs. These octavation signs may be placed below or above (meaning an octave higher or lower), and can take any value, including 15 for two octaves.



‘`clefs.ly`’

Clefs with `full-size-change` should be typeset in full size. For octaviated clefs, the “8” should appear closely above or below the clef respectively.



‘`clip-systems.ly`’

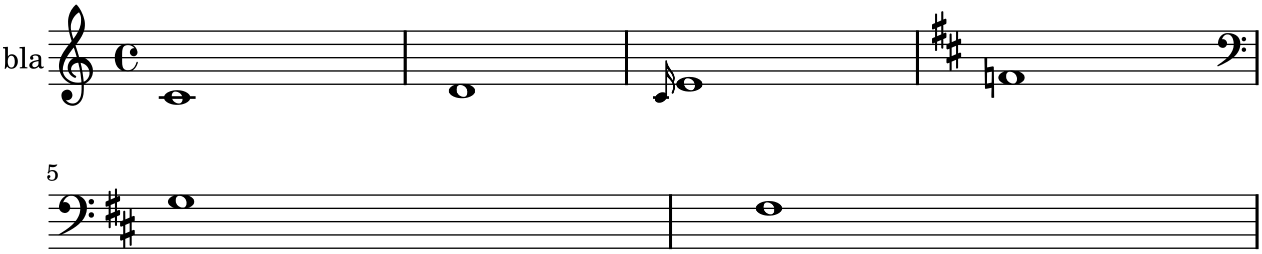
Clipping snippets from a finished score

Notes:

- If system starts and ends are included, they include extents of the System grob, eg. instrument names.
- Grace notes at the end point of the region are not included
- Regions can span multiple systems. In this case, multiple EPS files are generated.

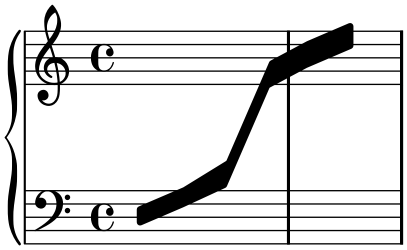
This file needs to be run separately with `-dclip-systems`; the collated-files.html of the regression test does not adequately show the results.

The result will be files named '*base*-from-*start*-to-*end*[-*count*].eps'.



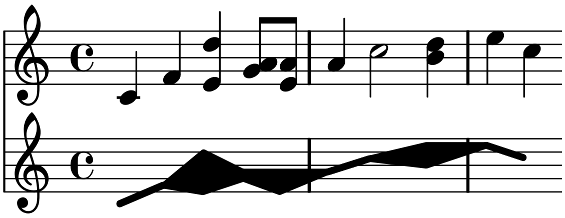
'`cluster-cross-staff.ly`'

Clusters can be written across staves.



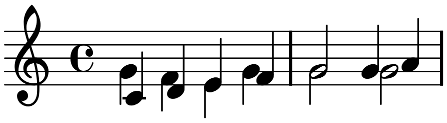
'`cluster.ly`'

Clusters are a device to denote that a complete range of notes is to be played.



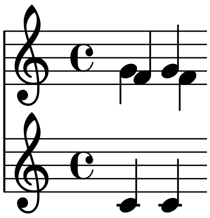
'`collision-2.ly`'

Single head notes may collide.



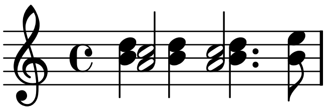
'`collision-alignment.ly`'

Notes in different staves should be aligned to the left-most note, in case of collisions.



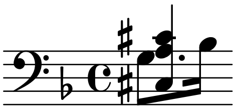
‘`collision-dots-invert.ly`’

When notes are colliding, the resolution depends on the dots: notes with dots should go to the right, if there could be confusion to which notes the dots belong.



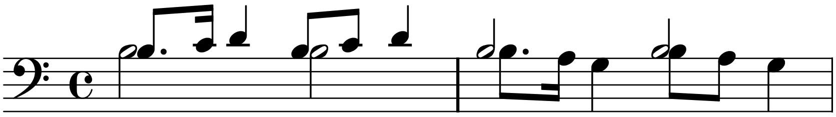
‘`collision-dots-move.ly`’

If collision resolution finds dotted note head must remain on left hand side, move dots to the right.



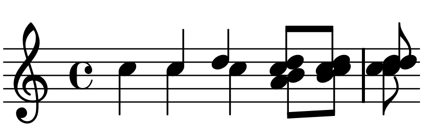
‘`collision-dots.ly`’

Collision resolution tries to put notes with dots on the right side.



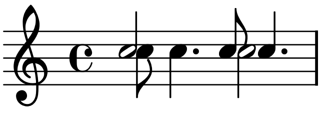
‘`collision-head-chords.ly`’

Note heads in collisions should be merged if they have the same positions in the extreme note heads.



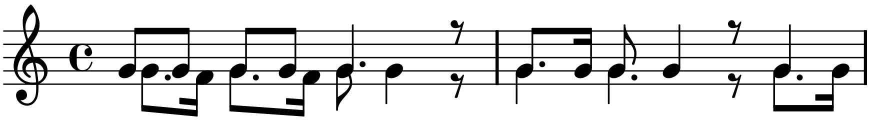
‘`collision-heads.ly`’

Open and black note heads are not merged by default.



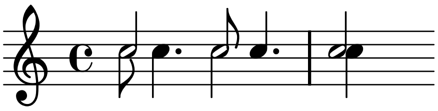
‘`collision-merge-differently-dotted.ly`’

If `NoteCollision` has `merge-differently-dotted = ##t` note heads that have differing dot counts may be merged anyway. Dots should not disappear when merging similar note heads.



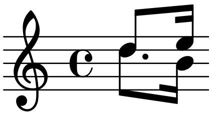
‘`collision-merge-differently-headed.ly`’

If `merge-differently-headed` is enabled, then open note heads may be merged with black noteheads, but only if the black note heads are from 8th or shorter notes.



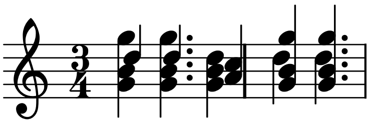
‘`collision-merge-dots.ly`’

When merging heads, the dots are merged too.



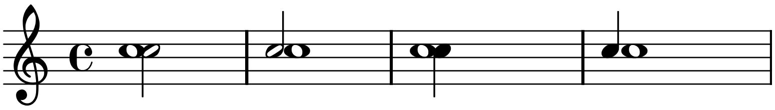
‘`collision-mesh.ly`’

Oppositely stemmed chords, meshing into each other, are resolved.



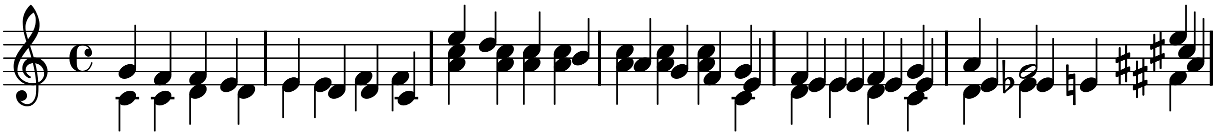
‘`collision-whole.ly`’

Mixed collisions with whole notes require asymmetric shifts.



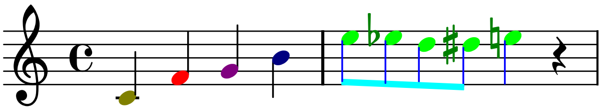
‘`collisions.ly`’

In addition to normal collision rules, there is support for polyphony, where the collosion are avoided by shifting middle voices horizontally.



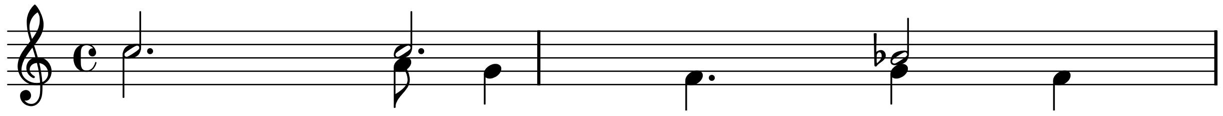
‘`color.ly`’

Each grob can have a color assigned to it. Use the `\override` and `\revert` expressions to set the `color` property.



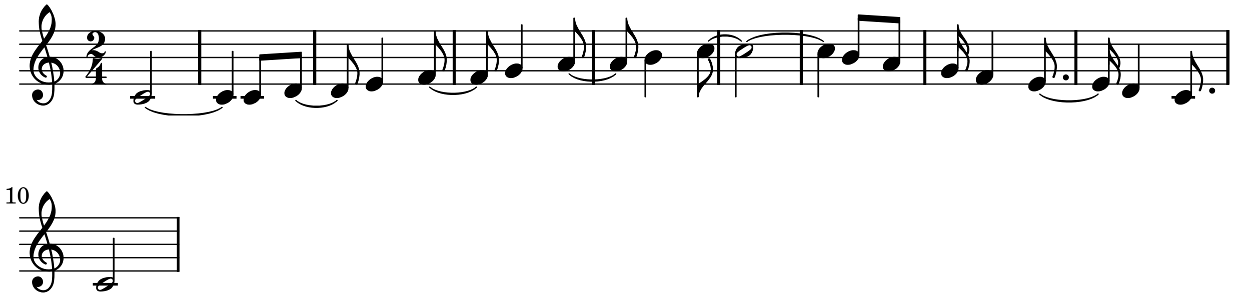
‘`completion-heads-polyphony.ly`’

Completion heads are broken across bar lines. This was intended as a debugging tool, but it can be used to ease music entry. Completion heads are not fooled by polyphony with a different rhythm.



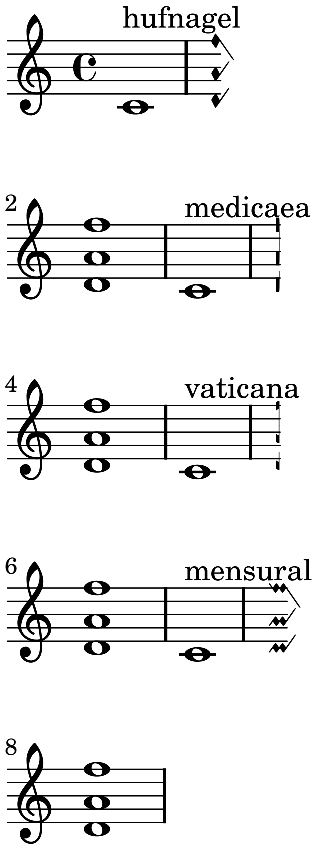
‘`completion-heads.ly`’

If the `Note_heads_engraver` is replaced by the `Completion_heads_engraver`, notes that cross bar lines are split into tied notes.



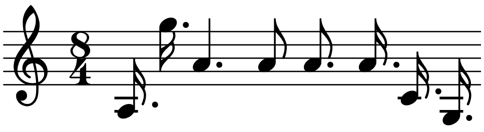
‘`custos.ly`’

Custodes may be engraved in various styles.



'`dot-flag-collision.ly`'

Dots move to the right when a collision with the (up)flag happens.

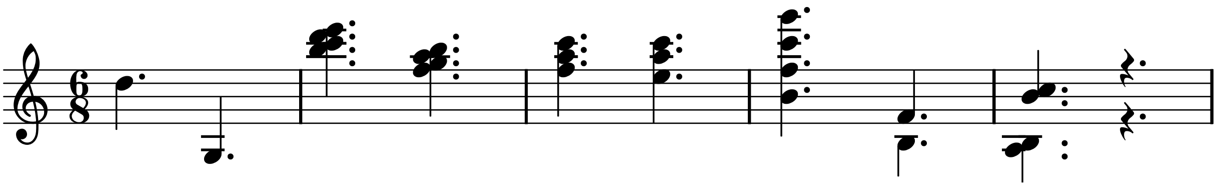


'`dots.ly`'

Noteheads can have dots, and rests too. Augmentation dots should never be printed on a staff line, but rather be shifted vertically. They should go up, but in case of multiple parts, the down stems have down shifted dots. In case of chords, all dots should be in a column. The dots follow the shift of rests when avoiding collisions.

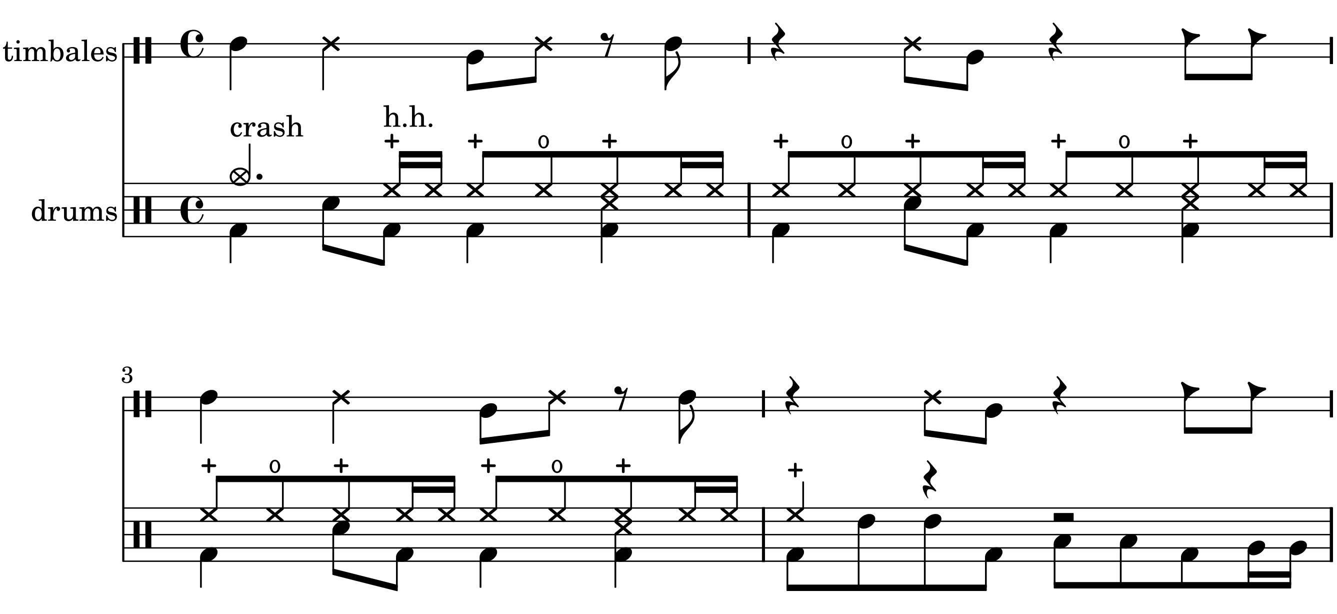
The priorities to print the dots are (ranked in importance):

- keeping dots off staff lines,
- keeping dots close to their note heads,
- moving dots in the direction specified by the voice,
- moving dots up.



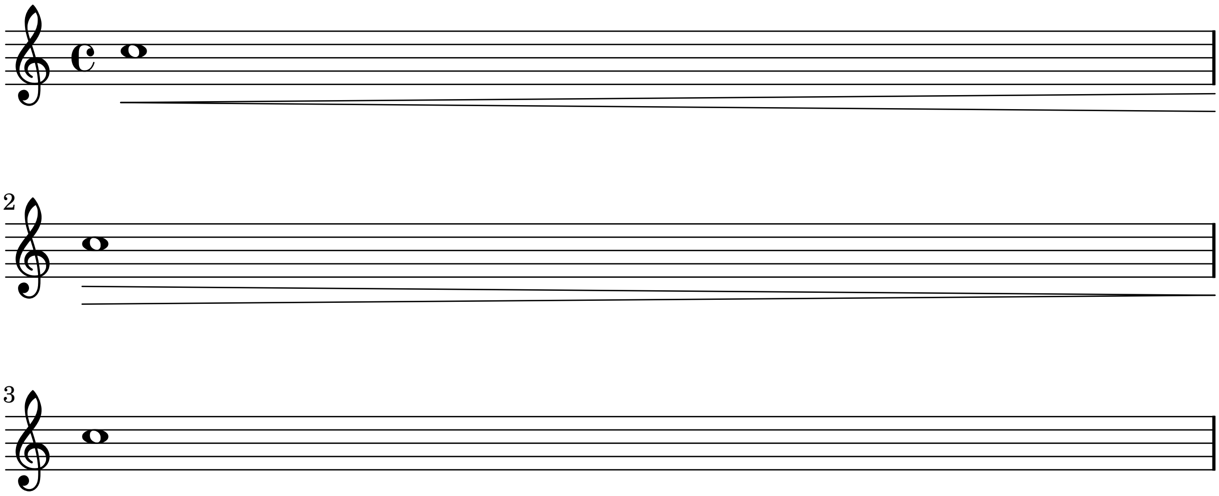
'`drums.ly`'

In drum notation, there is a special clef symbol, drums are placed to their own staff positions and have note heads according to the drum, an extra symbol may be attached to the drum, and the number of lines may be restricted.



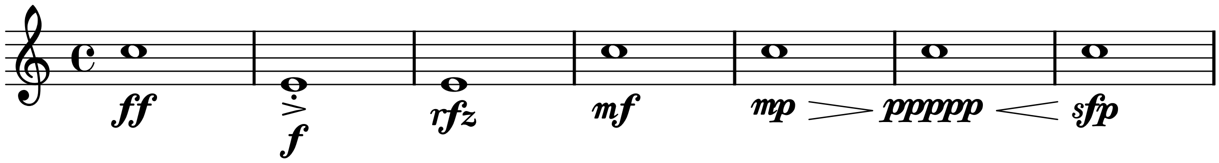
‘`dynamics-broken-hairpin.ly`’

Broken crescendi should be open on one side.



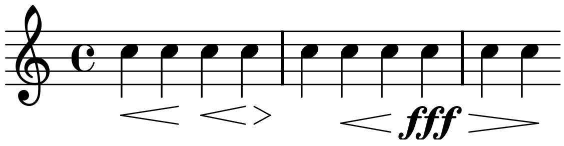
‘`dynamics-glyphs.ly`’

Dynamic letters are kerned, and their weight matches that of the hairpin signs. The dynamic scripts should be horizontally centered on the note head. Scripts that should appear closer to the note head (staccato, accent) are reckoned with.



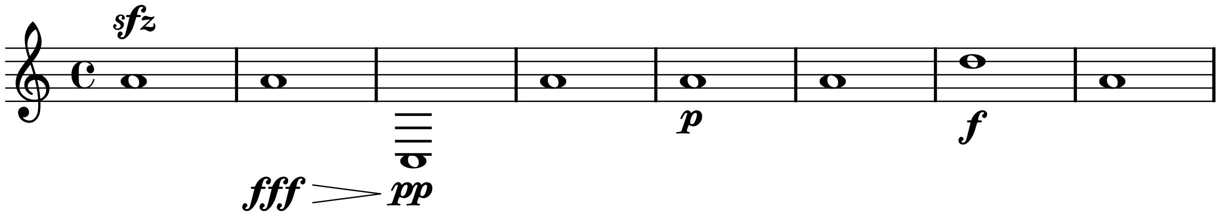
‘`dynamics-hairpin-length.ly`’

Hairpins extend to the extremes of the bound if there is no adjacent hairpin of dynamic-text. If there is, the hairpin extends to the center of the column or the bound of the text respectively.



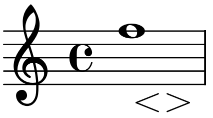
‘`dynamics-line.ly`’

Dynamics appear below or above the staff. If multiple dynamics are linked with (de)crescendi, they should be on the same line. Isolated dynamics may be forced up or down.



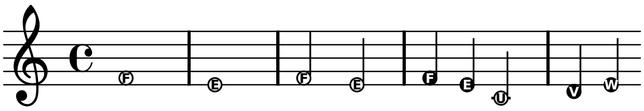
‘`dynamics-unbound-hairpin.ly`’

Crescendi may start off-notes, however, they should not collapse into flat lines.



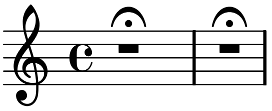
‘`easy-notation.ly`’

Easy-notation (or Ez-notation) prints names in note heads. You also get ledger lines, of course.



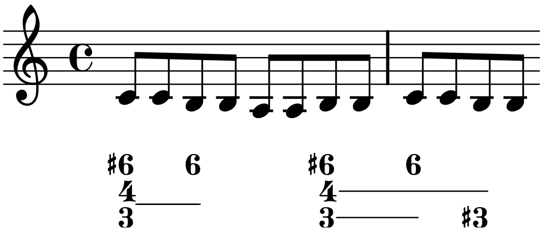
‘`fermata-rest-position.ly`’

Fermatas over multimeasure rests are positioned as over normal rests.



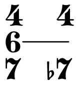
‘`figured-bass-continuation-center.ly`’

Pairs of congruent figured bass extender lines are vertically centered if `figuredBassCenterContinuations` is set to true.



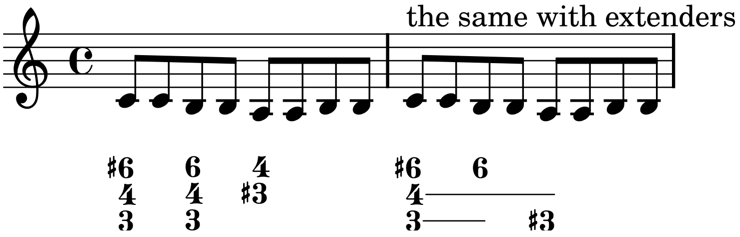
‘`figured-bass-continuation-forbid.ly`’

By adorning a bass figure with `\!`, an extender may be forbidden.



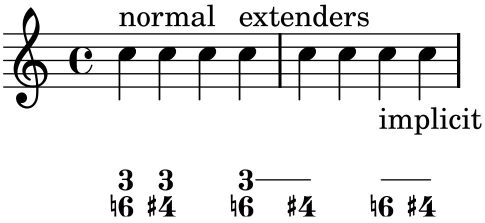
‘`figured-bass-continuation.ly`’

Figured bass extender lines run between repeated bass figures. They are switched on with `useBassFigureExtenders`



‘`figured-bass-implicit.ly`’

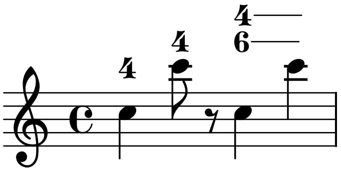
Implicit bass figures are not printed, but they do get extenders.



‘`figured-bass-staff.ly`’

Figured bass can also be added to Staff context directly. In that case, the figures must be entered with `\figuremode` and be directed to an existing `Staff` context.

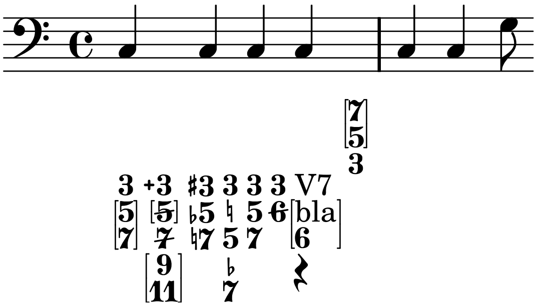
Since these engravers are on `Staff` level, properties controlling figured bass should be set in `Staff` context.



‘`figured-bass.ly`’

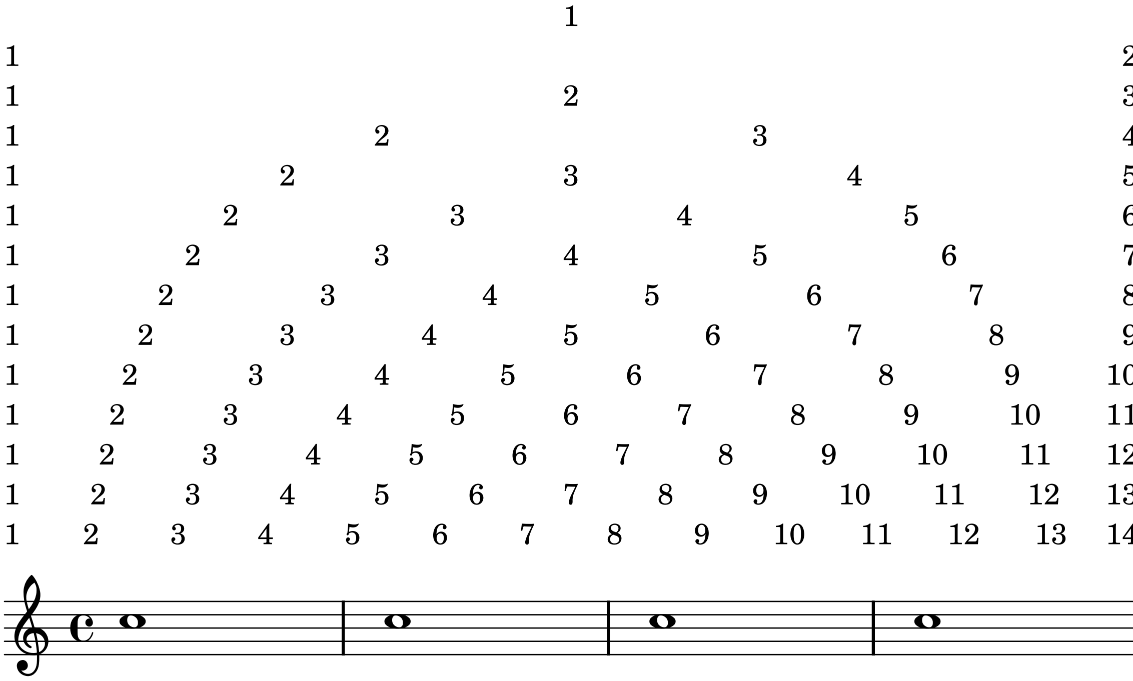
Figured bass is created by the FiguredBass context which responds to figured bass events and rest events. You must enter these using the special `\figuremode { }` mode, which allows you to type numbers, like `<4 6+>` and add slashes and pluses.

You can also enter markup strings. The vertical alignment may also be tuned.



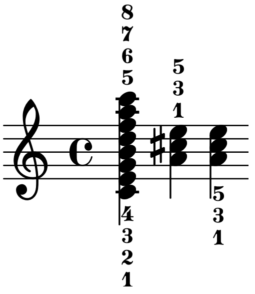
‘`fill-line-test.ly`’

The fill-line markup command should align texts in columns. For examlpe, the characters in the center should form one column.


1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12 13
1 2 3 4 5 6 7 8 9 10 11 12 13 14

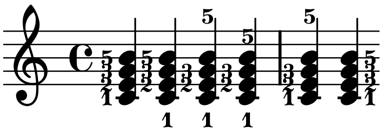
‘`finger-chords-order.ly`’

Ordering of the fingerings depends on vertical ordering of the notes, and is independent of up/down direction.



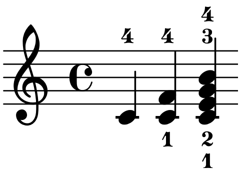
‘`finger-chords.ly`’

With the new chord syntax, it is possible to associate fingerings uniquely with notes. This makes it possible to add horizontal fingerings to notes.



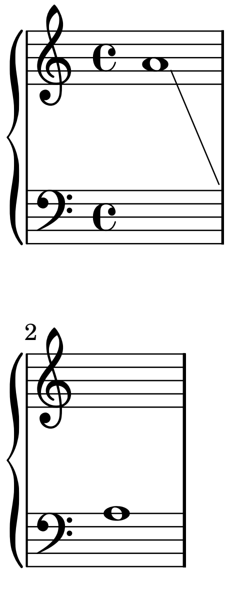
‘`fingering.ly`’

Automatic fingering tries to put fingering instructions next to noteheads.



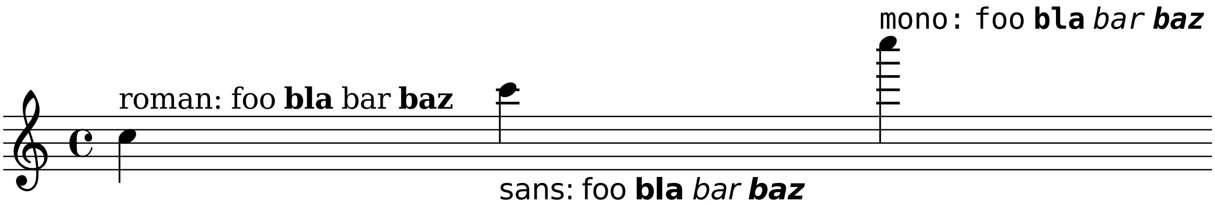
‘`follow-voice-break.ly`’

The line-spanners connects to the Y position of the note on the next line. When put across line breaks, only the part before the line break is printed.



‘`font-family-override.ly`’

The default font families for text can be overridden with `make-pango-font-tree`

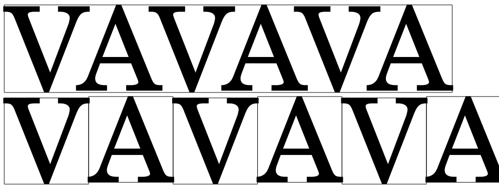

roman: foo **bla** bar **baz**
mono: foo **bla** *bar* ***baz***
sans: foo **bla** *bar* ***baz***

‘`font-kern.ly`’

Text set in TrueType Fonts that contain kerning tables, are kerned.

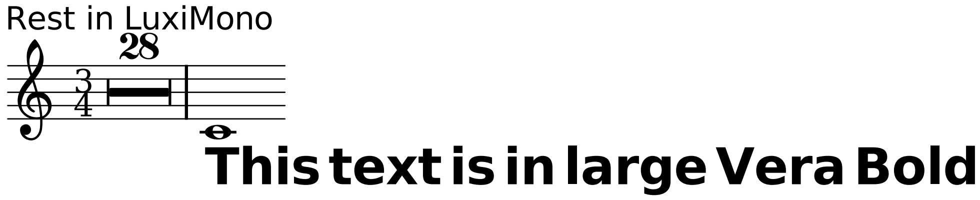
With kerning:

Without kerning:



‘`font-name.ly`’

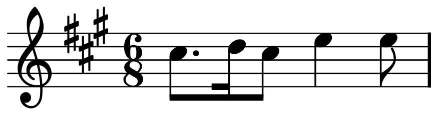
Other fonts can be used by setting `font-name` for the appropriate object. The string should be a Pango font description without size specification.



‘`font-postscript.ly`’

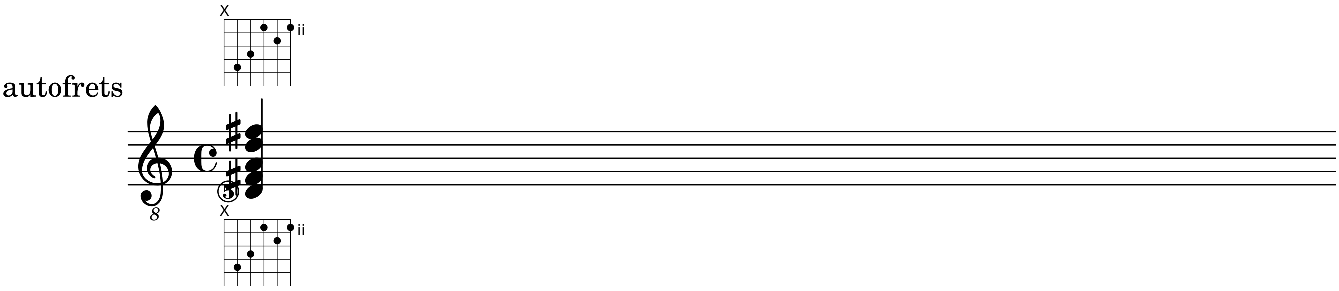
This file demonstrates how to load different (postscript) fonts. The file ‘`font.scm`’ shows how to define the scheme-function `make-century-schoolbook-tree`.

This file should be run with the TeX and extra options should be passed to LaTeX and dvips to help it find the uncb font.

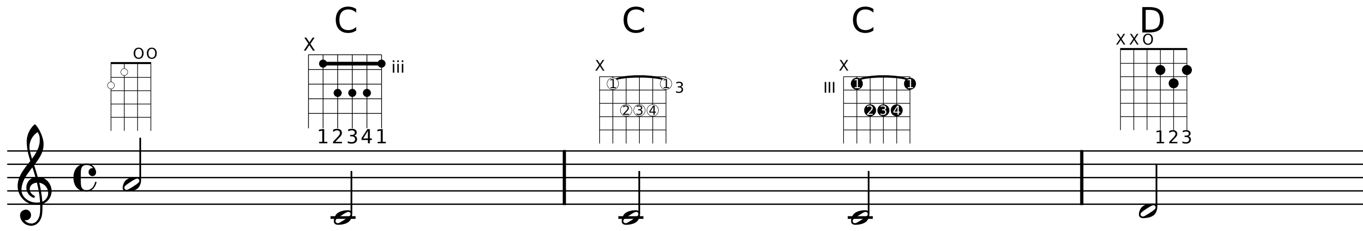


‘`fret-boards.ly`’

Frets can be assigned automatically. The results will be best when one string number is indicated in advance

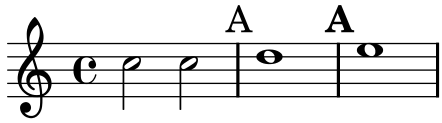


‘`fret-diagrams.ly`’



‘`generic-output-property.ly`’

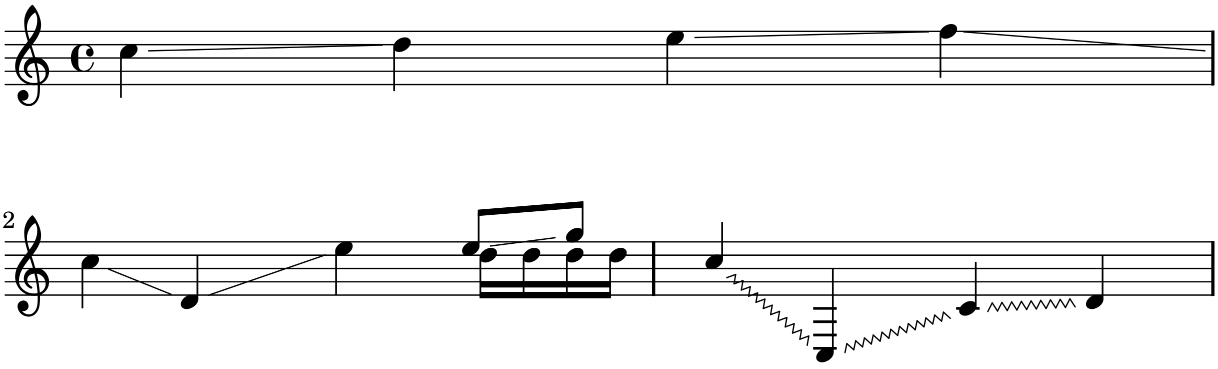
As a last resort, the placement of grobs can be adjusted manually, by setting the `extra-offset` of a grob.



‘`glissando.ly`’

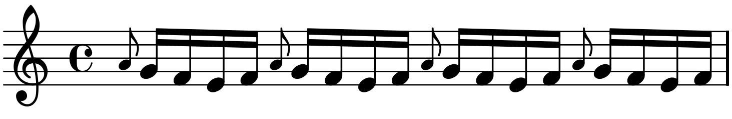
Between notes, there may be simple glissando lines. Here, the first two glissandi are not consecutive.

The engraver does no time-keeping, so it involves some trickery to get `<< {` s8 s8 s4 `} {` c4 \gliss d4 `} >>` working correctly.



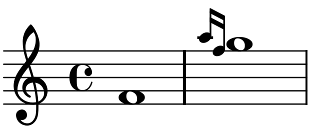
‘`grace-auto-beam.ly`’

The autobeamer is not confused by grace notes.



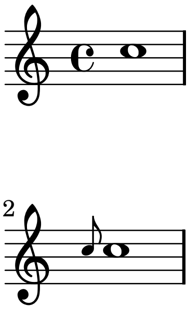
‘`grace-bar-line.ly`’

Bar line should come before the grace note.



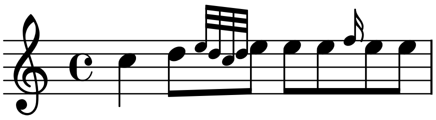
‘`grace-bar-number.ly`’

Grace notes do tricky things with timing. If a measure starts with a grace note, the measure does not start at 0, but earlier. Nevertheless, lily should not get confused. For example, line breaks should be possible at grace notes, and the bar number should be printed correctly.



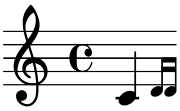
'`grace-beam.ly`'

Grace beams and normal beams may occur simultaneously. Unbeamed grace notes are not put into normal beams.



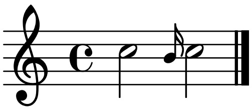
'`grace-end.ly`'

Grace notes after the last note do not confuse the timing code.



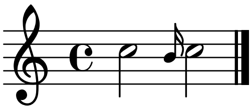
'`grace-nest.ly`'

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



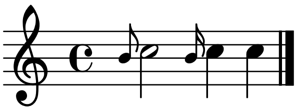
'`grace-nest1.ly`'

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



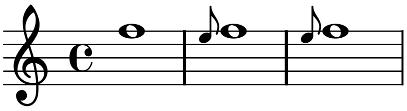
'`grace-nest2.ly`'

Grace code should not be confused by nested sequential musics, containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.



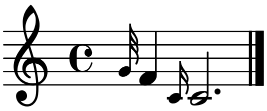
'`grace-nest3.ly`'

In nested syntax, graces are still properly handled.



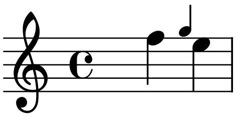
'`grace-nest4.ly`'

Also in the nested syntax here, grace notes appear rightly.



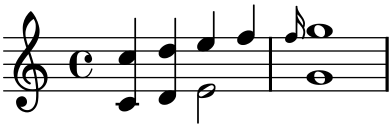
'`grace-nest5.ly`'

Graces notes may have the same duration as the main note.



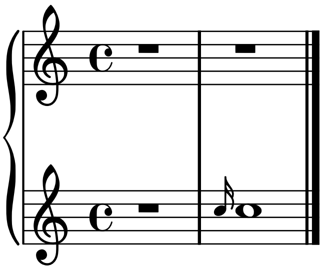
'`grace-part-combine.ly`'

Grace notes may be put in a `partcombiner`.



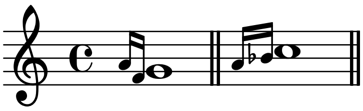
'`grace-staff-length.ly`'

Stripped version of trip.ly. Staves should be of correct length.



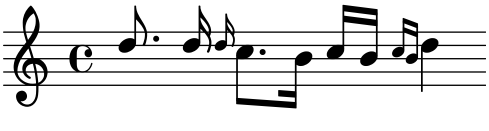
'`grace-start.ly`'

Pieces may begin with grace notes.



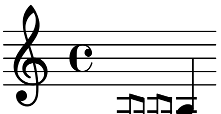
'`grace-stem-length.ly`'

Stem lengths for grace notes should be shorter than normal notes, if possible. They should never be longer, even if that would lead to beam quanting program.



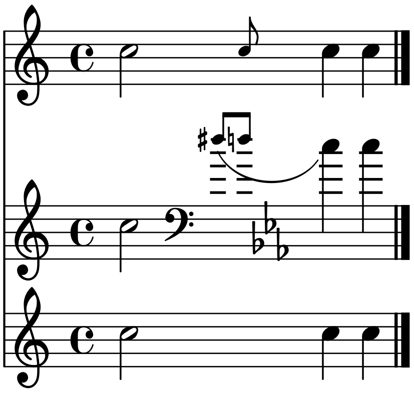
‘`grace-stems.ly`’

Here `startGraceMusic` should set `no-stem-extend` to true; the two grace beams should be the same here.



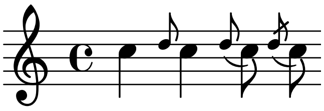
‘`grace-sync.ly`’

Grace notes in different voices/staves are synchronized.



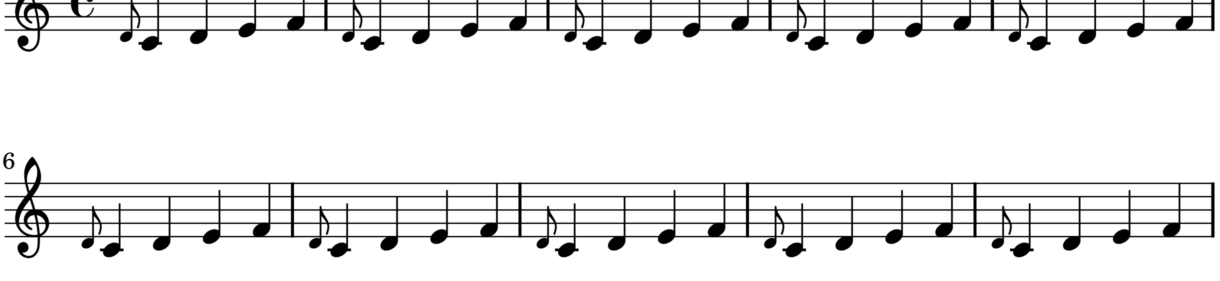
‘`grace-types.ly`’

There are three different kinds of grace types: the base grace switches to smaller type, the appoggiatura inserts also a slur, and the acciaccatura inserts a slur and slashes the stem.



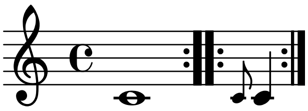
‘`grace-unfold-repeat.ly`’

When grace notes are entered with unfolded repeats, line breaks take place before grace notes.



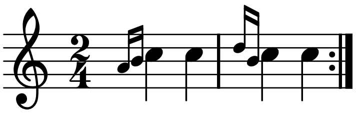
‘`grace-volta-repeat-2.ly`’

A volta repeat may begin with a grace. Consecutive ending and starting repeat bars are merged into one `:||:`.



‘`grace-volta-repeat.ly`’

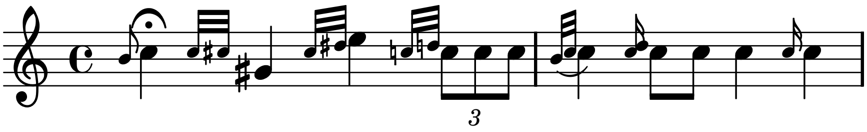
Repeated music can start with grace notes. Bar checks preceding the grace notes do not cause synchronization effects.



‘`grace.ly`’

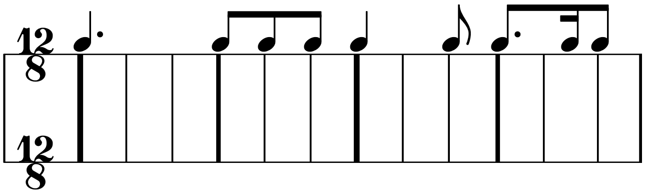
You can have beams, notes, chords, stems etc. within a `\grace` section. If there are tuplets, the grace notes will not be under the brace.

Main note scripts do not end up on the grace note.



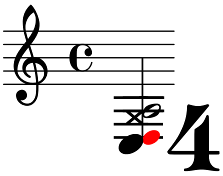
‘`grid-lines.ly`’

With grid lines, vertical lines can be drawn between staves synchronized with the notes.



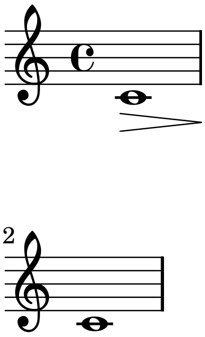
‘`grob-tweak.ly`’

With the `\tweak` function, individual grobs that are directly caused by events may be tuned directly.



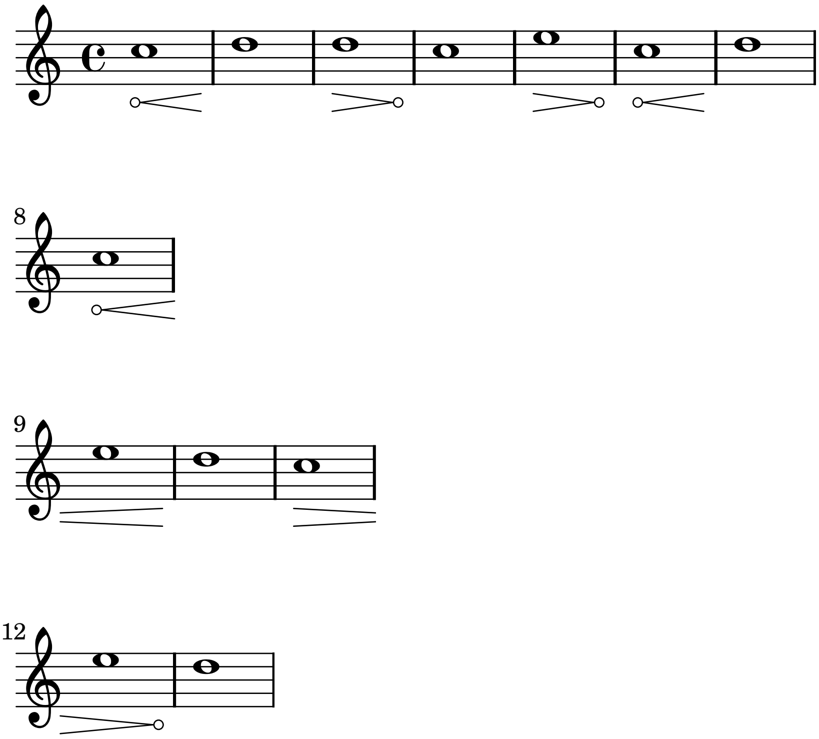
'`hairpin-barline-break.ly`'

If a hairpin ends on the first note of a new stave, we don't print that ending. But on the previous line, this hairpin should not be left open, and should end at the barline.



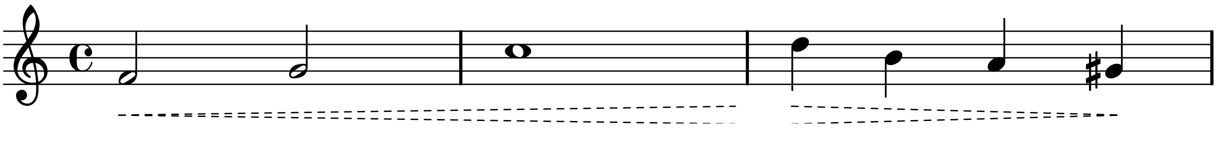
'`hairpin-circled.ly`'

Hairpins can have circled tips. A decrescendo del niente followed by a crescendo al niente should only print one circle.



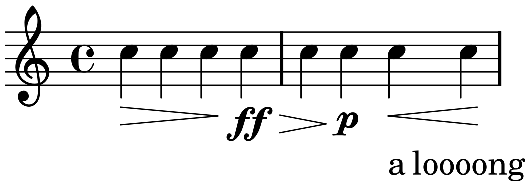
'`hairpin-dashed.ly`'

Hairpin crescendi may be dashed.



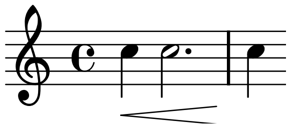
‘`hairpin-ending.ly`’

Hairpin dynamics start under notes if there are no text-dynamics. If there are text dynamics, the hairpin does not run into them.



‘`hairpin-to-barline.ly`’

By setting `hairpinToBarline`, hairpins will stop at the barline preceding the ending note.

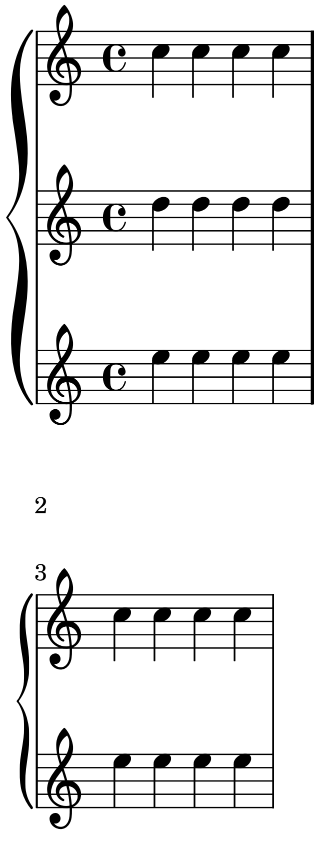


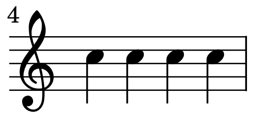
‘`hara-kiri-pianostaff.ly`’

Hara-kiri staves kill themselves if they are empty. This example really contains three staves, but as they progress, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered a bug, however, the scenario of all staves disappearing does not happen in practice.)

Any staff brackets and braces are removed, both in the single staff and no staff case.

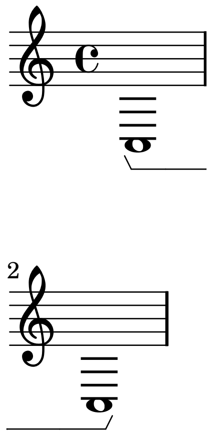
This example was done with a pianostaff, which has fixed distance alignment; this should not confuse the mechanism.





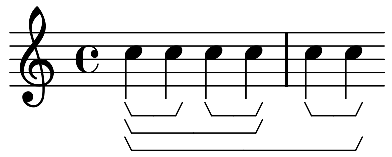
‘`horizontal-bracket-break.ly`’

Horizontal brackets connect over line breaks.



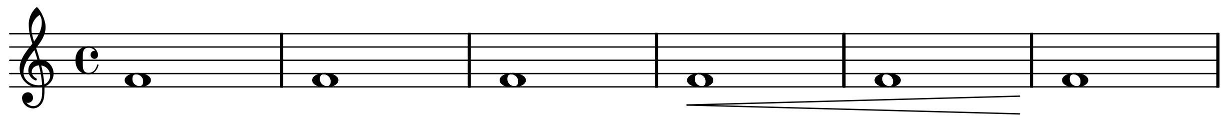
‘`horizontal-bracket.ly`’

Note grouping events are used to indicate where analysis brackets start and end.



‘`instrument-name-dynamic.ly`’

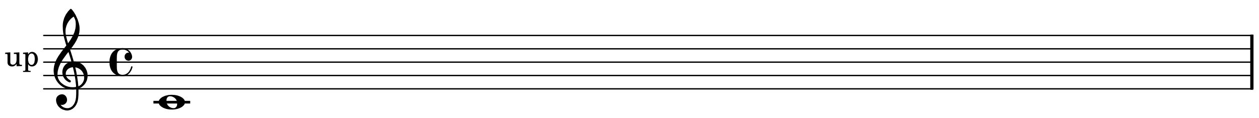
Instrument names (aligned on axis group spanners) ignore dynamic and pedal line spanners.



‘`instrument-name-hara-kiri.ly`’

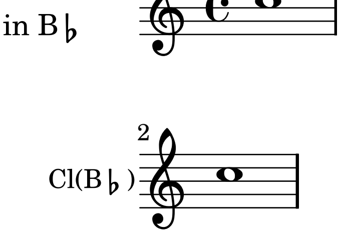
`PianoStaff.instrument` and `PianoStaff.instr` are removed when the staves are killed off.

In this example, the 2nd staff (marked by the barnumber 2) disappears as does the instrument name.



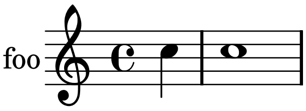
‘`instrument-name-markup.ly`’

Instrument names are set with `Staff.instrument` and `Staff.instr`. You can enter markup texts to create more funky names, including alterations.



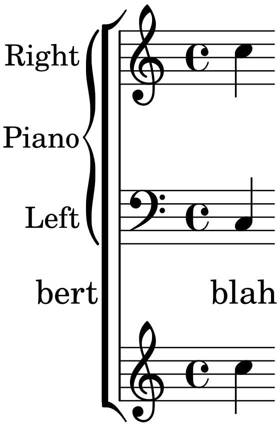
‘`instrument-name-partial.ly`’

Instrument names are also printed on partial starting measures.



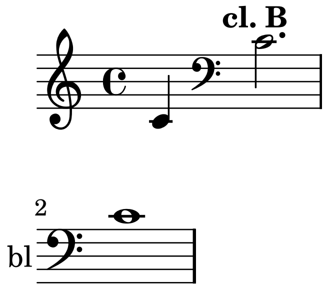
‘`instrument-name.ly`’

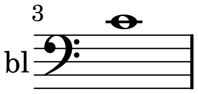
Staff margins are also markings attached to barlines. They should be left of the staff, and be centered vertically with respect to the staff. They may be on normal staves, but also on compound staves, like the PianoStaff.



‘`instrument-switch.ly`’

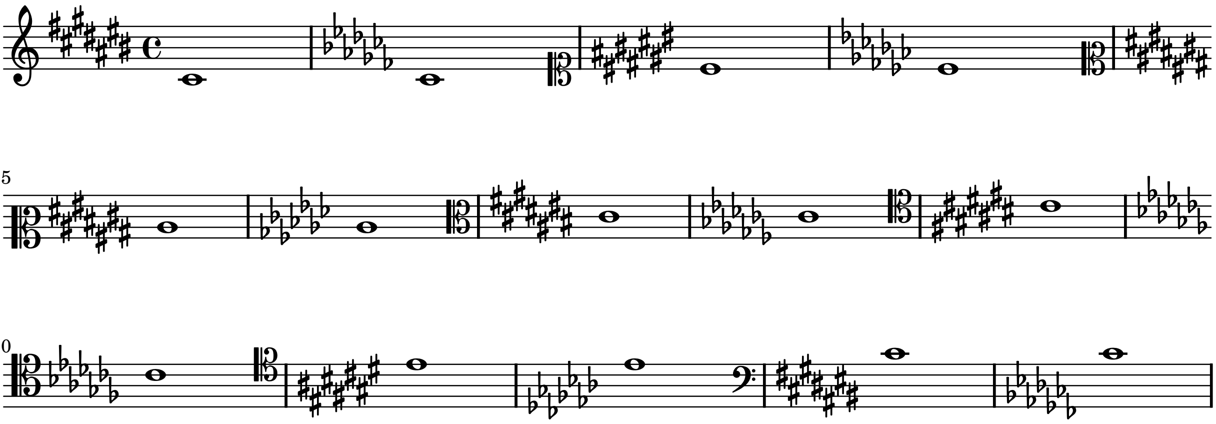
The `switchInstrument` music function modifies properties for an in staff instrument switch.





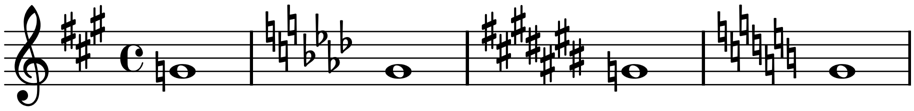
‘`key-clefs.ly`’

Each clef have own accidental placing rules.



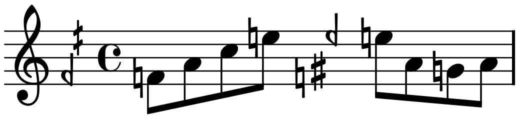
‘`key-signature-cancellation.ly`’

Key cancellation signs consists of naturals for pitches that are not in the new key signature. Naturals get a little padding so the stems don’t collide.



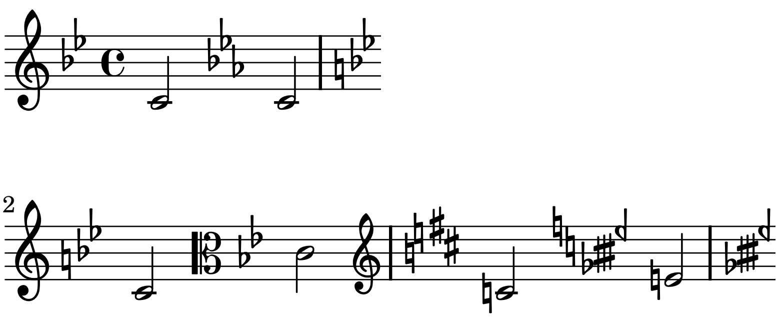
‘`key-signature-scordatura.ly`’

By setting `Staff.keySignature` directly, key signatures can be set invidually per pitch.



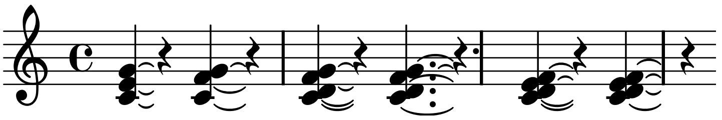
‘`keys.ly`’

Key signatures may appear on key changes, even without a barline. In the case of a line break, the restoration accidentals are printed at end of a line. If `createKeyOnClefChange` is set, key signatures are created also on a clef change.

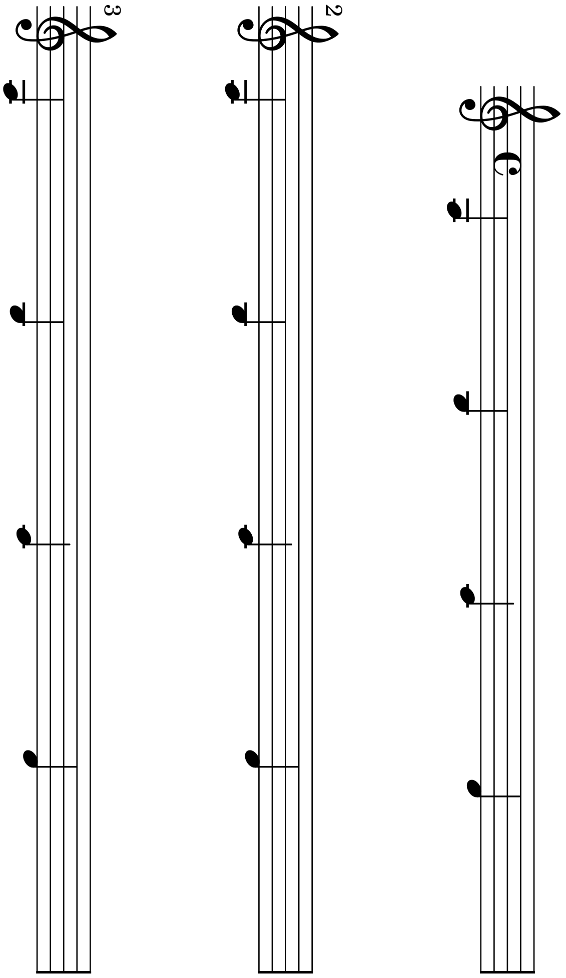


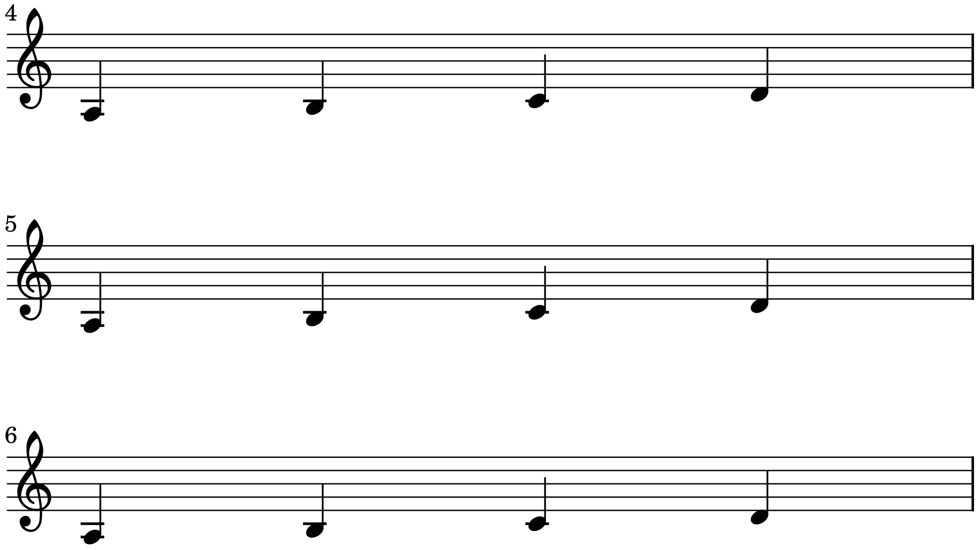
‘`laissez-vibrer-ties.ly`’

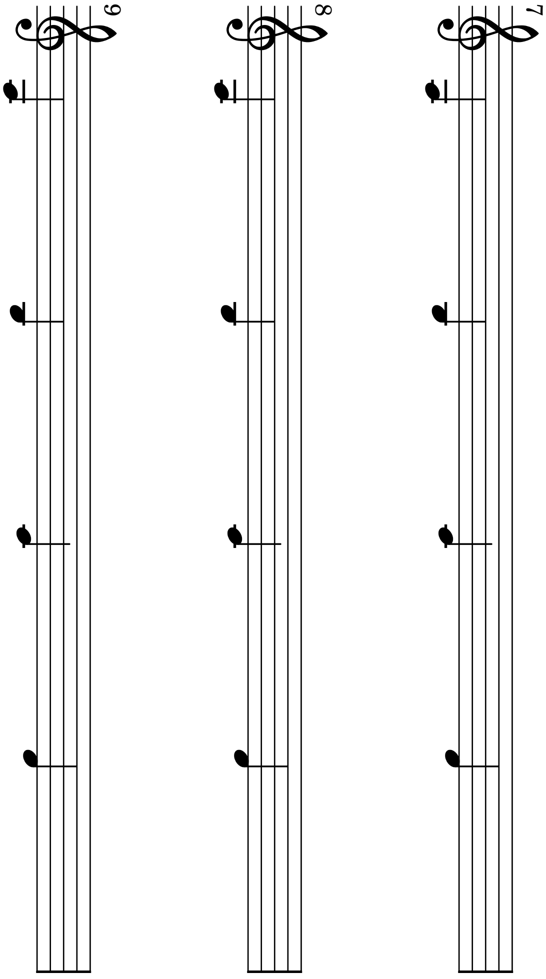
l.v. ties should avoid dots and staff lines, similar to normal ties. They have fixed size. Their formatting can be tuned with `tie-configuration`.

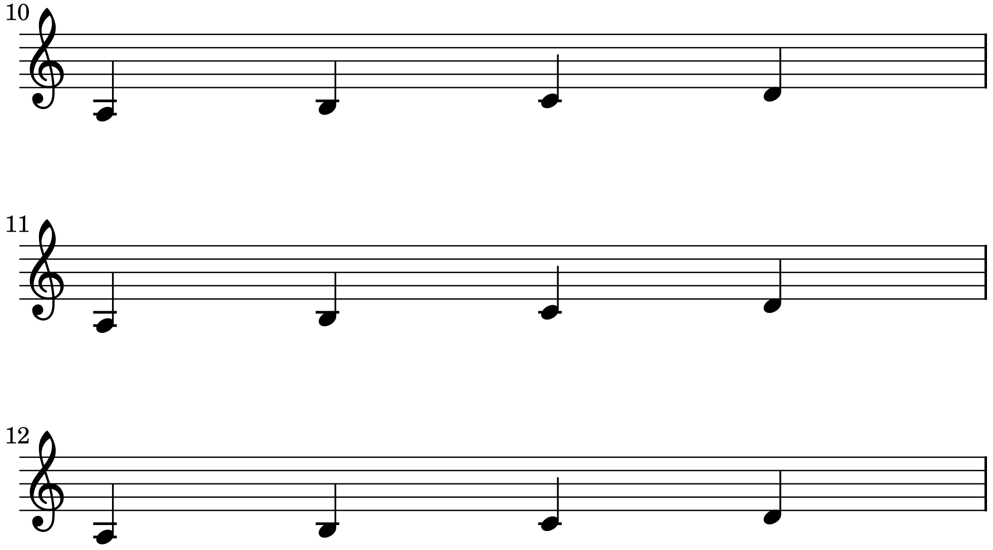


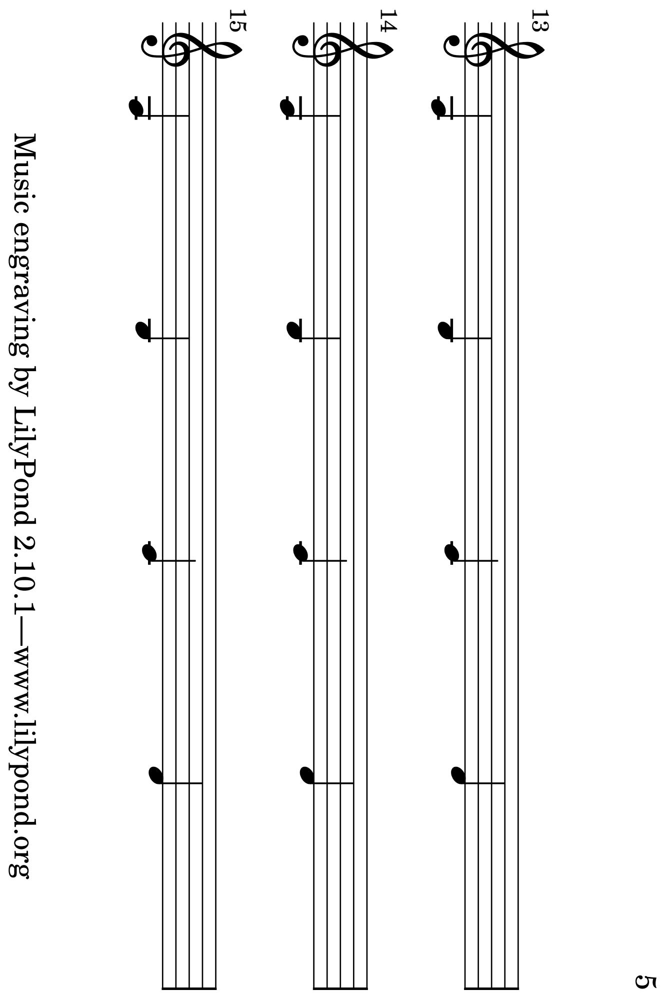
‘`landscape.ly`’




4
5
6

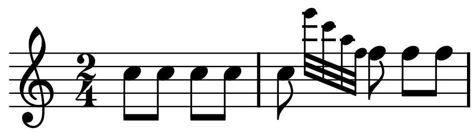

7
8
9


10
11
12



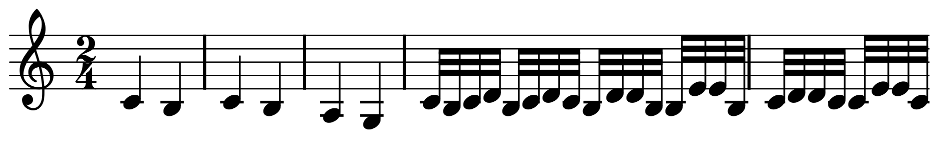
‘`ledger-line-minimum.ly`’

When ledgered notes are very close, for example, in grace notes, they are kept at a minimum distance to prevent the ledgers from disappearing.



‘`ledger-line-shorten.ly`’

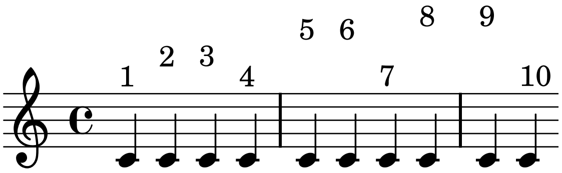
Ledger lines are shortened when they are very close. This ensures that ledgers lines stay separate.



‘`lily-in-scheme.ly`’

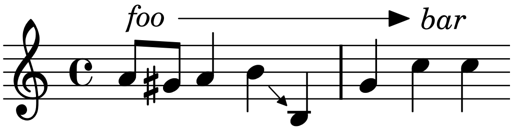
LilyPond syntax can be used inside scheme to build music expressions, with the `#{ ... #}` syntax. Scheme forms can be introduced inside these blocks by escaping them with a `$`, both in a LilyPond context or in a Scheme context.

In this example, the `\withpaddingA`, `\withpaddingB` and `\withpaddingC` music functions set different kinds of padding on the `TextScript` grob.



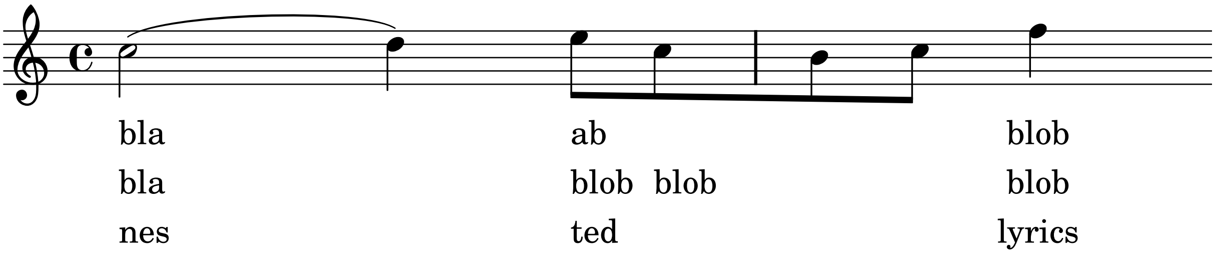
`'line-arrows.ly'`

Arrows can be applied to text-spanners and line-spanners (such as the Glissando)



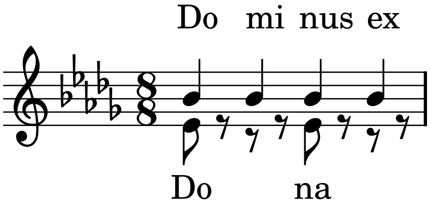
`'lyric-combine-new.ly'`

With the `\lyricsto` mechanism, individual lyric lines can be associated with one melody line. For each lyric line, can be tuned whether to follow melismata or not.



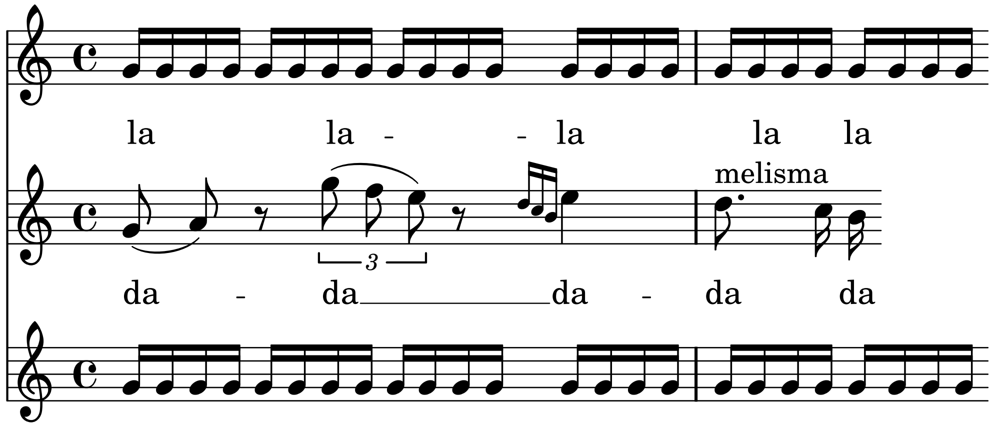
`'lyric-combine-polyphonic.ly'`

Polyphonic rhythms and rests do not disturb `\lyricsto`.



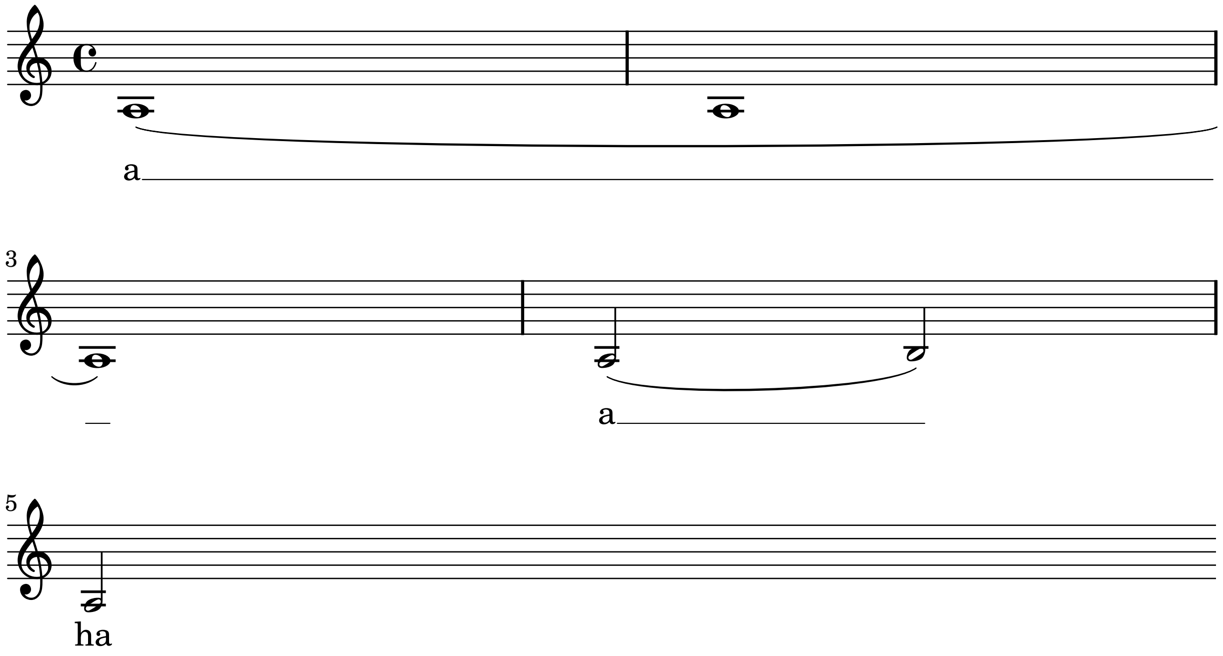
`'lyric-combine.ly'`

Lyrics can be set to a melody automatically. Excess lyrics will be discarded. Lyrics will not be set over rests. You can have melismata either by setting a property melismaBusy, or by setting automaticMelismas (which will set melismas during slurs and ties). If you want a different order than first Music, then Lyrics, you must precook a chord of staves/lyrics and label those. Of course, the lyrics ignores any other rhythms in the piece.



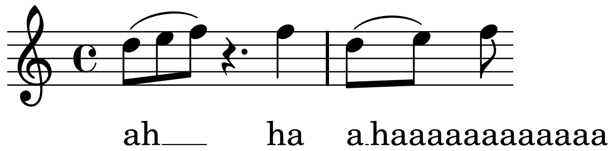
‘lyric-extender-broken.ly’

Lyric extenders run to the end of the line if it continues the next line. Otherwise, it should run to the last note of the melisma.



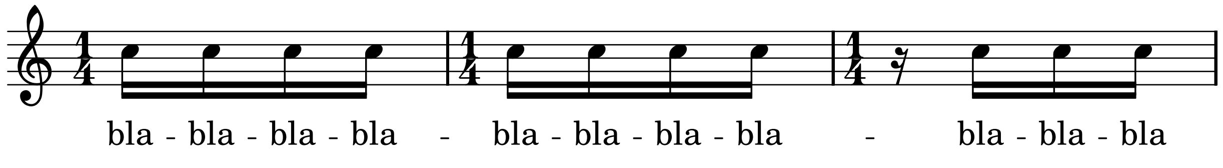
‘lyric-extender.ly’

A LyricExtender may span several notes. A LyricExtender does not extend past a rest, or past the next lyric syllable.



‘lyric-hyphen-break.ly’

Hyphens are print at the beginning of the line only when they go past the first note.

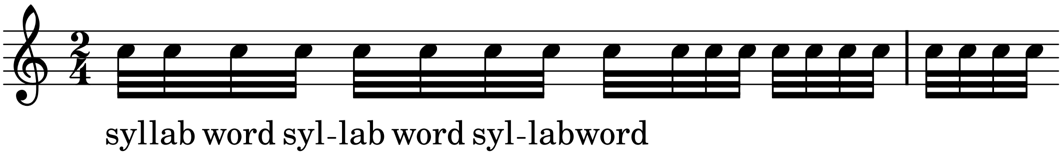

bla - bla - bla - bla - bla - bla - bla - bla - bla - bla - bla

'`lyric-hyphen-retain.ly`'

The minimum distance between lyrics are determined by the `minimum-distance` of `LyricHyphen` and `LyricSpace`.

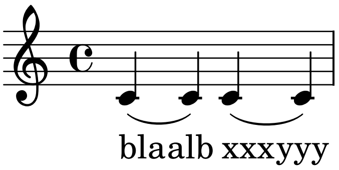
The ideal length of a hyphen is determined by its `length` property, but it may be shortened down to `minimum-length` in tight situations. If in this it still does not fit, the hyphen will be omitted.

Like all overrides within `\lyricsto` and `\addlyrics`, the effect of a setting is delayed is one syllable.



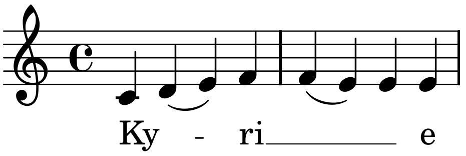
'`lyric-hyphen.ly`'

In lyrics, hyphens may be used.



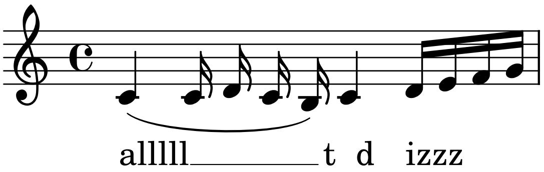
'`lyric-melisma-manual.ly`'

Melisma's may be entered manually by substituting `_` for lyrics on notes that are part of the melisma.



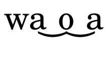
'`lyric-phrasing.ly`'

Normally, the lyric is centered on the note head. However, on melismata, the text is left aligned on the left-side of the note head.



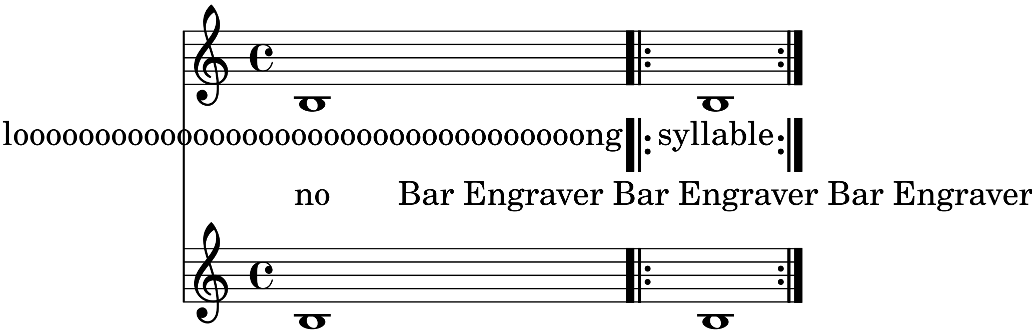
'`lyric-tie.ly`'

Tildes in lyric syllables are converted to tie symbols.



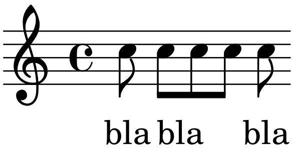
'`lyrics-bar.ly`'

Adding a `Bar_engraver` to the Lyrics context makes sure that lyrics do not collide with barlines.


loooooooooooooooooooooooooooooooooooooong
syllable
no
Bar Engraver Bar Engraver Bar Engraver

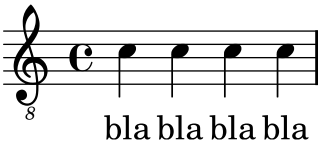
'`lyrics-melisma-beam.ly`'

Melismata are triggered by manual beams.



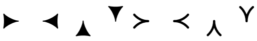
'`lyrics-tenor-clef.ly`'

Lyrics are not lowered despite the presence of an octavation 8.



'`markup-arrows.ly`'

The feta font has arrow heads



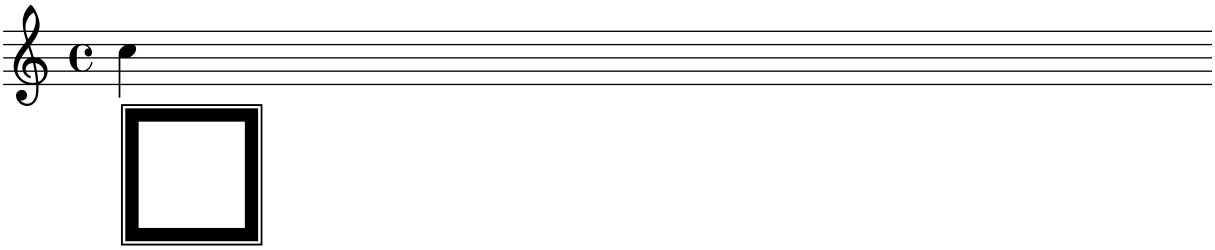
'`markup-bidi-pango.ly`'

A single pango string is considered to have one direction. The hebrew in this example (including punctuation) is set right-to-left, with the first word (containing 1) on the right.

לל1ללל, רר2רר.

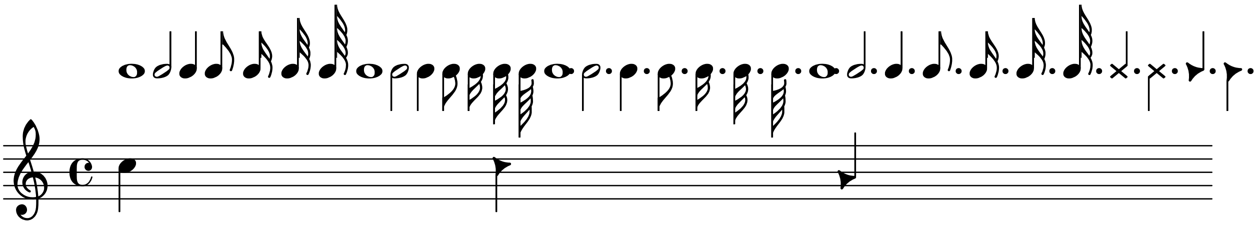
‘`markup-eps.ly`’

The epsfile markup command reads an EPS file



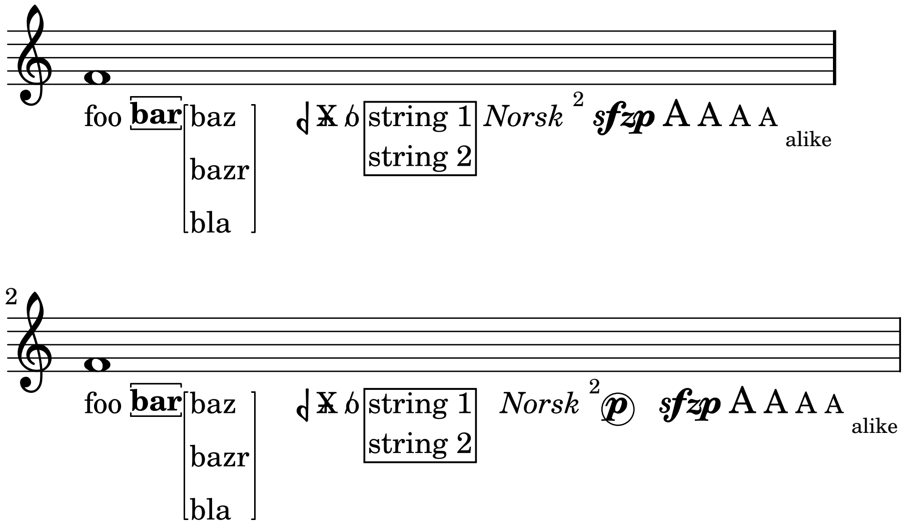
‘`markup-note.ly`’

The note markup function may be used to make metronome markings. It works for a variety of flag, dot and duration settings.



‘`markup-scheme.ly`’

There is a Scheme macro `markup` to produce markup texts using a similar syntax as `\markup`.



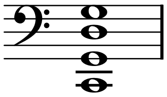
‘`markup-score.ly`’

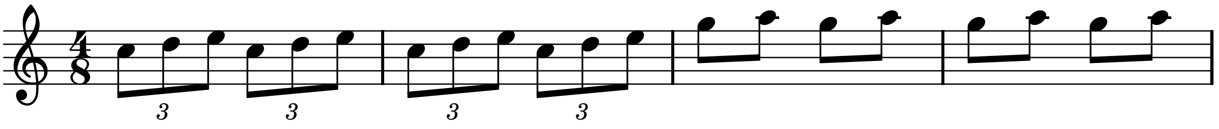
Use \score block as markup command.

Solo Cello Suites

Suite IV

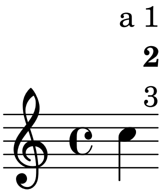
Originalstimmung:





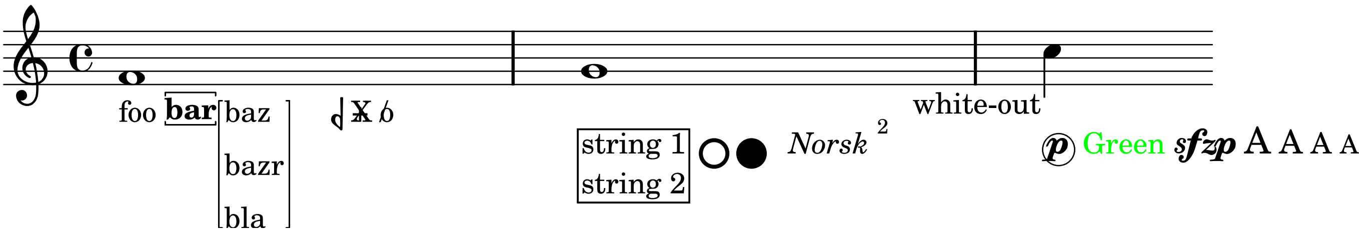
‘`markup-stack.ly`’

Markup scripts may be stacked.



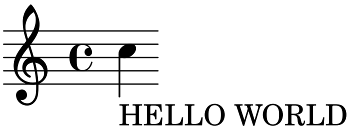
‘`markup-syntax.ly`’

Demo of markup texts, using LilyPond syntax.



‘`markup-user.ly`’

Own markup commands may be defined by using the `define-markup-command` scheme macro.



‘`markup-word-wrap.ly`’

The markup commands `\wordwrap` and `\justify` produce simple paragraph text.

this is normal text This is a test of the wordwrapping function. 1 This is a test of the wordwrapping function. 2 This is a test of the wordwrapping function. 3 This is a test of the wordwrapping function. 4 1a111 11111 **22222** *2222* continuing

this is normal text This is a test of the wordwrapping function, but with justification. 1 This is a test of the wordwrapping function, but with justification. 2 This is a test of $\frac{a}{b}$ the wordwrapping function, but with justification. 3 This is a test of the wordwrapping function, but with justification. bla bla continuing

Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum.

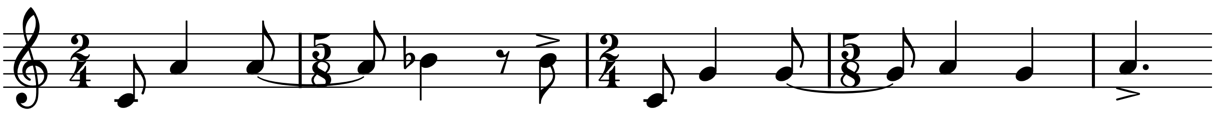
Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate.

Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum Om mani padme hum.

Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate Gate Gate paragate.

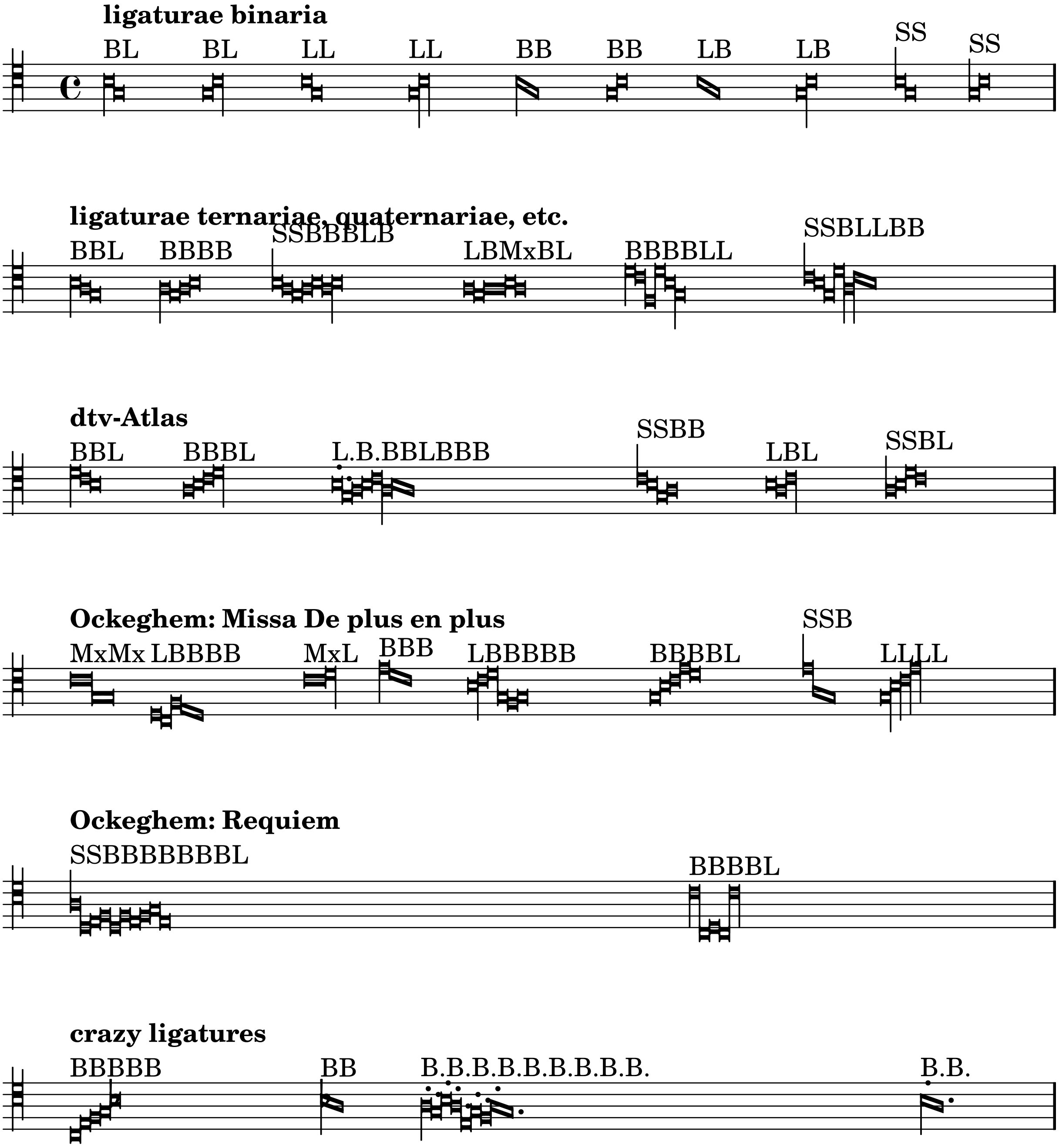
‘`measure-grouping.ly`’

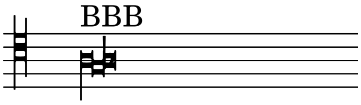
The `Measure_grouping_engraver` adds triangles and brackets above beats when the beats of a time signature are grouped.



‘`mensural-ligatures.ly`’

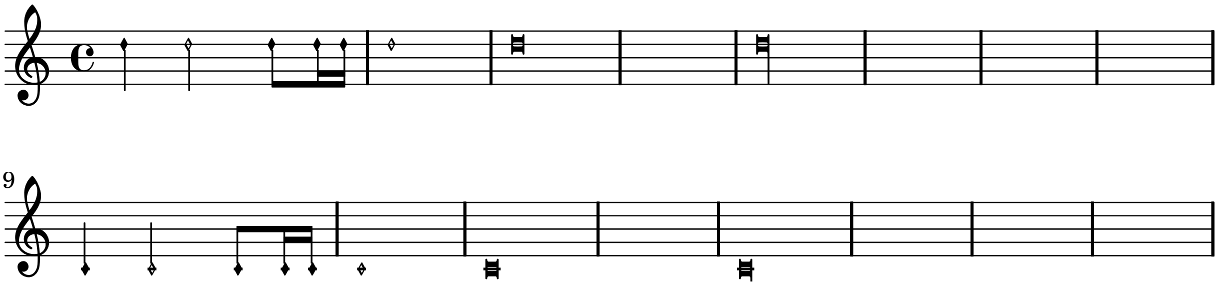
Mensural ligatures show different shapes, depending on the rhythmical pattern and direction of the melody line.





'`mensural.ly`'

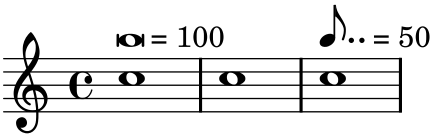
There is limited support for mensural notation: note head shapes are available. Mensural stems are centered on the note heads, both for up and down stems.



'`metronome-marking.ly`'

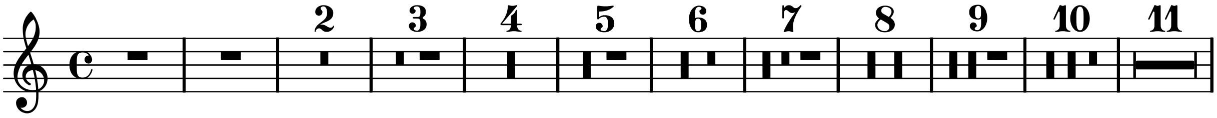
Here `\tempo` directives are printed as metronome markings.

The marking is left aligned with the time signature, if there is one.



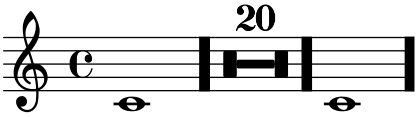
'`mm-rests2.ly`'

If `Score.skipBars` is set, the signs for four, two, and one measure rest are combined to produce the graphical representation of rests for up to 10 bars. The number of bars will be written above the sign.



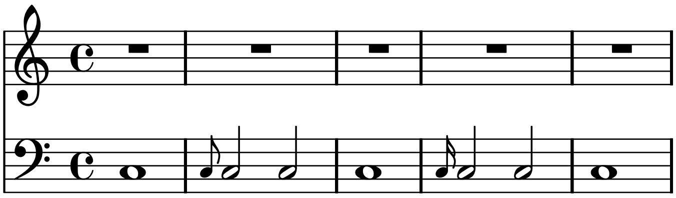
'`multi-measure-rest-center.ly`'

The multimeasure rest is centered exactly between bar lines.



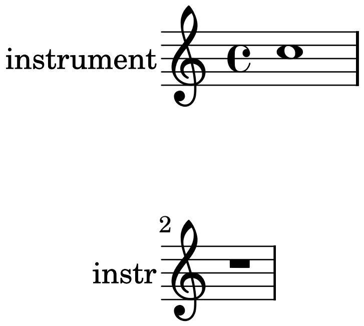
'`multi-measure-rest-grace.ly`'

Multi-measure rests are centered also in the case of grace notes.



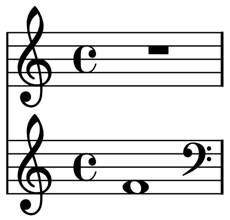
‘`multi-measure-rest-instr-name.ly`’

There are both long and short instrument names. Engraving instrument names should not be confused by the multimeasure rests.



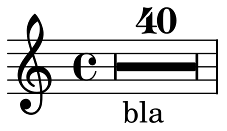
‘`multi-measure-rest-multi-staff-center.ly`’

The centering of multi-measure rests is independent on prefatory matter in other staves.



‘`multi-measure-rest-spacing.ly`’

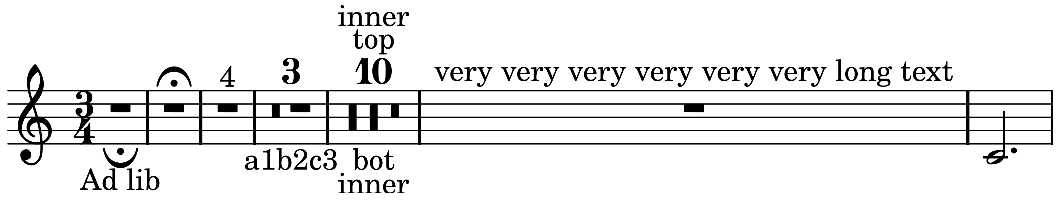
By setting texts starting with a multi-measure rest, an extra spacing column is created. This should not cause problems.



‘`multi-measure-rest-text.ly`’

Texts may be added to the multi-measure rests.

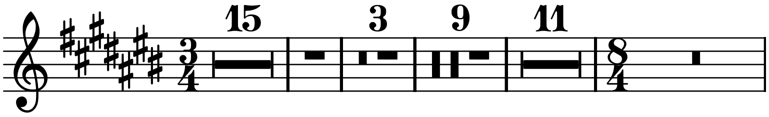
By setting the appropriate `spacing-procedure`, we can make measures stretch to accomodate wide texts.



‘`multi-measure-rest.ly`’

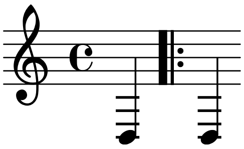
Multi-measure rests do not collide with barlines and clefs. They are not expanded when you set `Score.skipBars`. Although the multi-measure-rest is a Spanner, minimum distances are set to keep it colliding from barlines.

Rests over measures during longer than 2 wholes use breve rests. When more than 10 or more measures (tunable through `expand-limit`) are used then a different symbol is used.



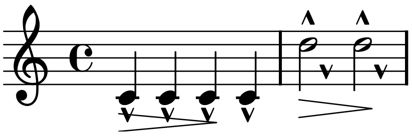
‘`music-function.ly`’

Music function are generic music transformation functions, which can be used to extend music syntax seamlessly. Here we demonstrate a `\myBar` function, which works similar to `\bar`, but is implemented completely in Scheme.



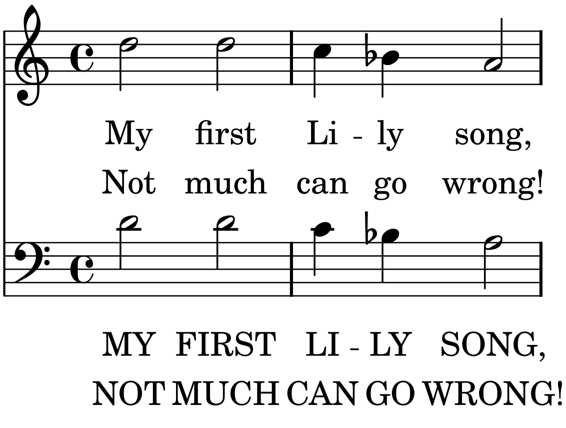
‘`music-map.ly`’

With `music-map`, you can apply functions operating on a single piece of music to an entire music expression. In this example, the the function `notes-to-skip` changes a note to a skip. When applied to an entire music expression in the 1st measure, the scripts and dynamics are left over. These are put onto the 2nd measure.



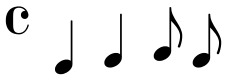
‘`newaddlyrics.ly`’

newlyrics, multiple stanzas, multiple lyric voices.



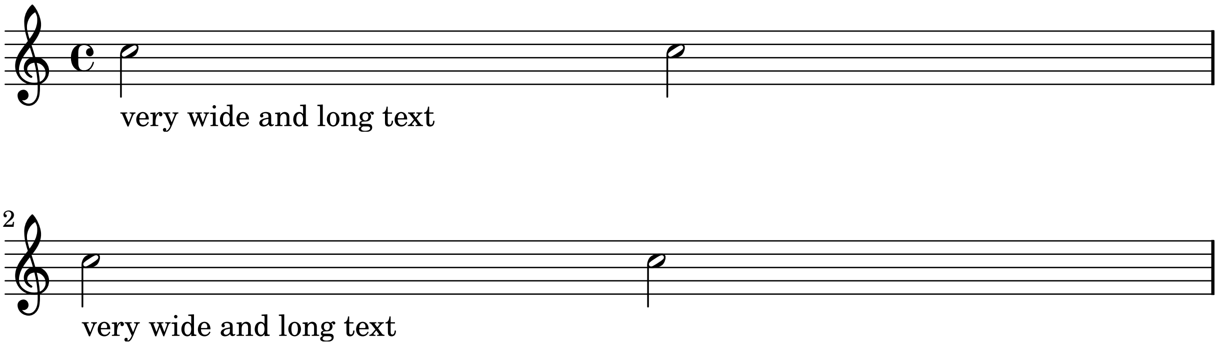
‘`no-staff.ly`’

The printing of the staff lines may be suppressed by removing the corresponding engraver.



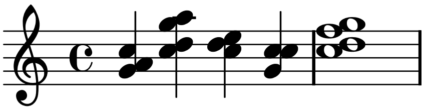
‘`non-empty-text.ly`’

By default, text is set with empty horizontal dimensions. The boolean property `no-spacing-rods` in `TextScript` is used to control the horizontal size of text.



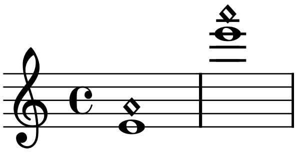
‘`note-head-chord.ly`’

Note heads are flipped on the stem to prevent collisions. It also works for whole heads that have invisible stems.



‘`note-head-harmonic-whole.ly`’

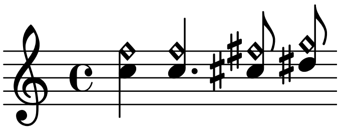
A harmonic note head must be centered if the base note is a whole note.



‘`note-head-harmonic.ly`’

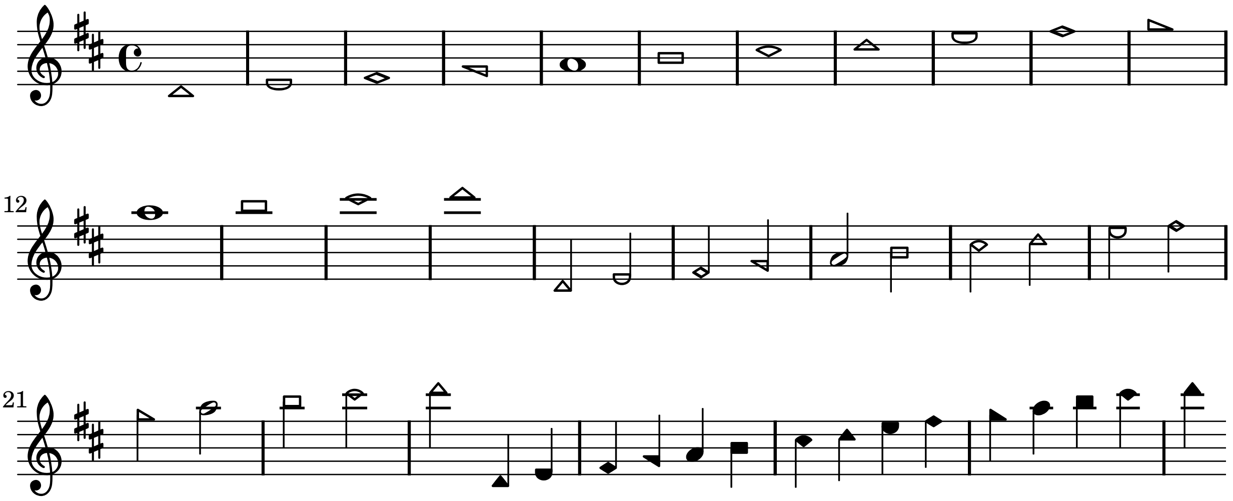
The handling of stems for harmonic notes must be completely identical to normal note heads.

Harmonic heads do not get dots. If `harmonicAccidentals` is unset, they also don’t get accidentals.



‘`note-head-solfa.ly`’

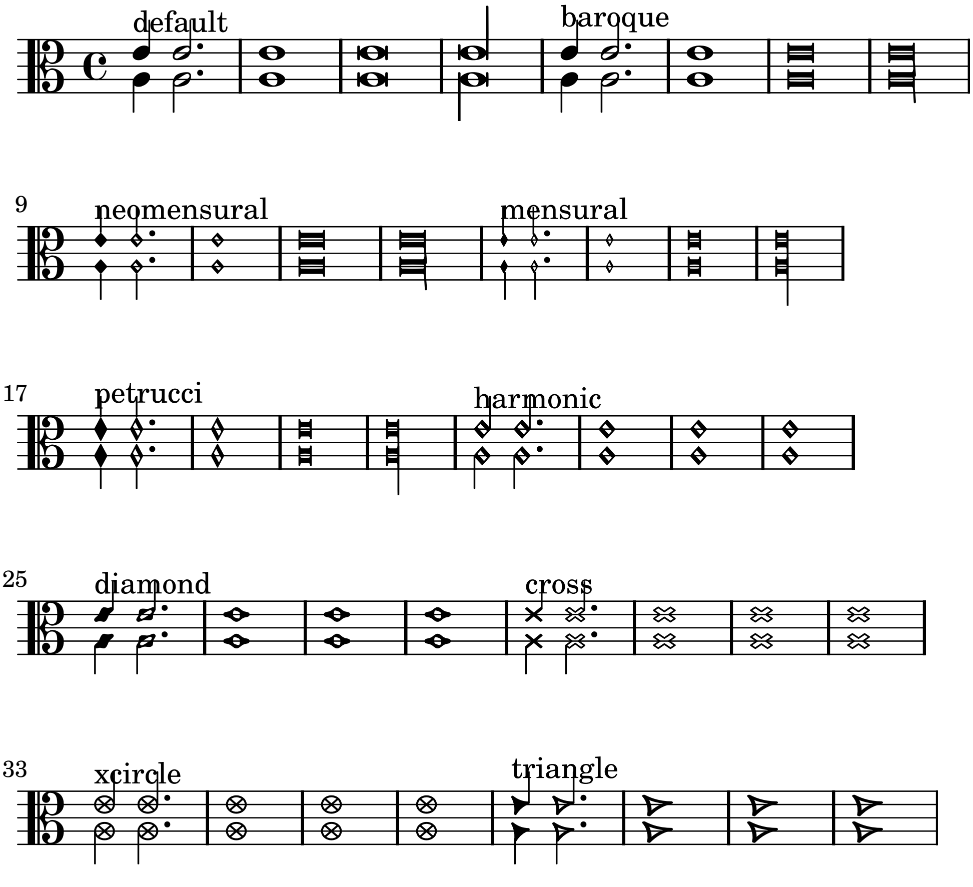
With `shapeNoteStyles`, the style of the note head is adjusted according to the step of the scale, as measured relative to the `tonic` property.

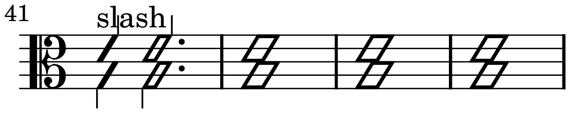


‘`note-head-style.ly`’

Note head shapes may be set from several choices. The stem endings should be adjusted according to the note head. If you want different note head styles on one stem, you must create a special context.

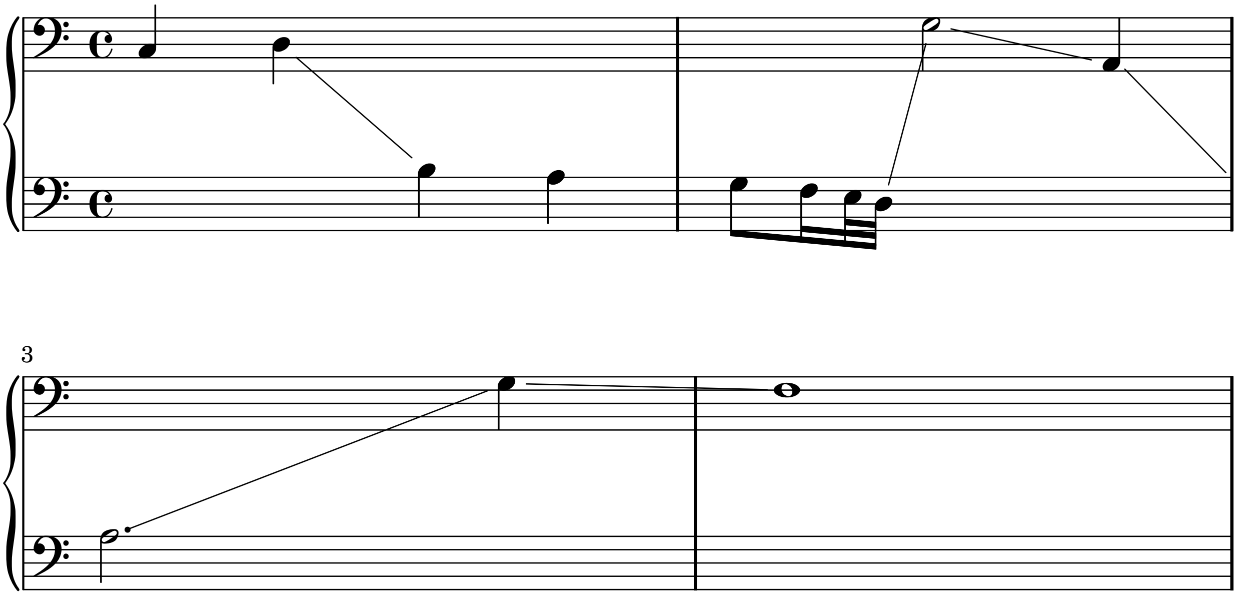
Harmonic notes have a different shape and different dimensions.





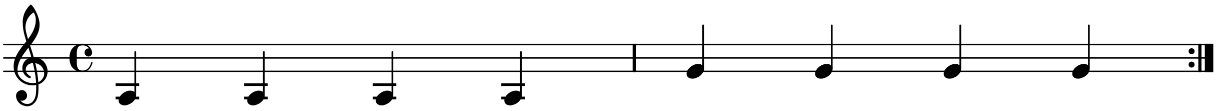
‘`note-line.ly`’

Note head lines (e.g. glissando) run between centers of the note heads.



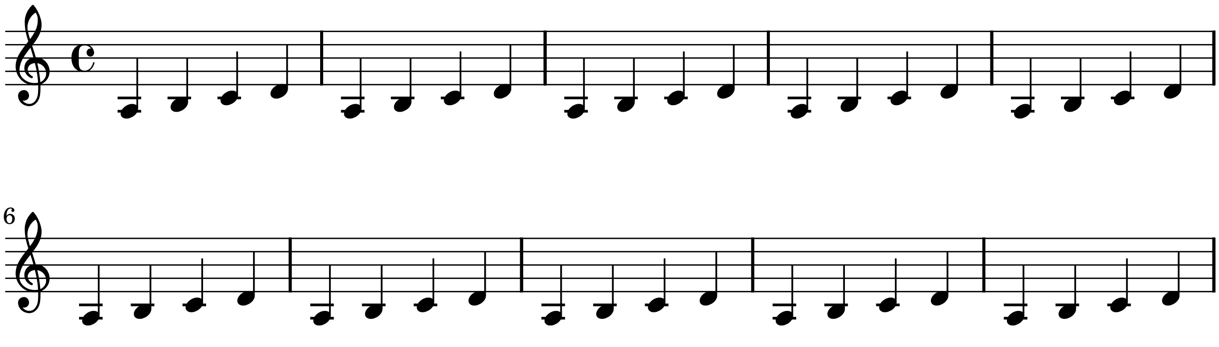
‘`number-staff-lines.ly`’

The number of stafflines of a staff can be set. Ledger lines both on note heads and rests, as well as barlines, are adjusted accordingly.



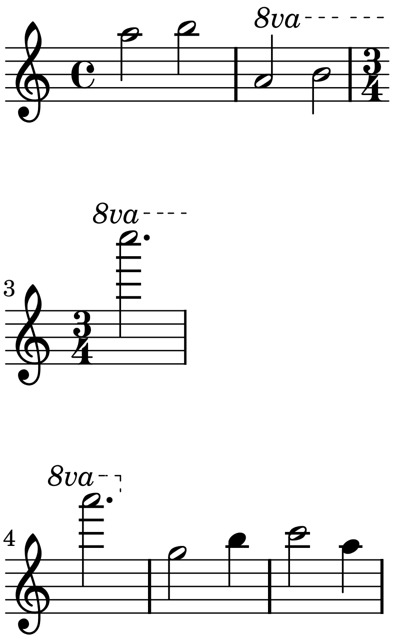
‘`optimal-page-breaking-hstretch.ly`’

The optimal page breaker will stretch the systems horizontally so that the vertical spacing will be more acceptable. The page-spacing-weight parameter controls the relative importance of vertical/horizontal spacing. Because ragged-last-bottom is on, only the first page should be horizontally stretched.



`‘ottava-broken.ly’`

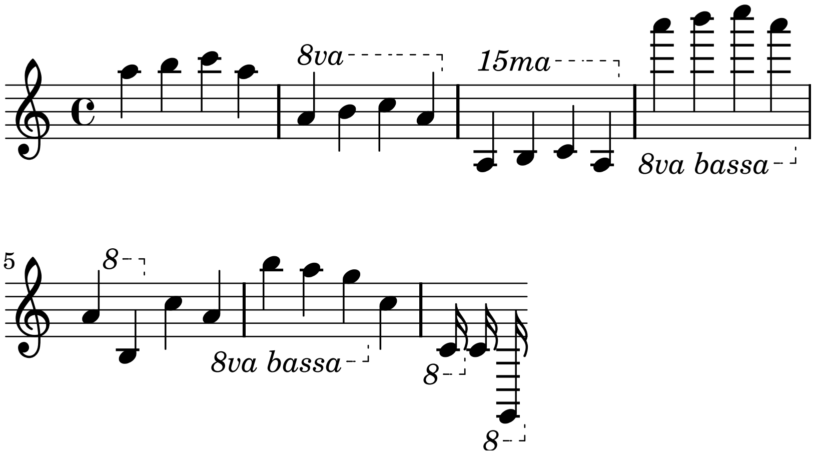
At line breaks, ottava brackets have no vertical line and their horizontal line does not stick out. The dashed line runs until the end of the line (regardless of prefatory matter).



`‘ottava.ly’`

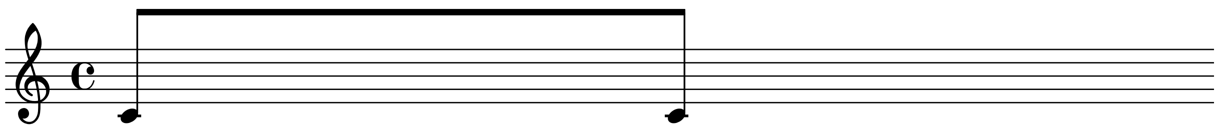
Ottava brackets are supported, through the use of the scheme function `set-octavation`.

The spanner should go below a staff for 8va bassa, and the ottavation string can be tuned with `Staff.ottavation`.



`‘override-nest.ly’`

Sublist of grob property lists may be also tuned. In the next example, the `beamed-lengths` property of the `Stem` grob is tweaked.



‘`page-breaks.ly`’

Stress optimal page breaking. This should look nice and even on 4 a6 pages.

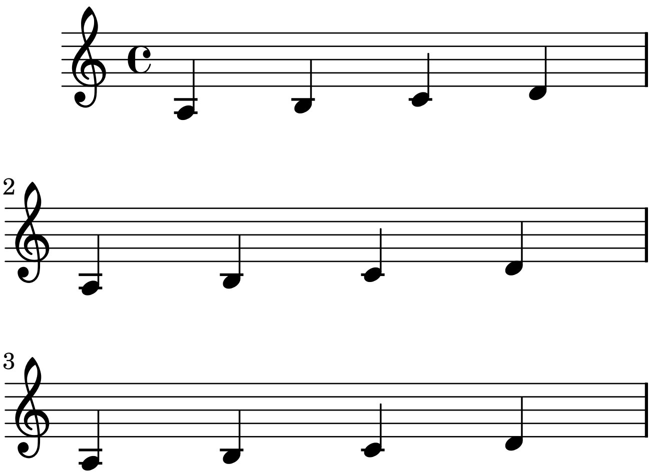
Title

(and (the) subtitle)

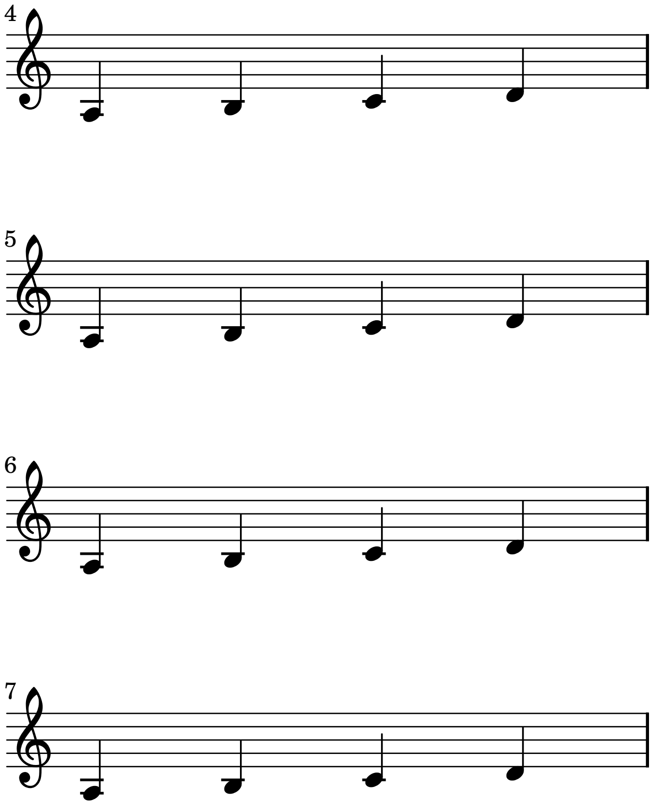
Sub sub title

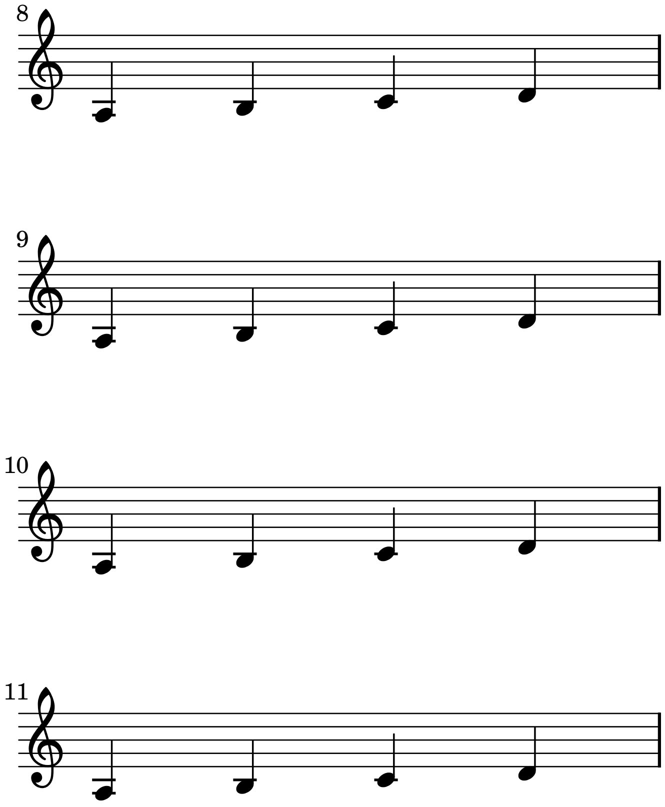
Poet **Instrument** Composer
Meter (huh?) Arranger

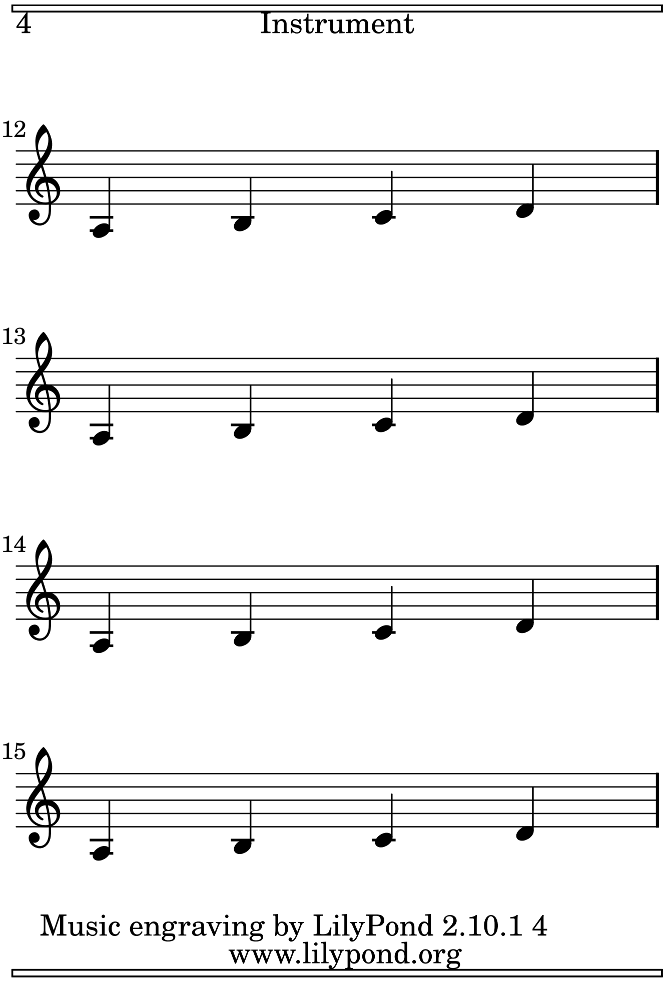
Piece opus 0



Copyright by /me


4
5
6
7


8
9
10
11

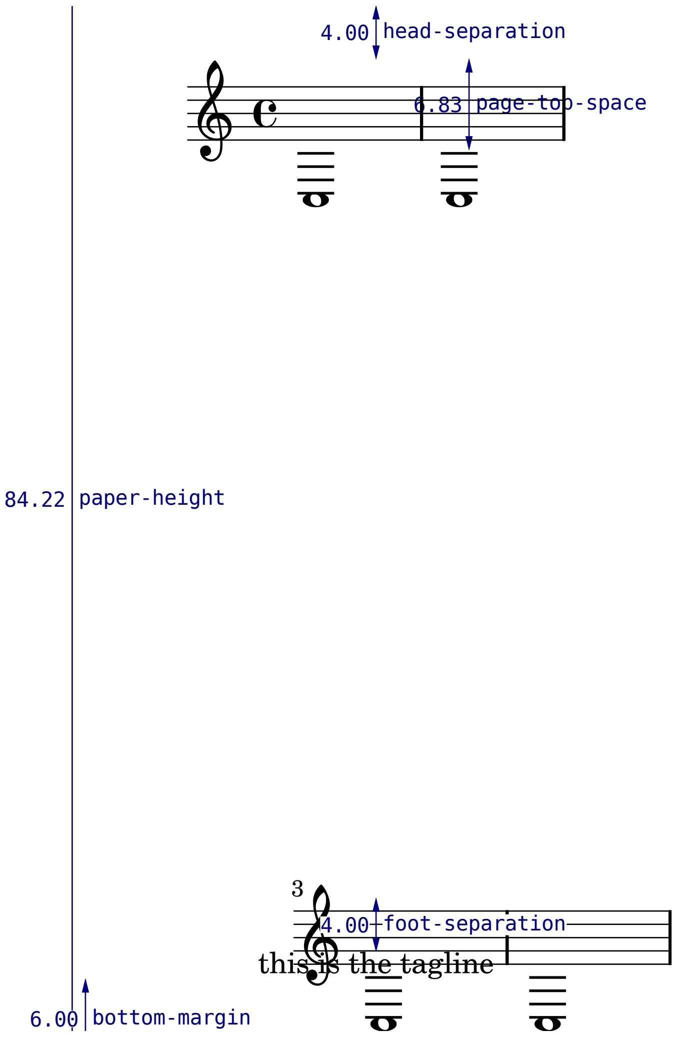


4 Instrument

Music engraving by LilyPond 2.10.1 4
www.lilypond.org

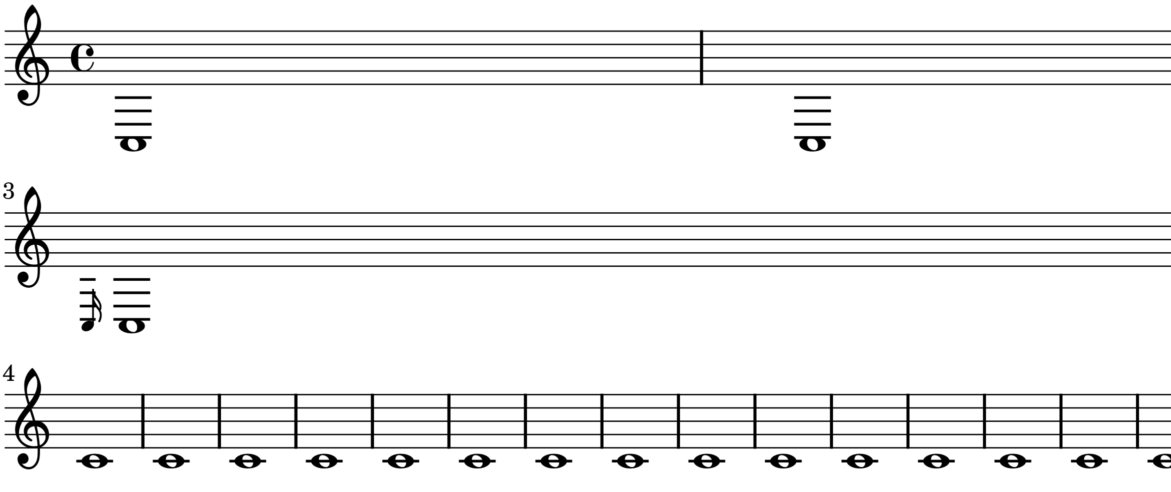
‘`page-layout-manual-position.ly`’

By setting `Y-offset` and `X-offset` for the `line-break-system-details` of `NonMusicalPaperColumn`, systems may be placed absolutely on the printable area of the page.



‘`page-layout-twopass.ly`’

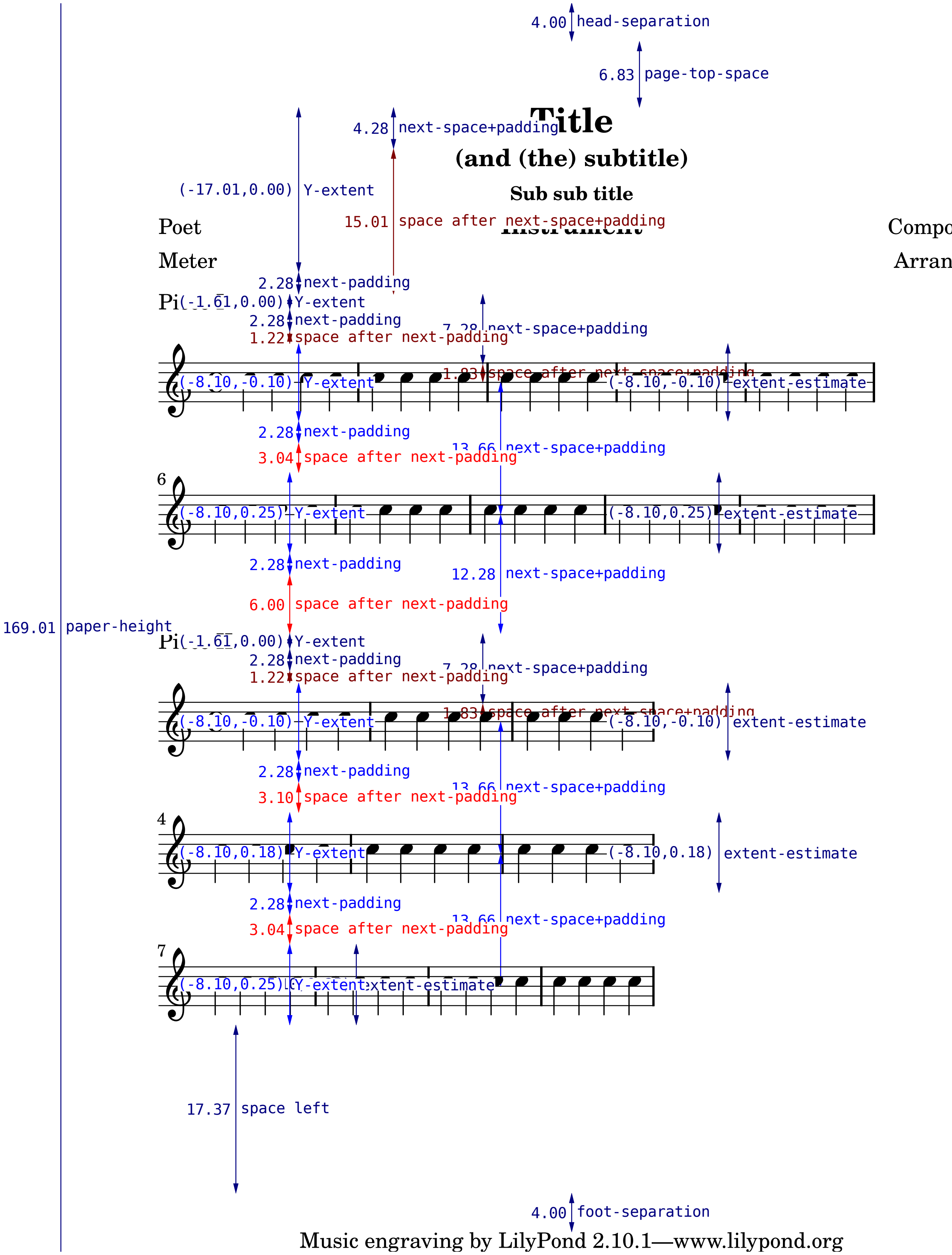
Page breaking details can be stored for later reference.


3
4

‘`page-layout.ly`’

This shows how different settings on \paper modify the general page layout. Basically \paper will set the values for the whole paper while \layout for each \score block.

This file is best viewed outside the collated files document.

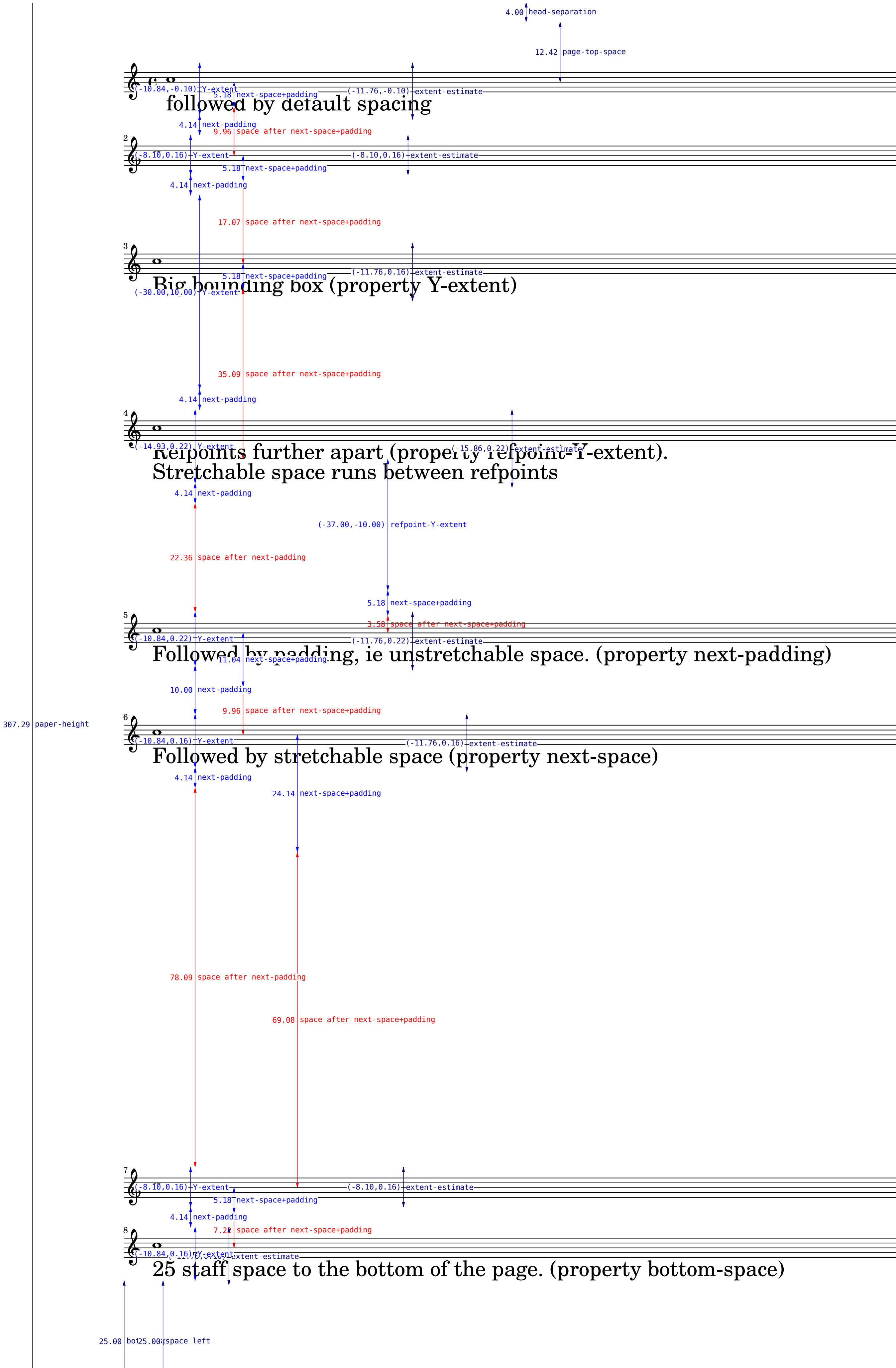


'`page-spacing.ly`'

By setting properties in NonMusicalPaperColumn, vertical spacing of page layout can be adjusted.

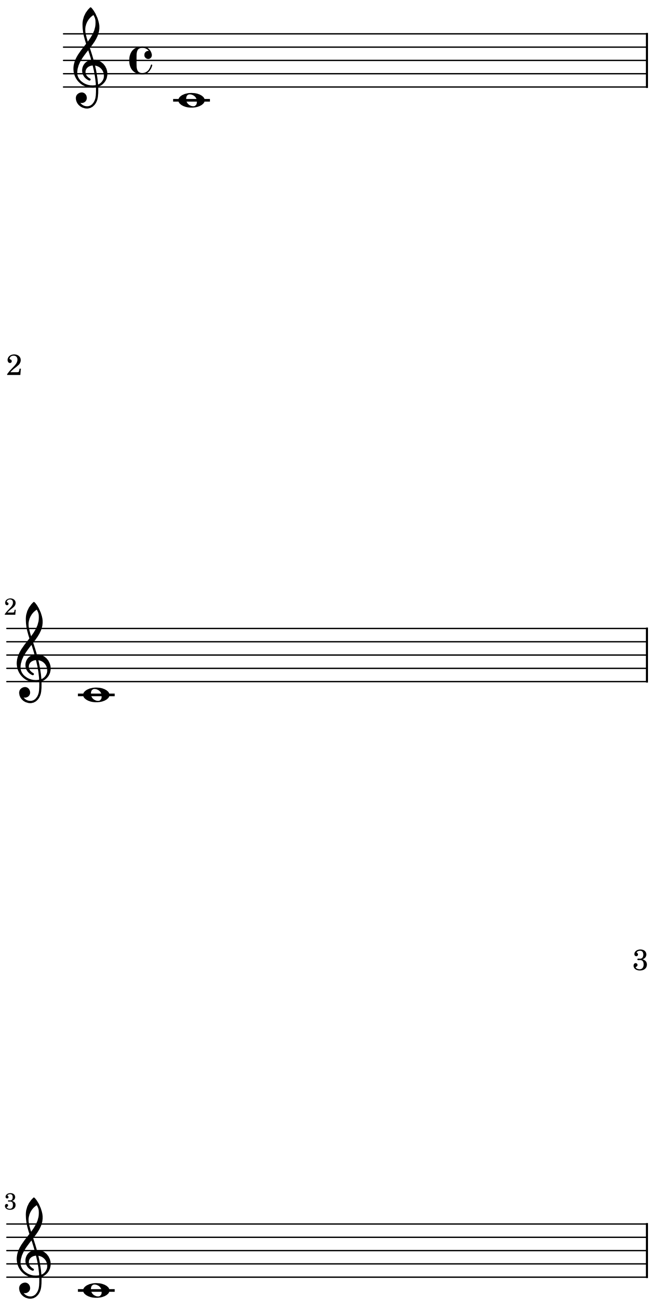
For technical reasons, `overrideProperty` has to be used for setting properties on individual object. `\override` may still be used for global overrides.

By setting `annotate-spacing`, we can see the effect of each property.

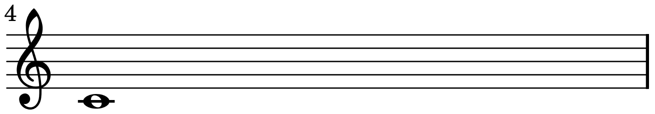

4.00 head-separation
12.42 page-top-space
(-10.84,-0.10) Y-extent
5.18 next-space+padding
(-11.76,-0.10) extent-estimate
followed by default spacing
4.14 next-padding
9.96 space after next-space+padding
2
(-8.10,0.16) Y-extent
(-8.10,0.16) extent-estimate
5.18 next-space+padding
4.14 next-padding
17.07 space after next-space+padding
3
5.18 next-space+padding
(-11.76,0.16) extent-estimate
Big bounding box (property Y-extent)
(-30.00,10.00) Y-extent
35.09 space after next-space+padding
4.14 next-padding
4
(-14.93,0.22) Y-extent
(-15.86,0.22) extent-estimate
Refpoints further apart (property refpoint-Y-extent).
Stretchable space runs between refpoints
4.14 next-padding
(-37.00,-10.00) refpoint-Y-extent
22.36 space after next-padding
5.18 next-space+padding
3.58 space after next-space+padding
5
(-10.84,0.22) Y-extent
(-11.76,0.22) extent-estimate
Followed by padding, ie unstretchable space. (property next-padding)
11.04 next-space+padding
10.00 next-padding
9.96 space after next-space+padding
307.29 paper-height
6
(-10.84,0.16) Y-extent
(-11.76,0.16) extent-estimate
Followed by stretchable space (property next-space)
4.14 next-padding
24.14 next-space+padding
78.09 space after next-padding
69.08 space after next-space+padding
7
(-8.10,0.16) Y-extent
(-8.10,0.16) extent-estimate
5.18 next-space+padding
4.14 next-padding
7.22 space after next-space+padding
8
(-10.84,0.16) Y-extent
extent-estimate
25 staff space to the bottom of the page. (property bottom-space)
25.00 bo
25.00 space left

‘`page-top-space.ly`’

By setting `page-top-space`, the Y position of the first system can be forced to be uniform.



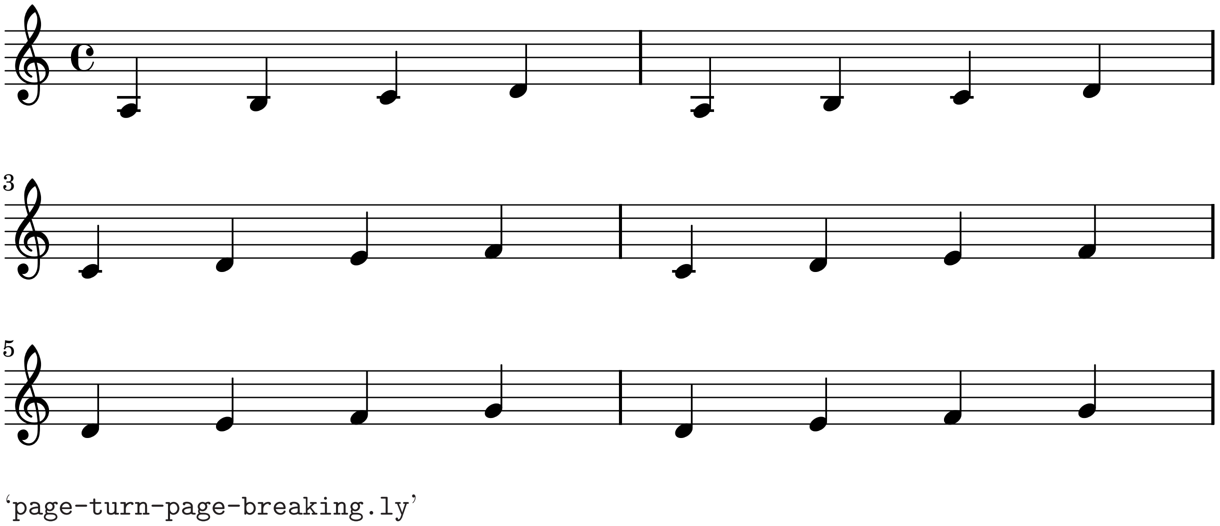
bla



Music engraving by LilyPond 2.10.1—www.lilypond.org

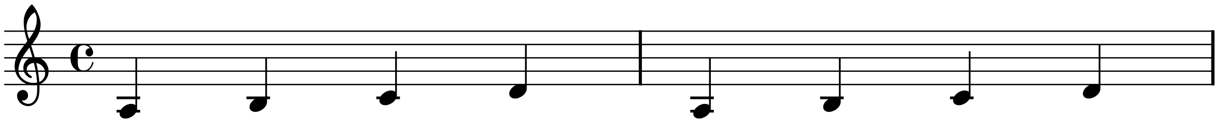
`‘page-turn-page-breaking-badturns.ly’`

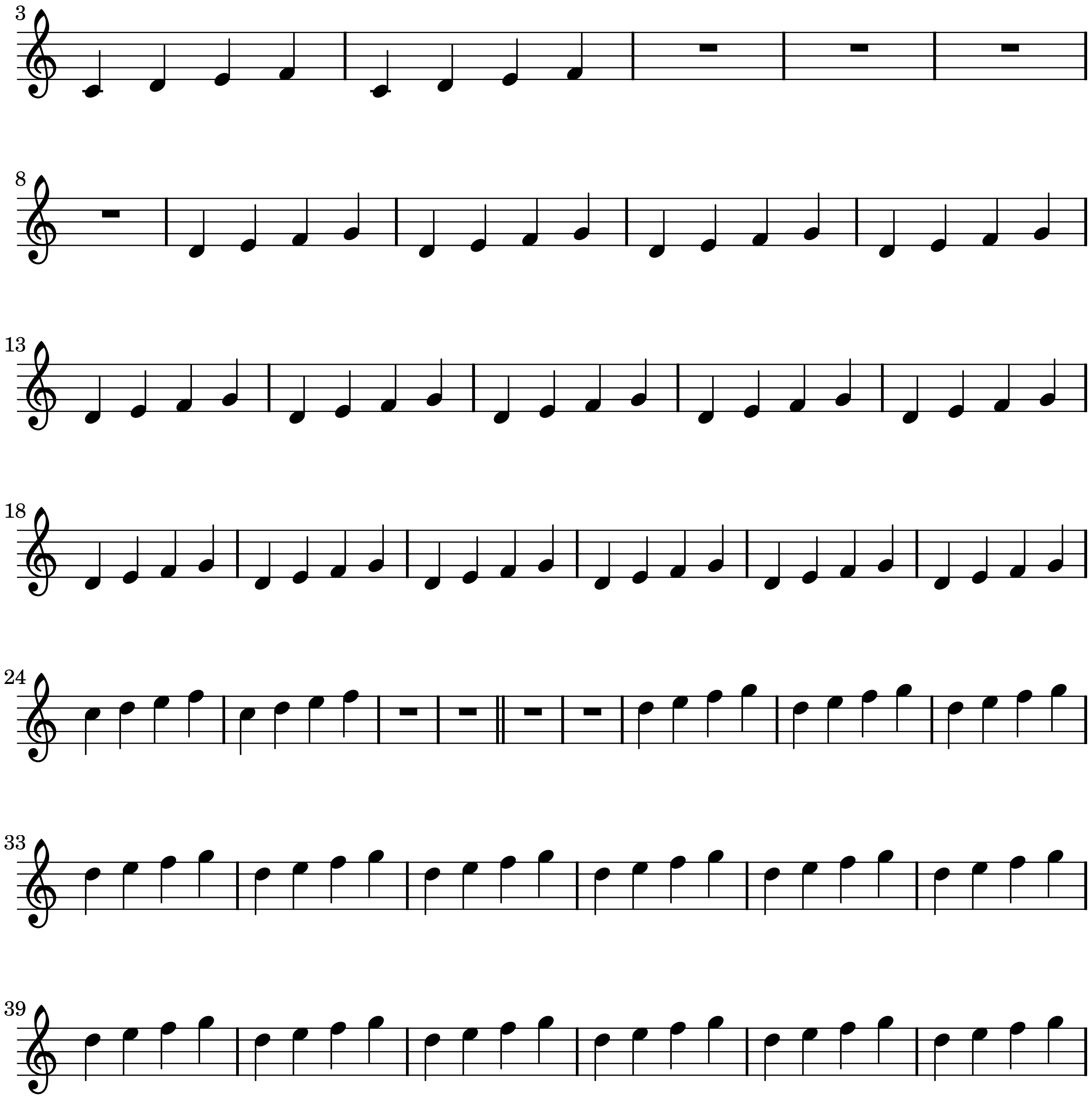
If there are no good places to have a page turn, the optimal-breaker will just have to recover gracefully. This should appear on 3 pages.



`‘page-turn-page-breaking.ly’`

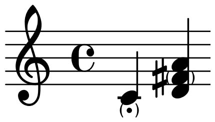
The page-turn breaker will put a page turn after a rest unless there is a ’special’ barline within the rest, in which case the turn will go after the special barline.





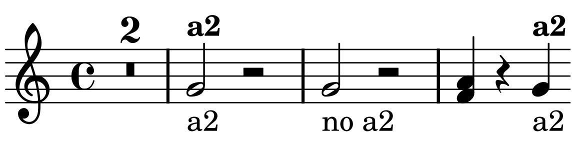
‘`parenthesize.ly`’

The parenthesize function is a special tweak that encloses objects in parentheses. The associated grob is `Score.ParenthesesItem`.



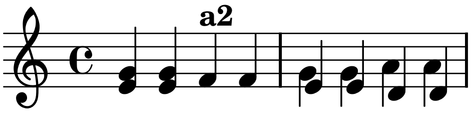
‘`part-combine-a2.ly`’

The a2 string is printed only on notes (i.e. not on rests), and only after chords, solo or polyphony.



‘`part-combine-cross.ly`’

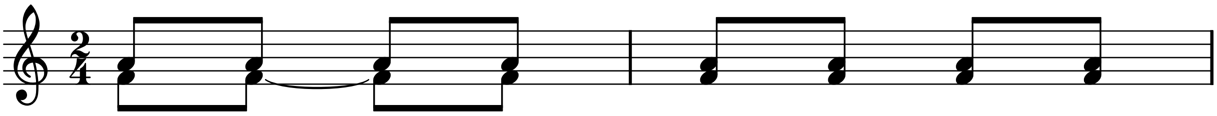
The part combiner stays apart for crossing voices.



‘`part-combine-global.ly`’

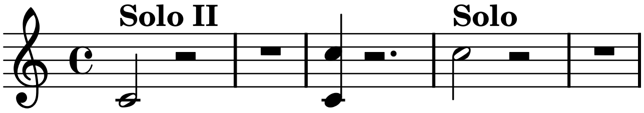
The analysis of the part combiner is non-local: in the following example, the decision for using separate voices in the 1st measure is made on the 2nd note, but influences the 1st note.

In the 2nd measure, the pattern without the tie, leads to combined voices.



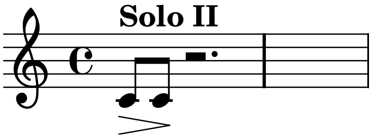
‘`part-combine-mmrest-after-solo.ly`’

Multimeasure rests are printed after solos, both for solo1 and for solo2.



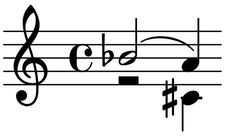
‘`part-combine-solo-end.ly`’

SOLO is printed even if the solo voice ends before the other one. Unfortunately, the multi-rest of the 1st voice (which is 2 bars longer than the 2nd voice) does not get printed.



‘`part-combine-solo-global.ly`’

In this example, solo1 should not be printed over the 1st note, because of the slur which is present from the one-voice to the two-voice situation.

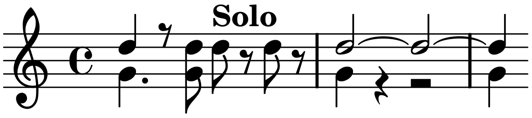


‘`part-combine-solo.ly`’

A solo string can only be printed when a note starts. Hence, in this example, there is no Solo-2 although the 2nd voice has a dotted quarter, while the first voice has a rest.

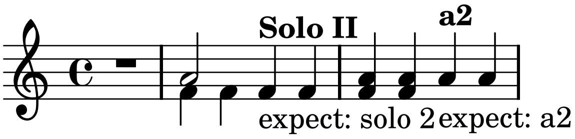
A Solo indication is only printed once; (shared) rests do not require reprinting a solo indication.

Solo 1/2 can not be used when a spanner is active, so there is no solo over any of the tied notes.



'`part-combine-text.ly`'

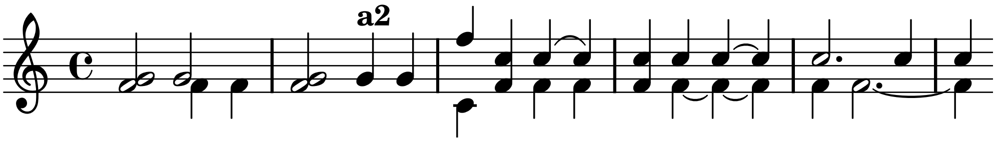
The new part combiner detects a2, solo1 and solo2, and prints i texts accordingly.



'`part-combine.ly`'

The new part combiner stays apart from:

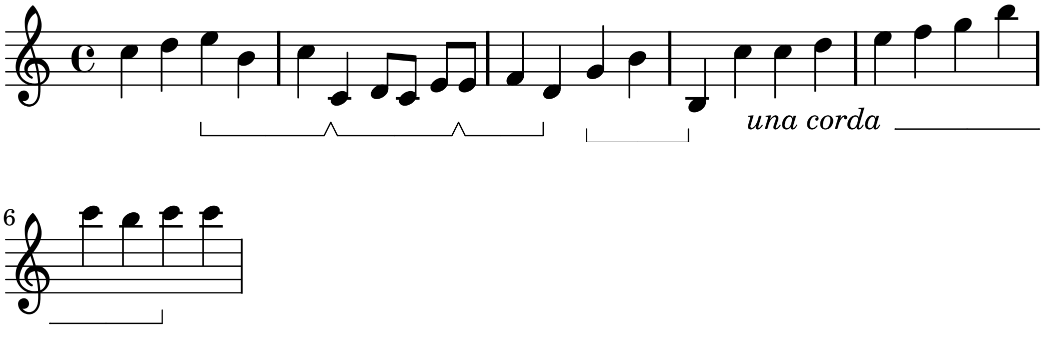
- different durations,
- different articulations (taking into account only slur/beam/tie), and
- wide pitch ranges.



'`pedal-bracket.ly`'

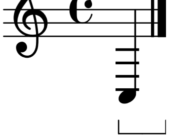
The brackets of a piano pedal should start and end at the left side of the note. If a note is shared between two brackets, these ends are flared.

At a line-break, there are no vertical endings.



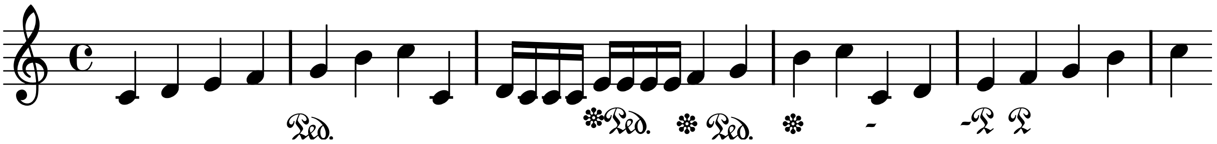
'`pedal-end.ly`'

Unterminated piano pedal brackets run to the end of the piece.



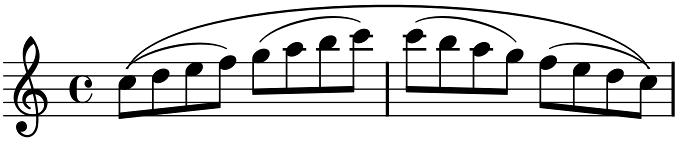
‘`pedal-ped.ly`’

The standard piano pedals style comes with Ped symbols. The pedal string can be also tuned, for example, to a shorter tilde/P variant at the end of the melody.



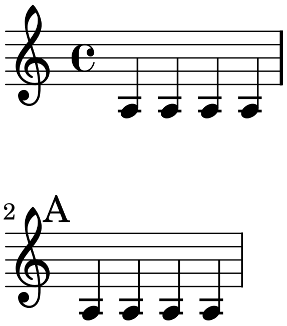
‘`phrasing-slur-slur-avoid.ly`’

`PhrasingSlur`s go over normal slurs.



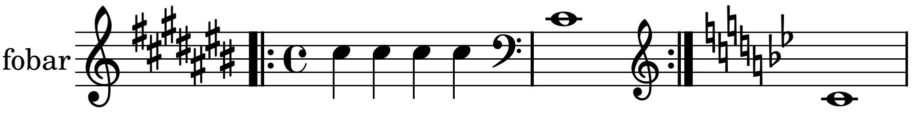
‘`prefatory-empty-spacing.ly`’

The A is atop an invisible barline. The barline, although invisible, is also translated because it is the last one of the break alignment.



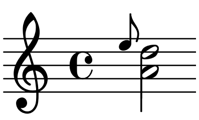
‘`prefatory-spacing-matter.ly`’

Distances between prefatory items (e.g. clef, bar, etc.) are determined by engraving standards. These distances depend on which items are combined. Mid-line, the order for clef and bar-line is different from the start of line.



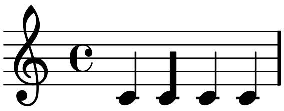
‘`property-grace-polyphony.ly`’

Property overrides and reverts from `\grace` do not interfere with the overrides and reverts from polyphony.



‘`property-once.ly`’

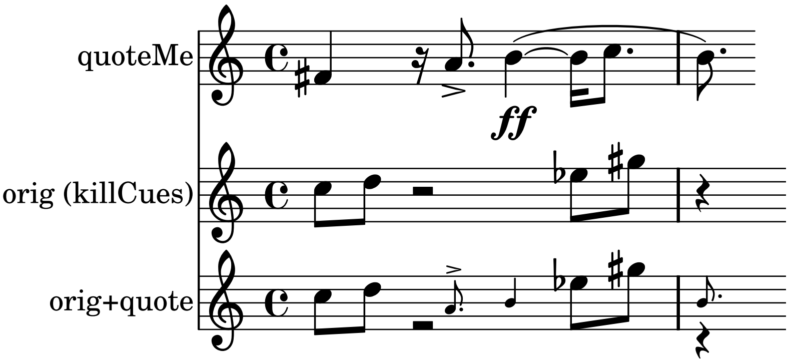
Once properties take effect during a single time step only.



‘`quote-cue-during.ly`’

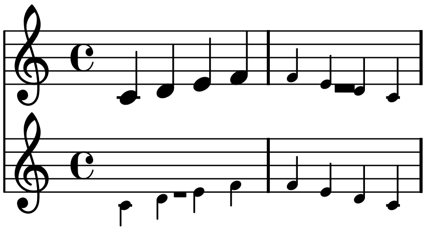
The `cueDuring` form of quotation will set stem directions on both quoted and main voice, and deliver the quoted voice in the `cue Voice`. The music function `\killCues` can remove all cue notes.

Spanners run to the end of a cue section, and are not started on the last note.



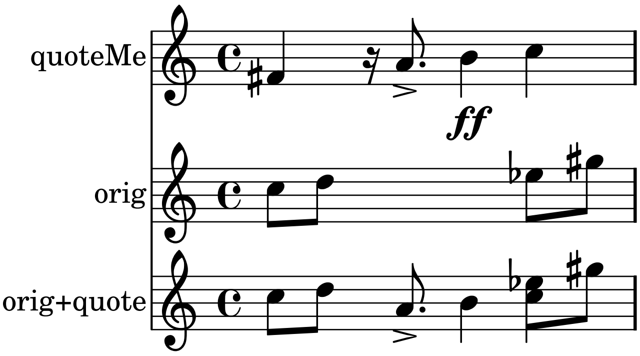
‘`quote-cyclic.ly`’

Two quoted voices may refer to each other. In this example, there are notes with each full-bar rest.



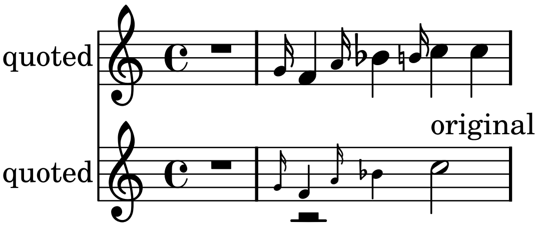
‘`quote-during.ly`’

With `\cueDuring` and `\quoteDuring`, fragments of previously entered music may be quoted. `quotedEventTypes` will determines what things are quoted. In this example, a 16th rests is not quoted, since `rest-event` is not in `quotedEventTypes`.



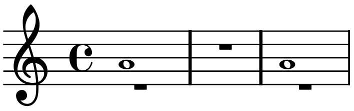
'`quote-grace.ly`'

Quotes may contain grace notes. The grace note leading up to an unquoted note is not quoted.



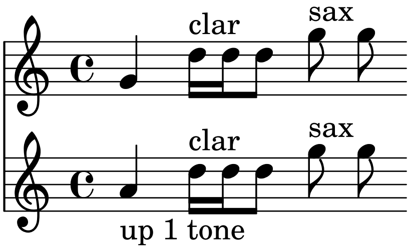
'`quote-tie.ly`'

Voices from different cues must not be tied together. In this example, the first note has a tie. This note should not be tied to the 2nd note.



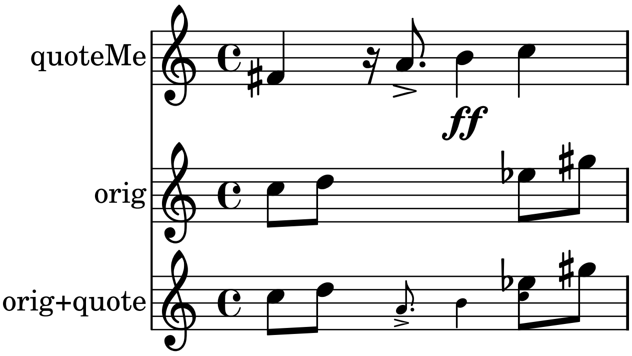
'`quote-transposition.ly`'

Quotations take into account the transposition of both source and target. In this example, all instruments play sounding central C, the target is a instrument in F. The target part may be `\transpose`d. In this case, all the pitches (including the quoted ones) will transposed as well.



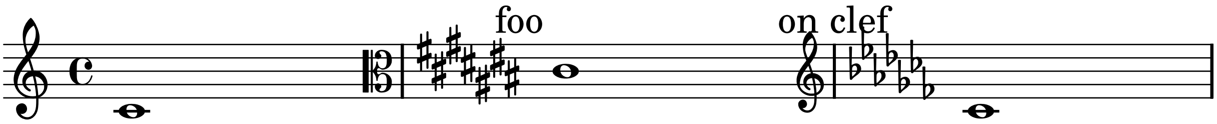
'`quote.ly`'

With `\quote`, fragments of previously entered music may be quoted. `quotedEventTypes` will determines what things are quoted. In this example, a 16th rests is not quoted, since `rest-event` is not in `quotedEventTypes`.



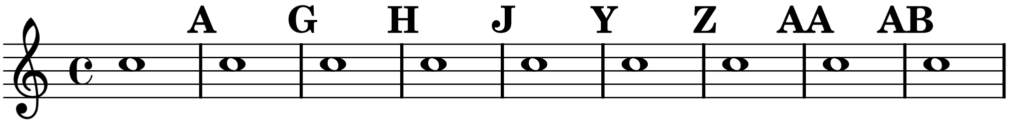
'`rehearsal-mark-align.ly`'

The rehearsal mark is put on top a breakable symbol, according to the value of `break-align-symbol` value of the `RehearsalMark`. The same holds for `BarNumber` grobs.



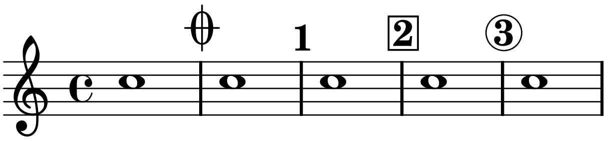
'`rehearsal-mark-letter.ly`'

Rehearsal marks in letter style: the I is skipped, and after Z, double letters are used. The mark may be set with `\mark NUMBER`, or with `Score.rehearsalMark`.



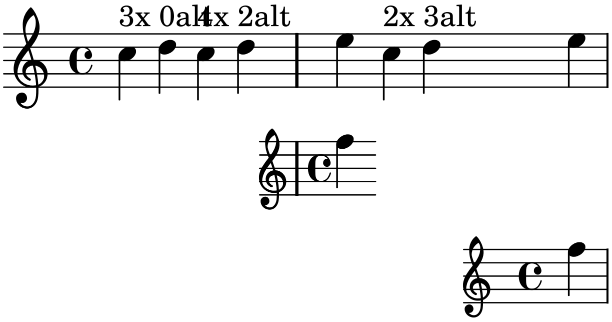
'`rehearsal-mark-number.ly`'

Marks can be printed as numbers. By setting `markFormatter` we may choose a different style of mark printing. Also, marks can be specified manually, with a markup argument.



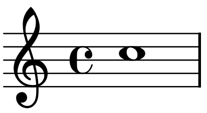
'`repeat-fold.ly`'

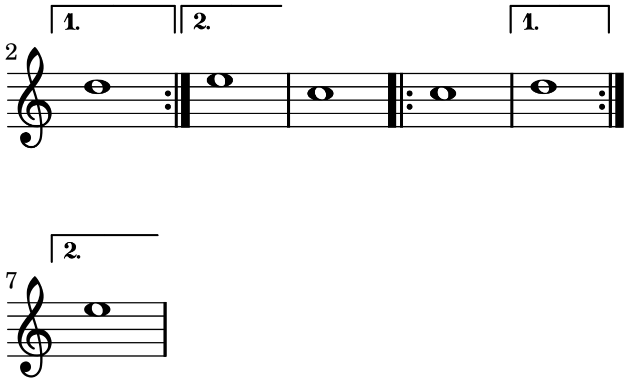
Folded repeat may not make sense without alternatives, and there should not be more alternatives than repeats.



'`repeat-line-break.ly`'

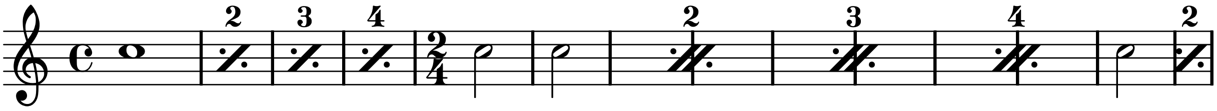
Across linebreaks, the left edge of a first and second alternative bracket should be equal.





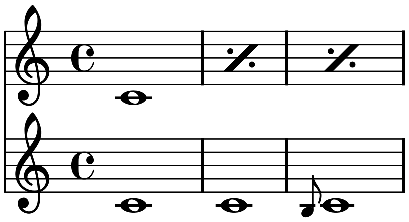
‘`repeat-percent-count.ly`’

Percent repeats get incremental numbers when `countPercentRepeats` is set, to indicate the repeat counts, but only if there are more than two repeats.



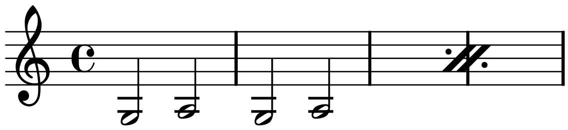
‘`repeat-percent-grace.ly`’

Percent repeats are also centered when there is a grace note in a parallel staff.



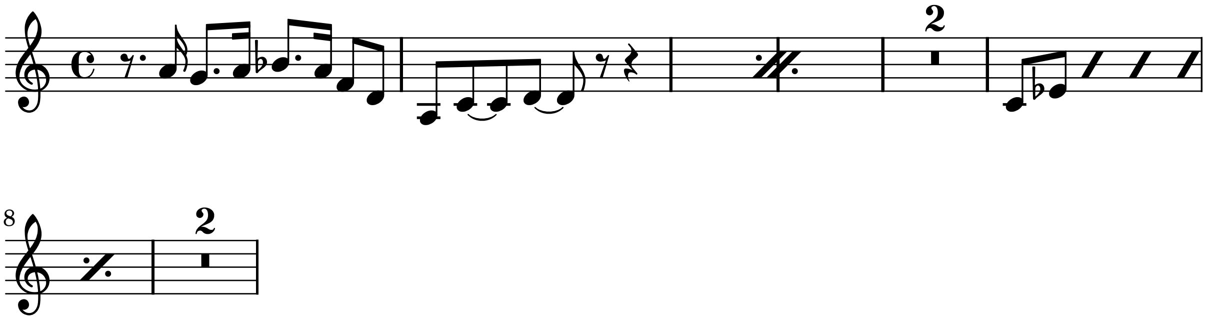
‘`repeat-percent-skipbars.ly`’

Percent repeats are not skipped, even when `skipBars` is set.



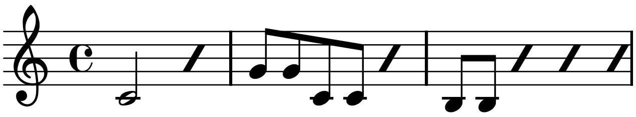
‘`repeat-percent.ly`’

Measure repeats may be nested with beat repeats.



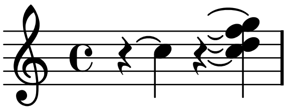
‘`repeat-slash.ly`’

Within a bar, beat repeats denote that a music snippet should be played again.



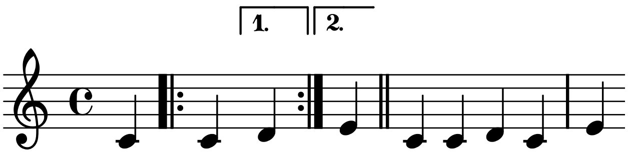
‘`repeat-tie.ly`’

Repeat ties are only connected on the right side to a note head.



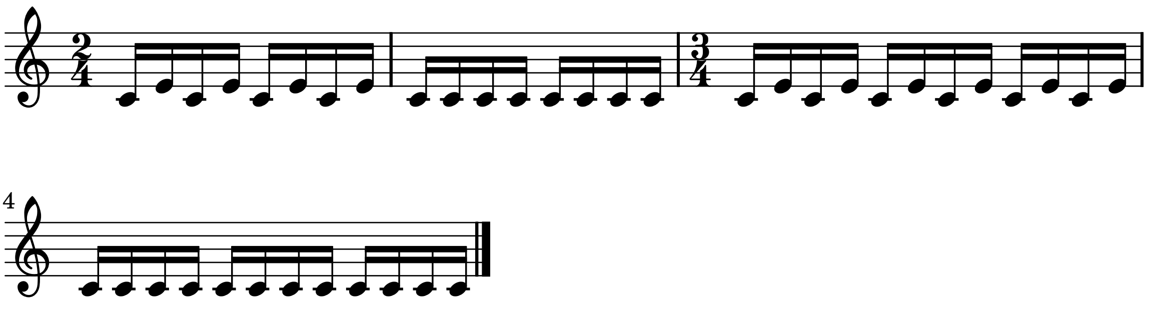
‘`repeat-unfold-all.ly`’

Volta repeats may be unfolded through the music function `\unfoldRepeats`.



‘`repeat-unfold-tremolo.ly`’

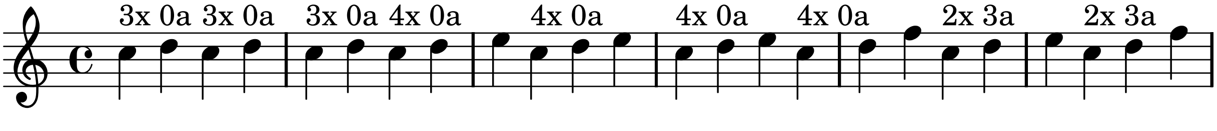
Unfolding tremolo repeats. All fragments fill one measure with 16th notes exactly.



‘`repeat-unfold.ly`’

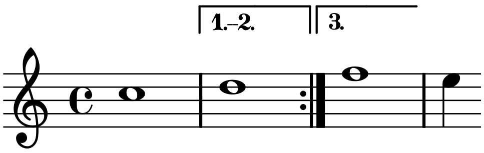
LilyPond has three modes for repeats: folded, unfolded and semi-unfolded. Unfolded repeats are fully written out. Semi unfolded repeats have the body written and all alternatives sequentially. Folded repeats have the body written and all alternatives simultaneo.ly. If the number of alternatives is larger than the repeat count, the excess alternatives are ignored. If the number of alternatives is smaller, the first alternative is multiplied to get to the number of repeats.

Unfolded behavior:



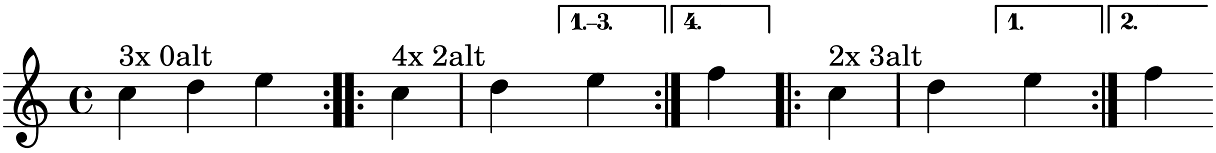
`repeat-volta-skip-alternatives.ly`

When too few alternatives are present, the first alternative is repeated, by printing a range for the 1st repeat.



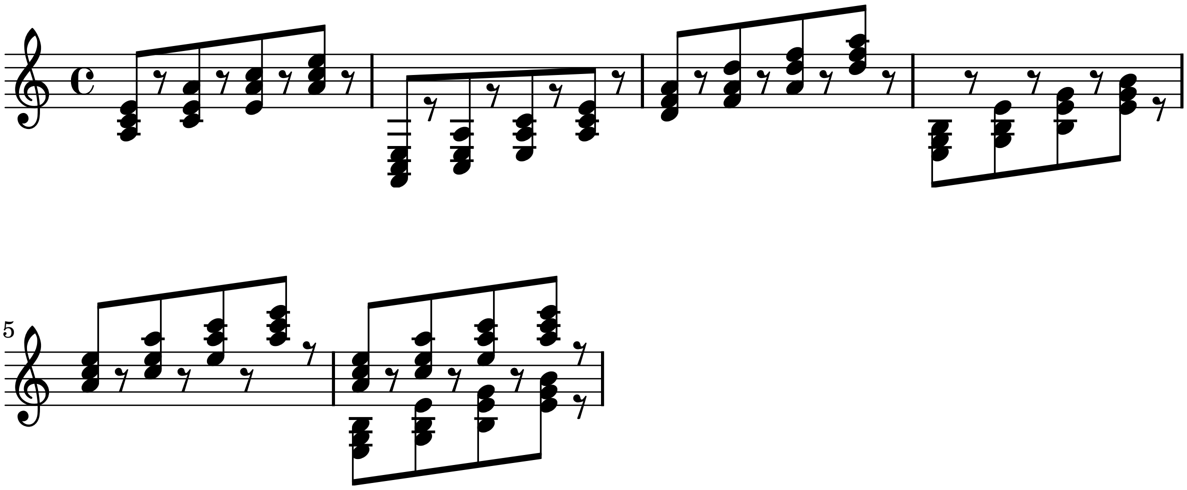
`repeat-volta.ly`

Volta (Semi folded) behavior. Voltas can start on non-barline moments. If they don't barlines should still be shown.



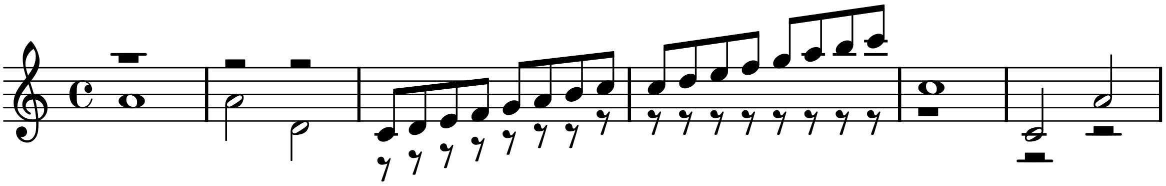
`rest-collision-beam.ly`

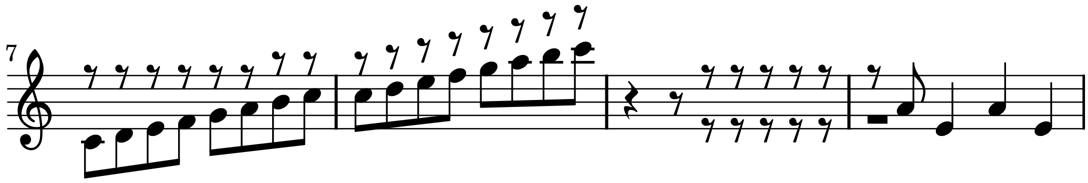
Rests under beams are only moved if necessary.



`rest-collision.ly`

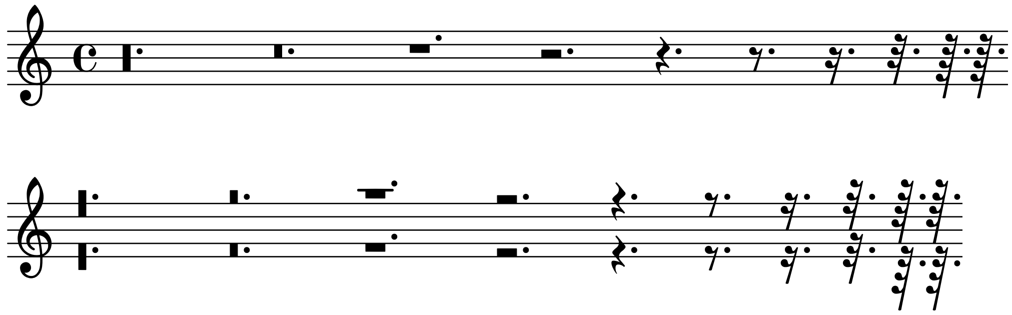
Rests should not collide with beams, stems and noteheads. Rests may be under beams. Rests should be move by integral number of spaces inside the staff, and by half spaces outside. Notice that the half and whole rests just outside the staff get ledger lines in different cases.





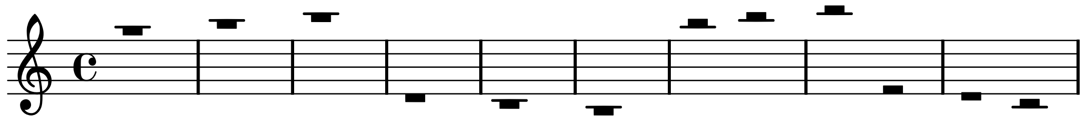
‘rest-dot-position.ly’

Dots of rests should follow the rest positions.



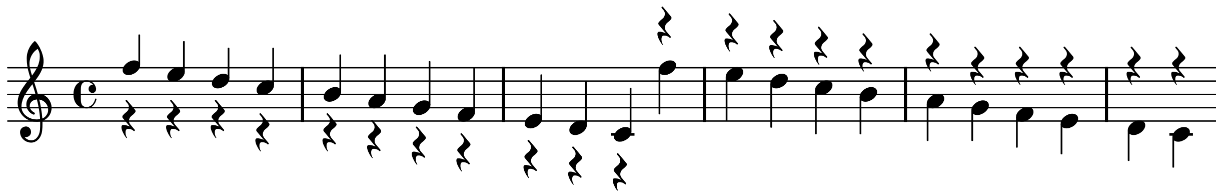
‘rest-ledger.ly’

Whole and half rests moving outside the staff should get ledger lines.



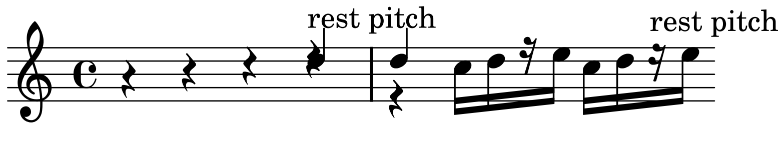
‘rest-note-collision.ly’

In rest-note collisions, the rest moves in discrete steps, and inside the staff, it moves in whole staff spaces.



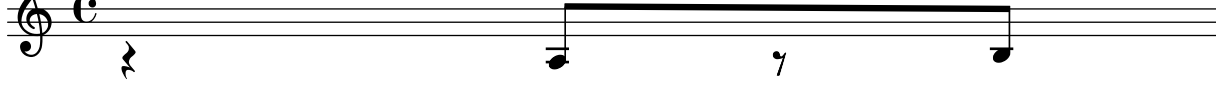
‘rest-pitch.ly’

Rests can have pitches–these will be affected by transposition and relativization. If a rest has a pitch, rest/rest and beam/rest collision resolving will leave it alone.



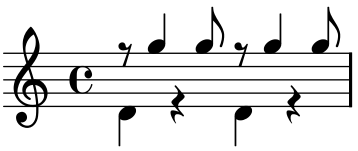
‘rest-pitched-beam.ly’

Pitched rests under beams.



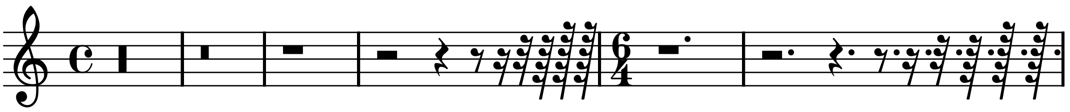
‘`rest-polyphonic.ly`’

In polyphonic situations, rests are moved down even if there is no opposite note or rest. The amount is two `staff-spaces`.



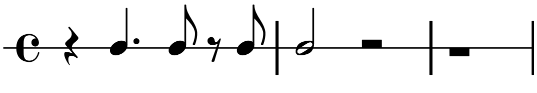
‘`rest.ly`’

There is a big variety of rests. Note that the dot of 8th, 16th and 32nd rests rest should be next to the top of the rest. All rests except the whole rest are centered on the middle staff line.



‘`rhythmic-staff.ly`’

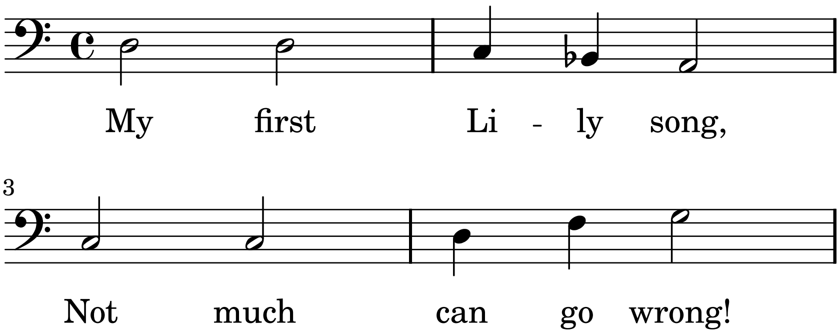
In rhythmic staves stems should go up, and bar lines have the size for a 5 line staff. The whole rest hangs from the rhythmic staff.



‘`score-text.ly`’

Markup texts are rendered above or below a score.

High up above

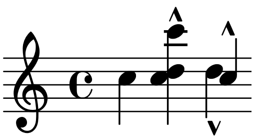


2. My next Li-ly verse
Now it's getting worse!

3. My last Li-ly text
See what will be next!

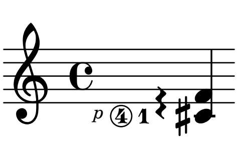
‘`script-collision.ly`’

Scripts are put on the utmost head, so they are positioned correctly when there are collisions.



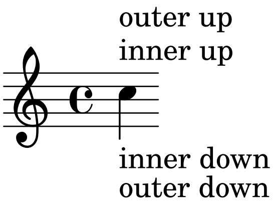
‘`script-stack-horizontal.ly`’

horizontal scripts are ordered, so they do not overlap. The order may be set with script-priority.



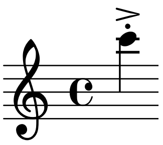
‘`script-stack-order.ly`’

Scripts can be stacked. The order is determined by a priority field, but when objects have the same priority, the input order determines the order. Objects specified first are closest to the note.



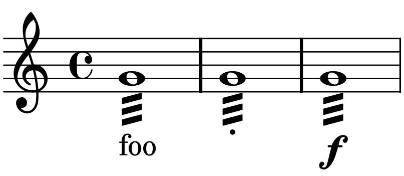
‘`script-stacked.ly`’

Scripts may be stacked.



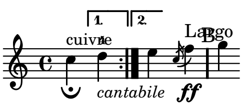
‘`script-stem-tremolo.ly`’

Scripts avoid stem tremolos even if there is no visible stem.



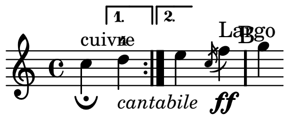
‘`size11.ly`’

Different text styles are used for various purposes.



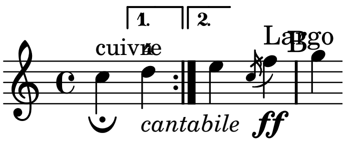
'`size13.ly`'

Different text styles are used for various purposes.



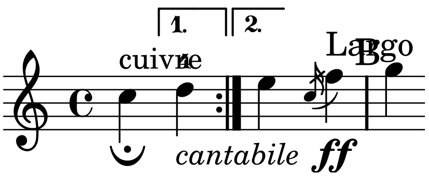
'`size16.ly`'

Different text styles are used for various purposes.



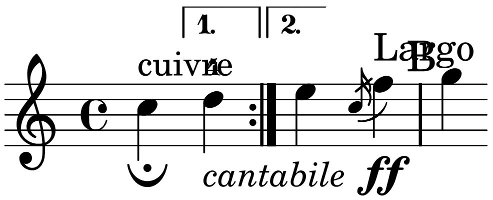
'`size20.ly`'

Different text styles are used for various purposes.



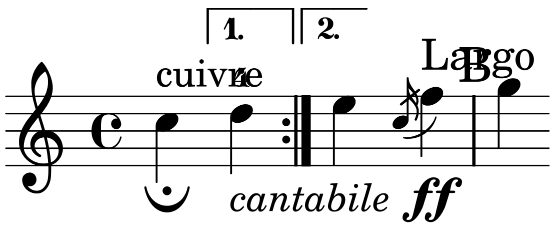
'`size23.ly`'

Different text styles are used for various purposes.



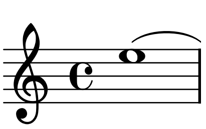
'`size26.ly`'

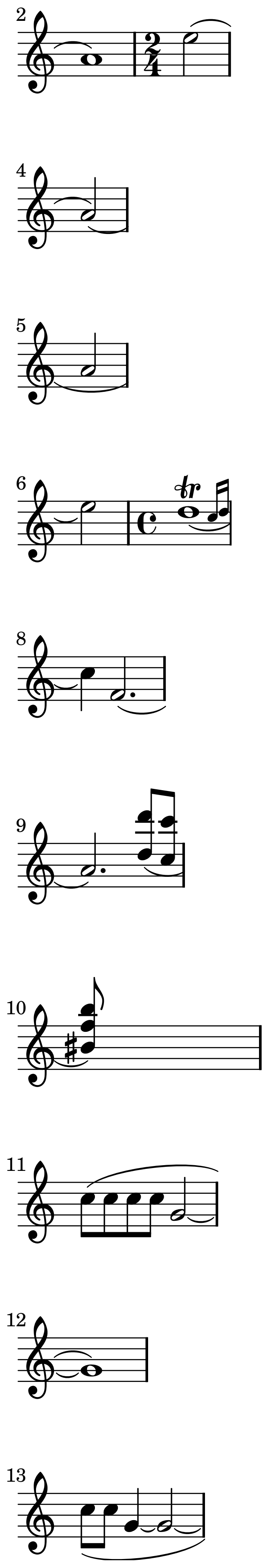
Different text styles are used for various purposes.

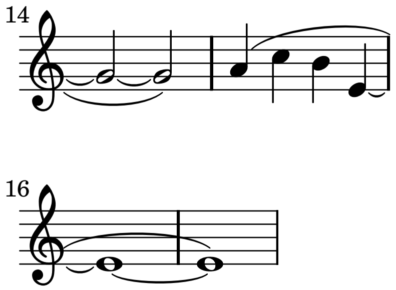


'`slur-broken-trend.ly`'

Across line breaks, slurs behave nicely. On the left, they extend to just after the preferatory matter, and on the right to the end of the staff. A slur should follow the same vertical direction it would have in unbroken state.

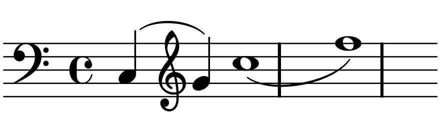



2
4
5
6
tr
8
9
10
11
12
13



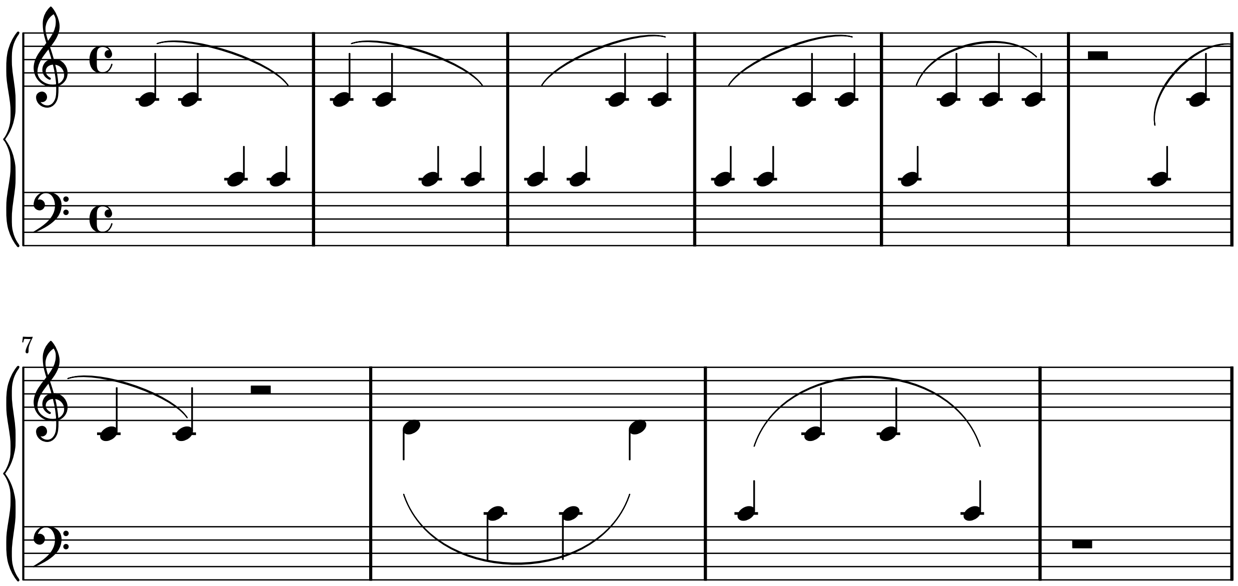
`‘slur-clef.ly’`

Slurs avoid clefs, but don’t avoid barlines.



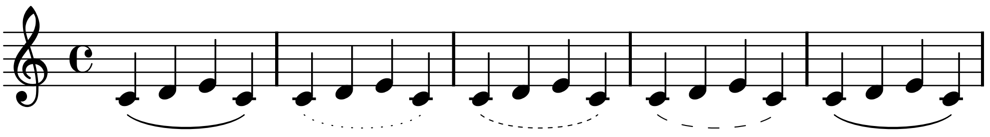
`‘slur-cross-staff.ly’`

Slurs behave decently when broken across a linebreak.



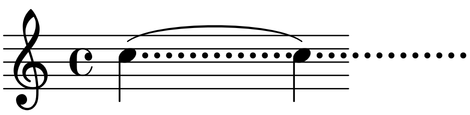
`‘slur-dash.ly’`

The appearance of slurs may be changed from solid to dotted or dashed.



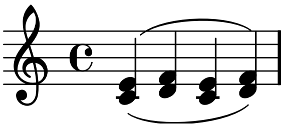
`‘slur-dots.ly’`

Slurs should not get confused by augmentation dots. With a lot of dots, the problems becomes more visible.



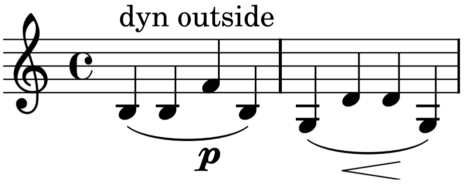
‘`slur-double.ly`’

Some composers use slurs both above and below chords. This can be typeset by setting `doubleSlurs`



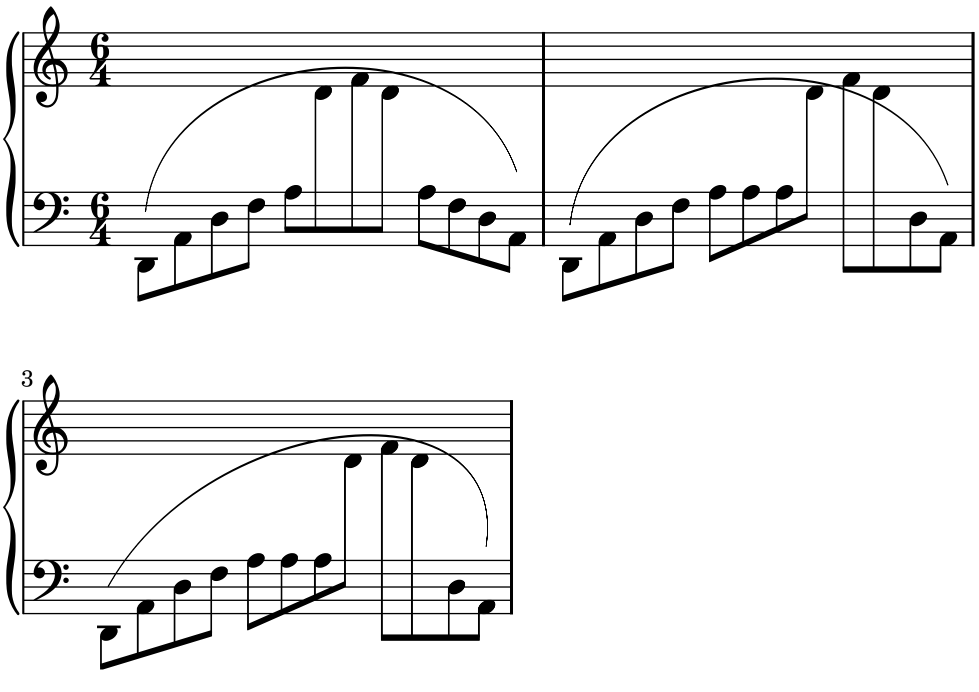
‘`slur-dynamics.ly`’

Dynamics avoid collision with slur.



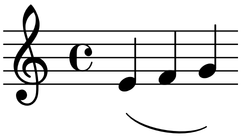
‘`slur-extreme.ly`’

Extreme slurs are scaled to fit the pattern, but only symmetrically. Asymmetric slurs are created by setting `eccentricity`.



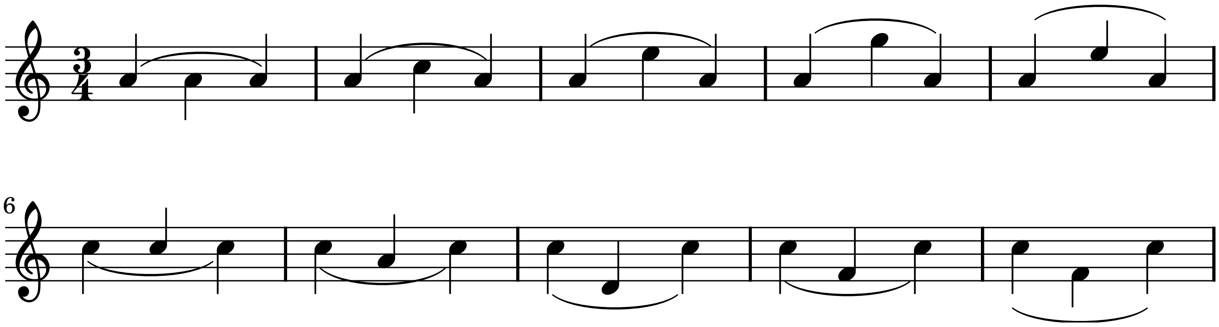
‘`slur-manual.ly`’

Setting `positions` overrides the automatic positioning of the slur. It selects the slur configuration closest to the given pair.



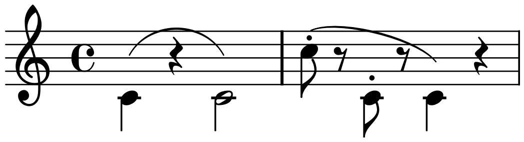
'`slur-nice.ly`'

Slurs should look nice and symmetric. The curvature may increase only to avoid noteheads, and as little as possible. Slurs never run through noteheads or stems.



'`slur-rest.ly`'

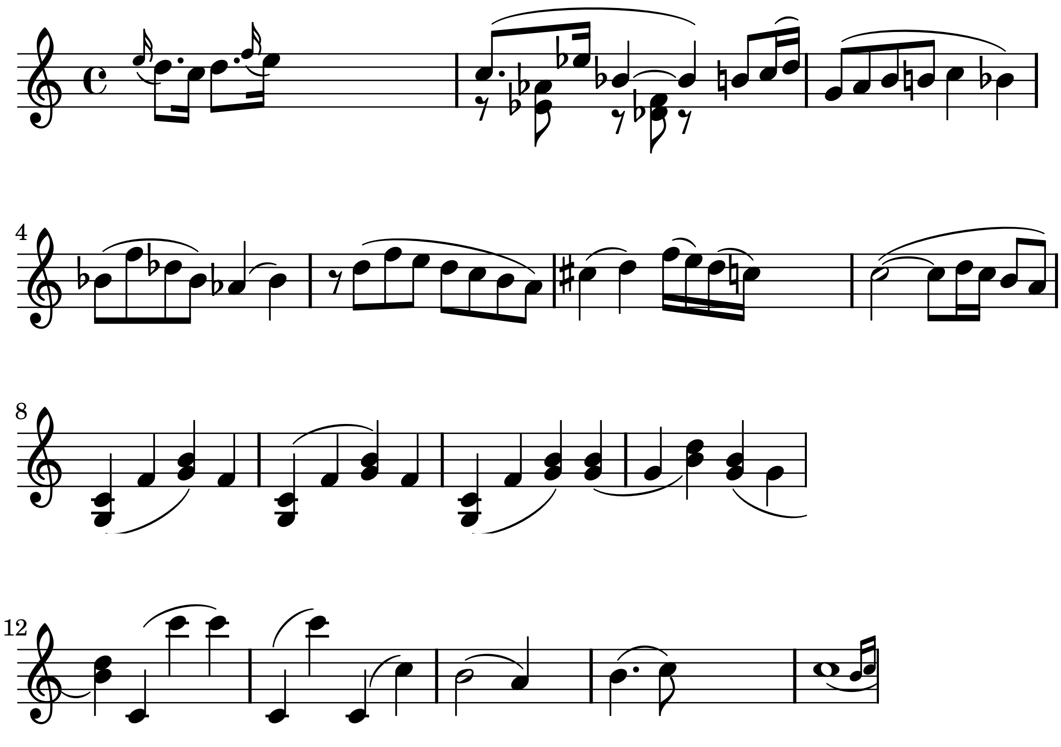
Slurs may be placed over rest. The slur will avoid colliding with the rest.

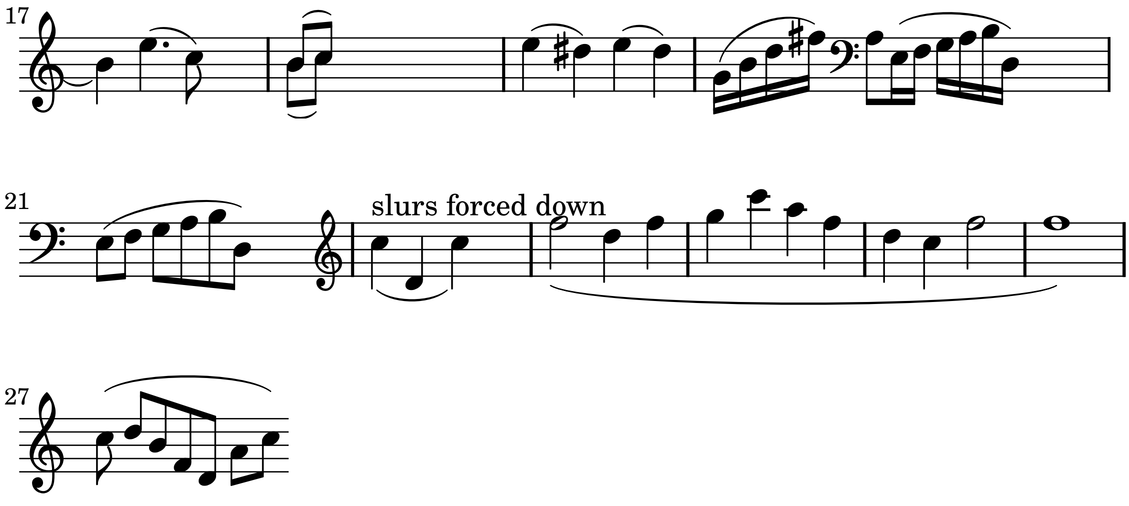


'`slur-scoring.ly`'

Slur formatting is based on scoring. A large number of slurs are generated. Each esthetic aspect gets demerits, the best configuration (with least demerits) wins. This must be tested in one big file, since changing one score parameter for one situation may affect several other situations.

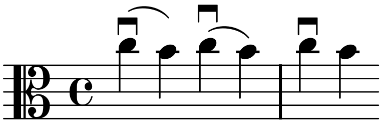
Tunable parameters are in '`scm/slur.scm`'.





‘`slur-script-inside.ly`’

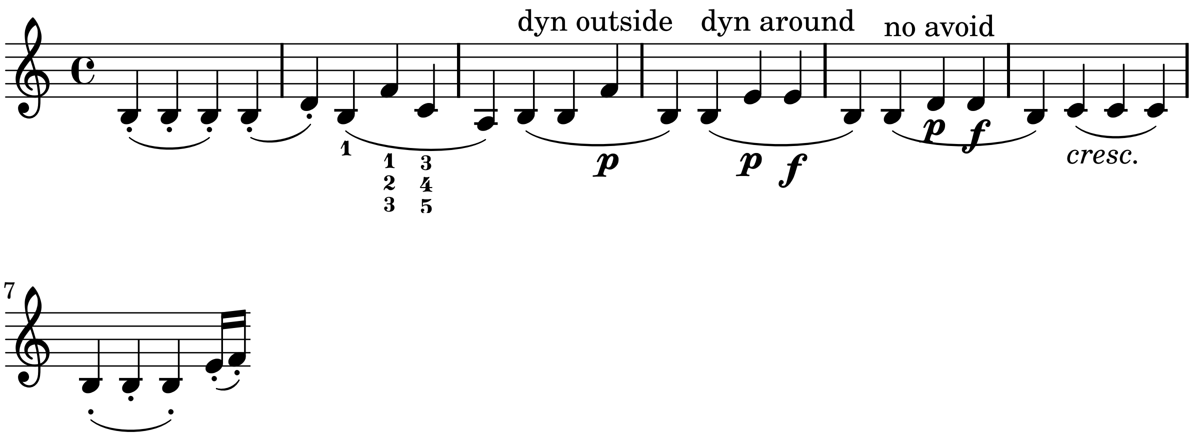
Slurs avoid scripts with `avoid-slur` set to `inside`, scripts avoid slurs with `avoid-slur` set to `around`. Slurs and scripts keep a distance of `slur-padding`.



‘`slur-script.ly`’

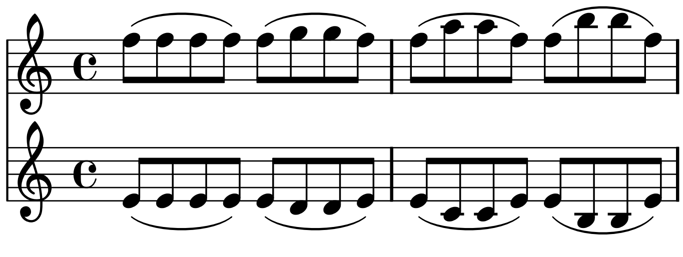
A slur avoids collisions with scripts. Articulations go inside the slur, dynamic markings go outside the slur. Fingerings and texts are placed either inside or outside.

For different configurations, the defaults can be changed, and scripts can be moved manually.



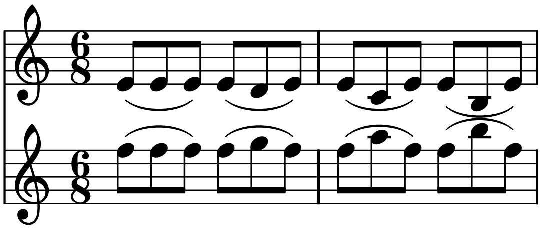
‘`slur-symmetry-1.ly`’

Symmetric figures should lead to symmetric slurs.



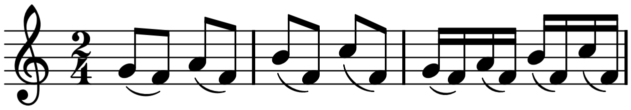
‘slur-symmetry.ly’

Symmetric figures should lead to symmetric slurs.



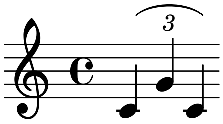
‘slur-tilt.ly’

The attachment point for strongly sloped slurs is shifted horizontally slightly. Without this correction, slurs will point into one note head, and point over another note head.



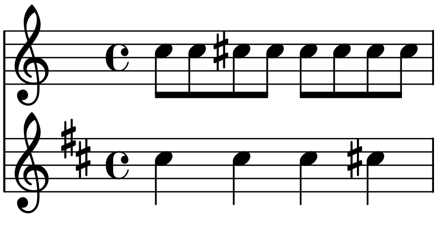
‘slur-tuplet.ly’

`TupletNumber` grobs are always inside slurs. This may not work if the slur starts after the tuplet.



‘spacing-accidental-staffs.ly’

Accidentals in different staves do not affect the spacing of the eighth notes here.

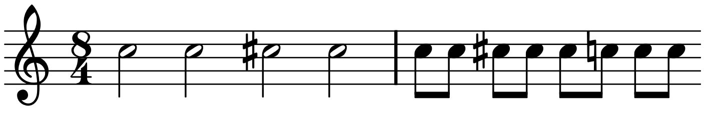


‘spacing-accidental-stretch.ly’

Accidentals do not influence the amount of stretchable space. The accidental does add a little non-stretchable space.

‘spacing-accidental.ly’

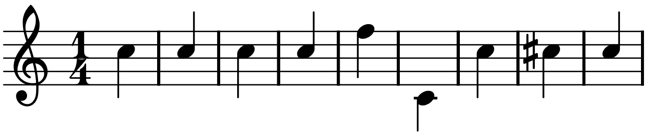
Accidentals sticking out to the left of a note will take a little more space, but only if the spacing is tight.



‘`spacing-bar-stem.ly`’

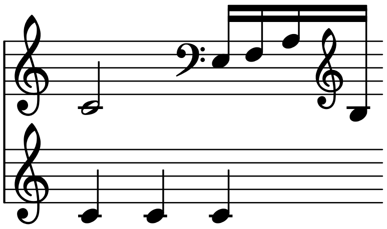
Downstem notes following a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.

Accidentals after the barline get some space as well.



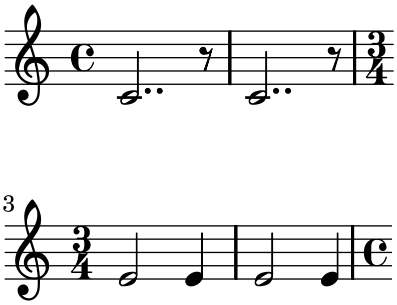
‘`spacing-clef-first-note.ly`’

Clef changes at the start of a line get much more space than clef changes halfway the line.



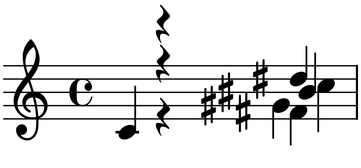
‘`spacing-end-of-line.ly`’

Broken engraving of a bar at the end of a line does not upset the space following rests and notes.



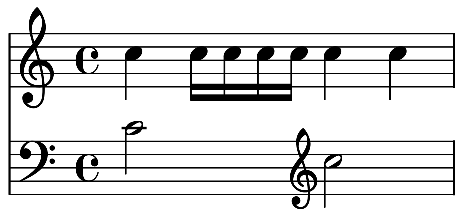
‘`spacing-ended-voice.ly`’

A voicelet (a very short voice to get polyphonic chords correct) should not confuse the spacing engine.



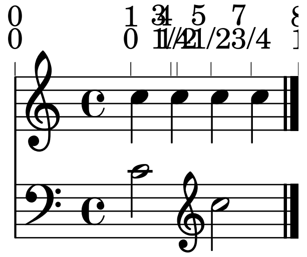
‘`spacing-folded-clef.ly`’

A clef can be folded below notes in a different staff, if this does not disrupt the flow of the notes.



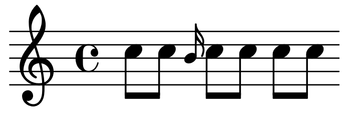
‘`spacing-folded-clef2.ly`’

A clef can be folded below notes in a different staff, if there is space enough. With `Paper_column` stencil callbacks we can show where columns are in the score.



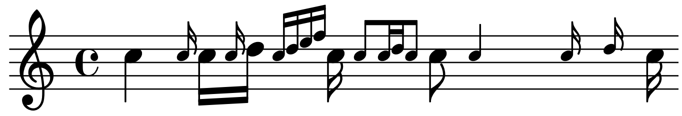
‘`spacing-grace-duration.ly`’

Spacing uses the duration of the notes, but disregards grace notes for this. In this example, the 8ths around the grace are spaced exactly as the other 8th notes.



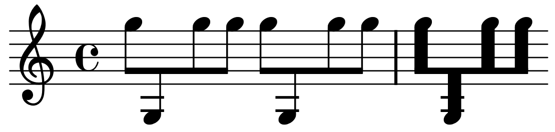
‘`spacing-grace.ly`’

Grace note runs have their own spacing variables in `Score.GraceSpacing`. So differing grace note lengths inside a run are spaced accordingly.



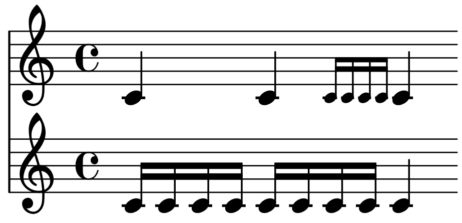
‘`spacing-knee.ly`’

For knees, the spacing correction is such that the stems are put at regular distances. This effect takes into account the width of the note heads and the thickness of the stem.



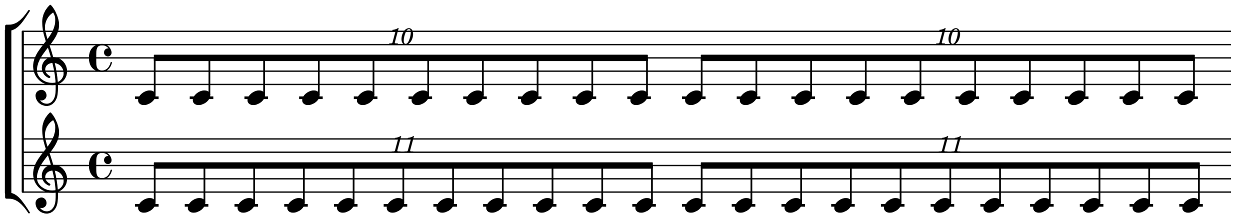
‘`spacing-loose-grace.ly`’

With `strict-grace-spacing`, grace notes don’t influence spacing.



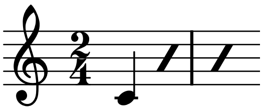
'`spacing-multi-tuplet.ly`'

Concurrent tuplets should be equidistant on all staffs. Such equidistant spacing is it at odds with elegant engraver spacing; hence it must be switched on explicitly with the `uniform-stretching` property of `SpacingSpanner`.



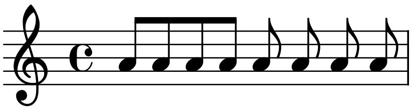
'`spacing-no-note.ly`'

In the absence of NoteSpacings, wide objects still get extra space. In this case, the slash before the barline gets a little more space.



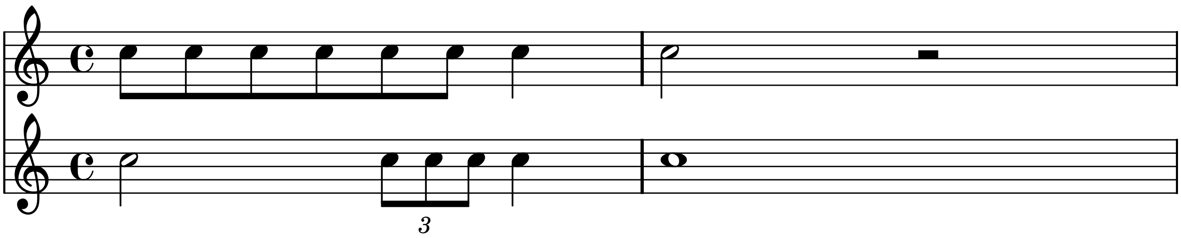
'`spacing-note-flags.ly`'

The flags of 8th notes take some space, but not too much: the space following a flag is less than the space following a beamed 8th head.



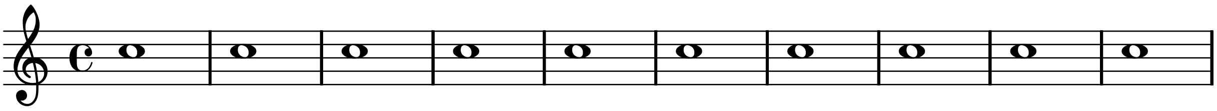
'`spacing-proportional.ly`'

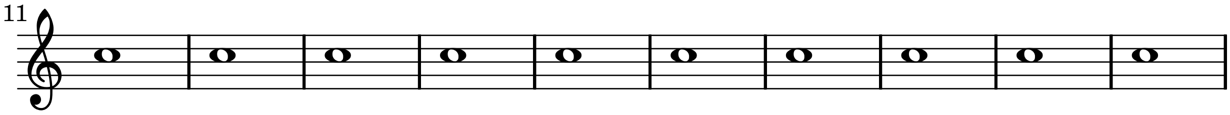
Proportional notation can be created by setting `proportionalNotationDuration`. Notes will be spaced proportional to the distance for the given duration.



'`spacing-ragged-last.ly`'

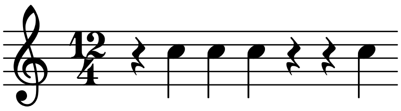
If `raggedlast` is set, the systems are broken similar to paragraph formatting in text: the last line is unjustified.





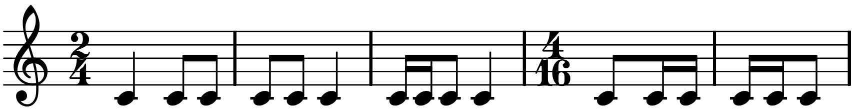
‘`spacing-rest.ly`’

Rests get a little less space, since they are narrower. However, the quarter rest in feta font is relatively wide, causing this effect to be very small.



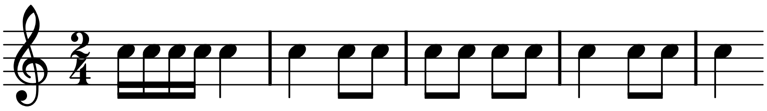
‘`spacing-section.ly`’

New sections for spacing can be started with `ewSpacingSection`. In this example, a section is started at the 4/16, and a 16th in the second section takes as much space as a 8th in first section.



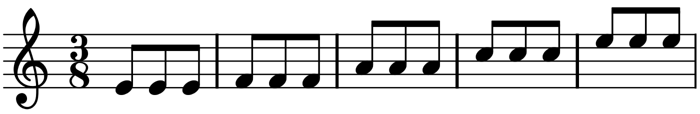
‘`spacing-short-notes.ly`’

Notes that are shorter than the common shortest note get a space (i.e. without the space needed for the note) proportional to their duration. So, the 16th notes get 1/2 of the space of an eigth note. The total distance for a 16th (which includes note head) is 3/4 of the eighth note.



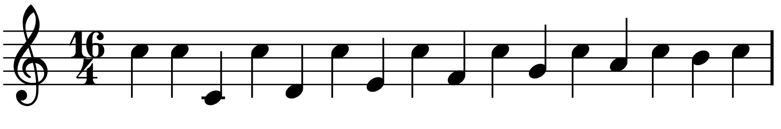
‘`spacing-stem-bar.ly`’

Upstem notes before a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.



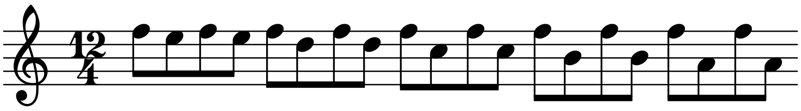
‘`spacing-stem-direction.ly`’

There are optical corrections to the spacing of stems. The overlap between two adjacent stems of different direction is used as a measure for how much to correct.



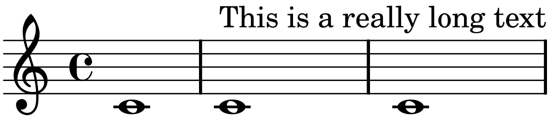
‘`spacing-stem-same-direction.ly`’

For juxtaposed chords with the same direction, a slight optical correction is used. It is constant, and works only if two chords have no common head-positions range.



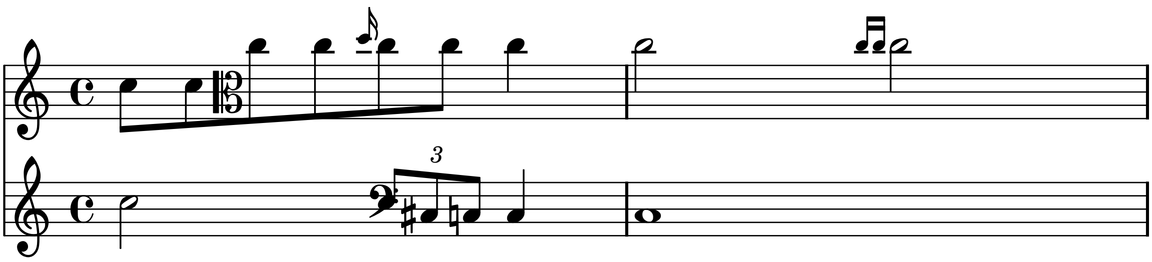
‘`spacing-stick-out.ly`’

If `keep-inside-line` is set for the relevant PaperColumn, LilyPond will space a line to prevent text sticking out of the right margin.



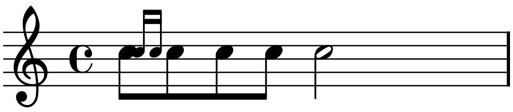
‘`spacing-strict-notespacing.ly`’

If `strict-note-spacing` is set, then spacing of notes is not influenced by bars and clefs half-way on the system. Rather, they are put just before the note that occurs at the same time. This may cause collisions.



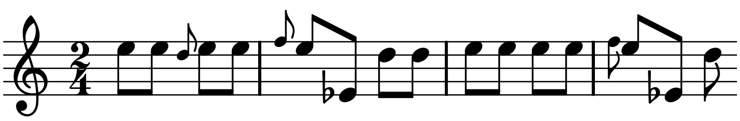
‘`spacing-strict-spacing-grace.ly`’

With `strict-note-spacing` spacing for grace notes (even multiple ones), is floating as well.



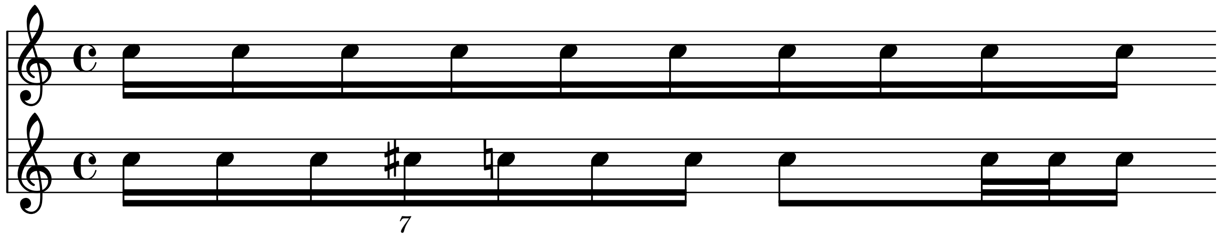
‘`spacing-to-grace.ly`’

Space from a normal note (or barline) to a grace note is smaller than to a normal note.



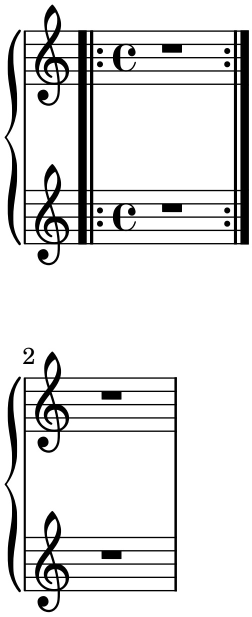
‘`spacing-uniform-stretching.ly`’

Notes are spaced exactly according to durations, if `uniform-stretching` is set. Accidentals are ignored, and no optical-stem spacing is performed.



‘`span-bar-break.ly`’

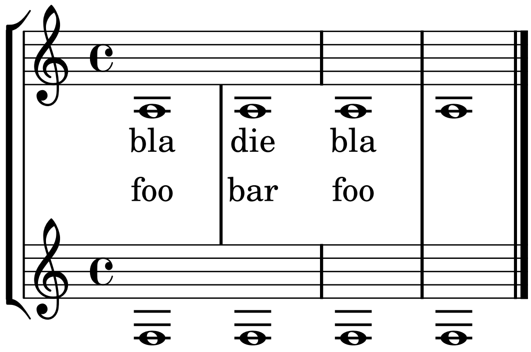
At the beginning of a system, the `|:` repeat barline is drawn between the staves, but the `:|` is not.



‘`span-bar.ly`’

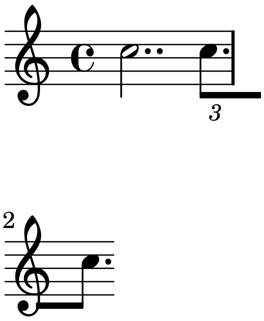
Span bars are drawn only between staff bar lines. By setting bar lines to transparent, they are shown only between systems.

Setting `SpanBar` transparent removes the barlines between systems.



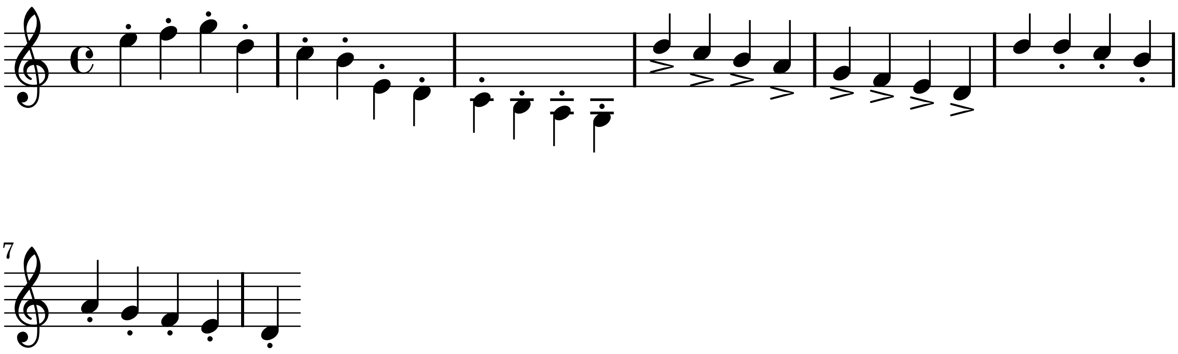
‘`spanner-break-overshoot.ly`’

The `break-overshoot` property sets the amount that a spanner (in this case: the beam) in case of a line break extends beyond the rightmost column and extends to the left beyond the prefatory matter.



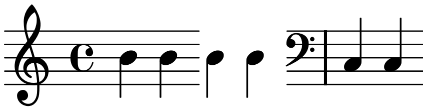
‘`staccato-pos.ly`’

Some scripts must have quantized postions. VErtical position descend monotonously for a descending scale. The staccato dot is close to the notehead. If the head is in a space, then the dot is in the space next to it.



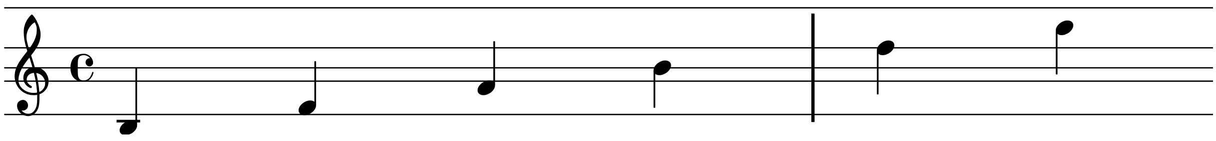
‘`staff-halfway.ly`’

Staves can be started and stopped at command.



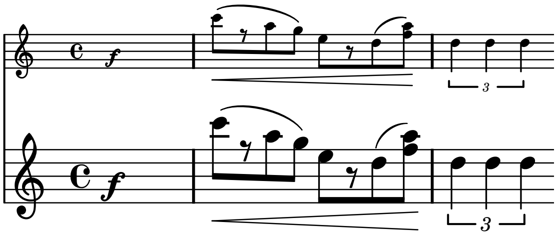
‘`staff-line-positions.ly`’

The vertical positions of staff lines may be specified individually, by setting the `line-positions` property of the StaffSymbol.



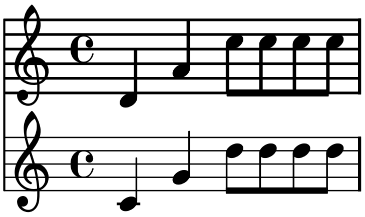
‘`staff-mixed-size.ly`’

Staves may be present in several sizes within a score. This is achieved with an internal scaling factor. If the scaling factor is forgotten in some places, objects generally become too thick or too large on smaller staves.



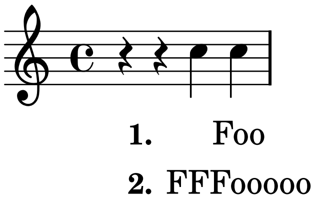
‘`staff-tweak.ly`’

The staff is a grob (graphical object) which may be adjusted as well, for example, to have 6 thick lines and a slightly large `staff-space`. However, beams remain correctly quantized.



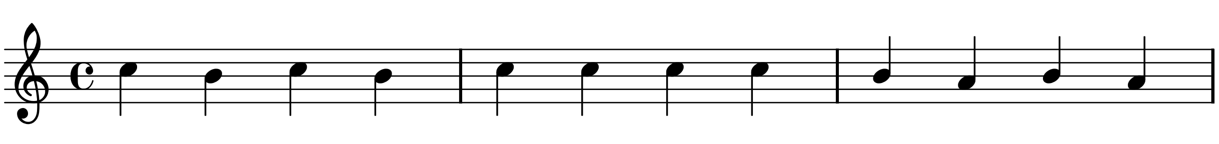
‘`stanza-number.ly`’

Stanza numbers are put left of their lyric. They are aligned in a column.



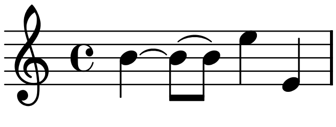
‘`stem-direction-context.ly`’

Stem directions for notes on the middle staff line are determined by the directions of their neighbors.



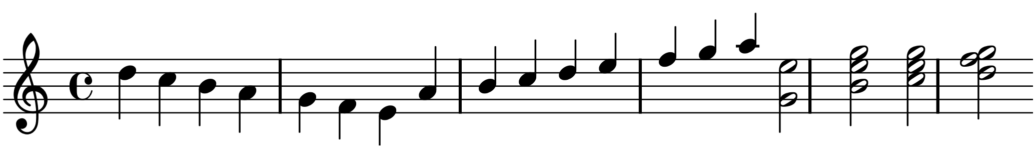
‘`stem-direction.ly`’

Stems, beams, ties and slurs should behave similarly, when placed on the middle staff line. Of course stem-direction is down for high notes, and up for low notes.



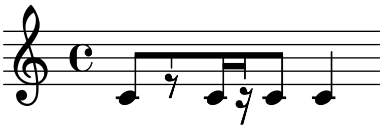
‘`stem-shorten.ly`’

If note head is ‘over’ the center line, the stem is shortened. This happens with forced stem directions, and with some chord configurations.



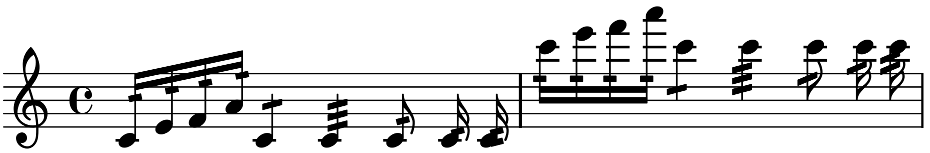
‘`stem-stemlet.ly`’

Stemlets are small stems under beams over rests. Their length can be set with `stemlet-length`.



‘`stem-tremolo-position.ly`’

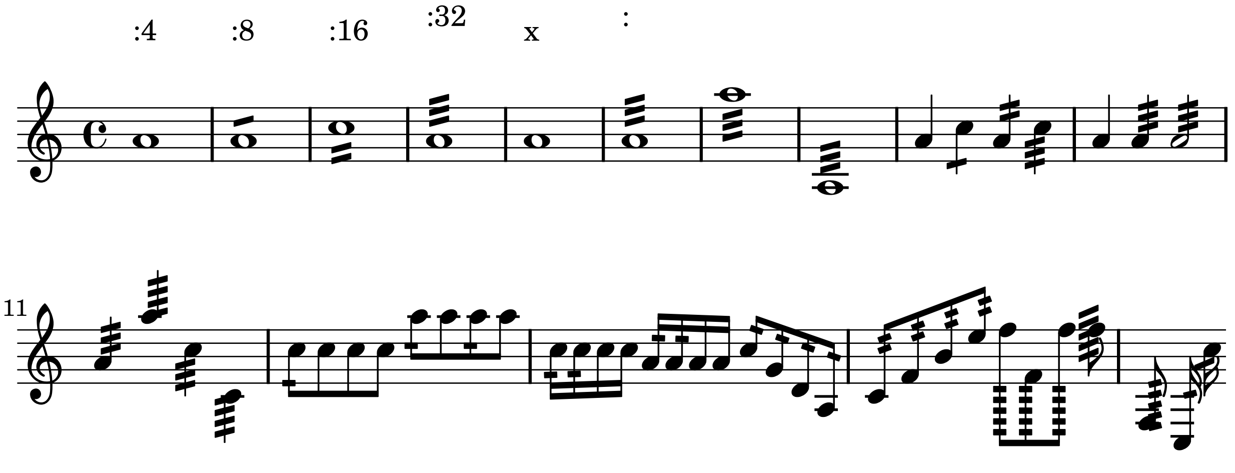
Tremolos are positioned a fixed distance from the end of the beam. Tremolo flags are shortened and made rectangular on beamed notes or on stem-up notes with a flag. Tremolo flags are tilted extra on stem-down notes with a flag.



‘`stem-tremolo.ly`’

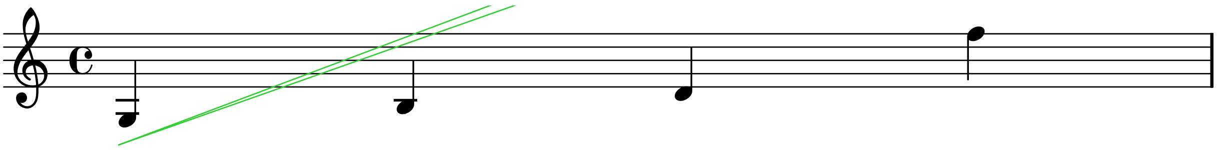
Stem tremolos or rolls are tremolo signs that look like beam segments crossing stems. If the stem is in a beam, the tremolo must be parallel to the beam. If the stem is invisible (e.g. on a whole note), the tremolo must be centered on the note. If the note has a flag (eg. an unbeamed 8th note), the tremolo should be shortened if the stem is up and tilted extra if the stem is down.

The tremolos should be positioned a fixed distance from the end of the stems unless there is no stem, in which case they should be positioned a fixed distance from the note head.



'`stencil-color-rotation.ly`'

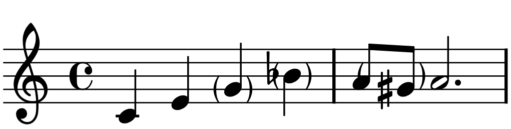
Combinations of rotation and color do work.



'`stencil-hacking.ly`'

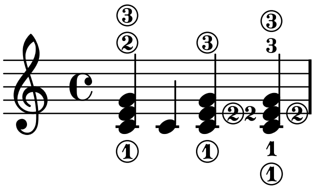
You can write stencil callbacks in Scheme, thus providing custom glyphs for notation elements. A simple example is adding parentheses to existing stencil callbacks.

The parenthesized beam is less successful due to implementation of the Beam. The note head is also rather naive, since the extent of the parens are also not seen by accidentals.



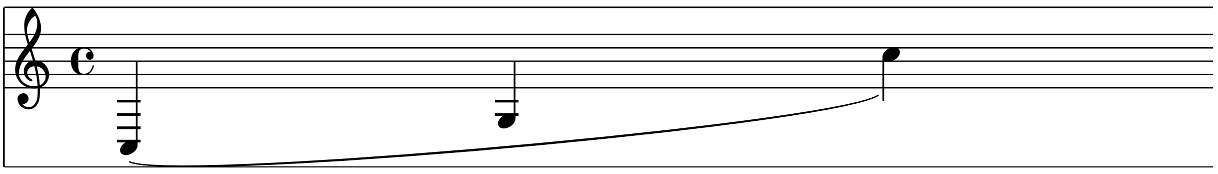
'`string-number.ly`'

String numbers can be added to chords. They use the same positioning mechanism as finger instructions.



'`system-extents.ly`'

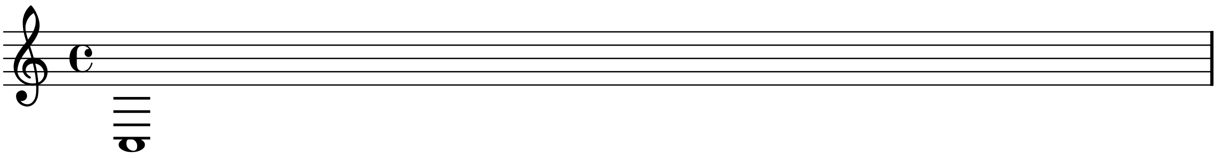
The size of every system is correctly determined; this includes postscript constructs such as slurs.

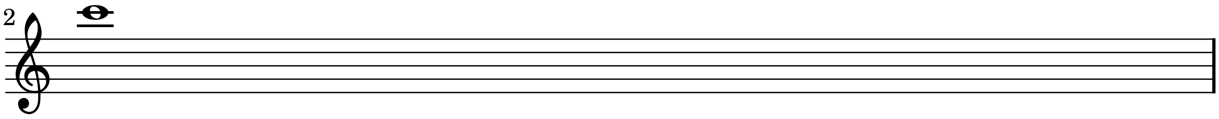


'`system-overstrike.ly`'

By setting between-system-padding to a negative value, it is possible to eliminate the anti-collision constraints. Then setting `between-system-space` to a low (nonzero) value, print systems in overstrike.

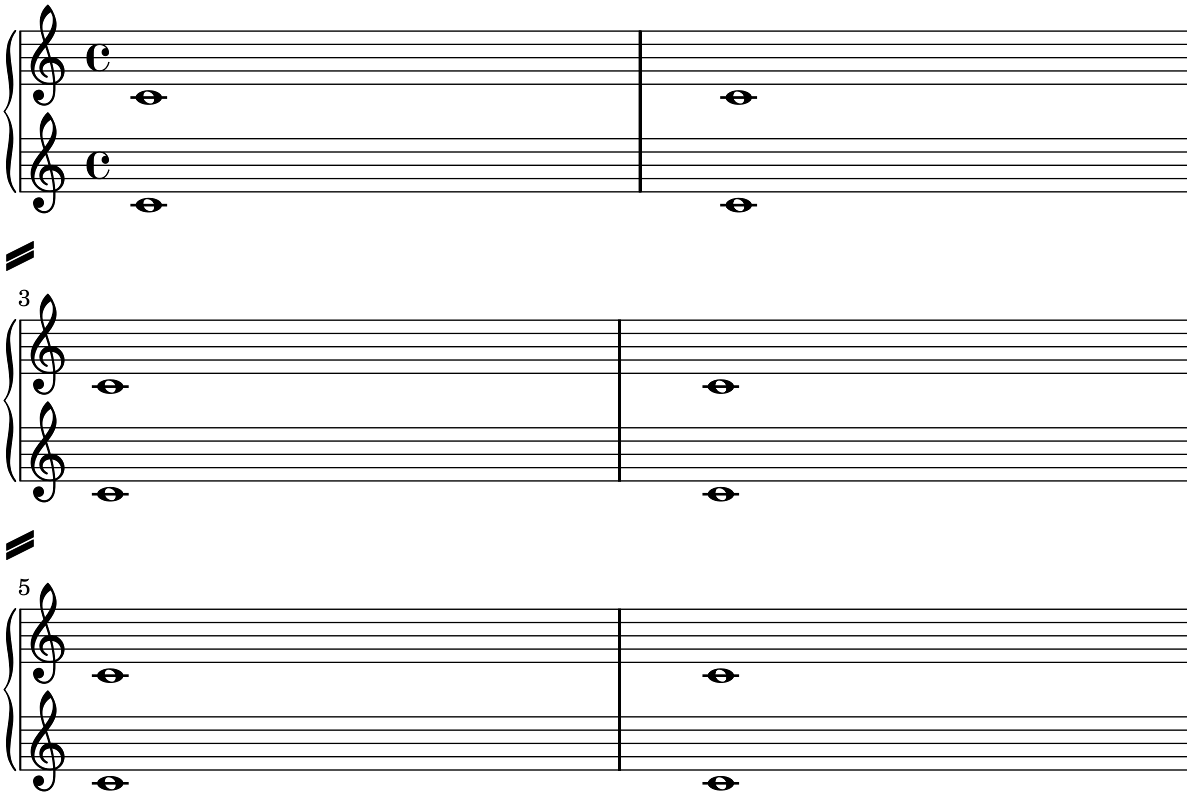
Unfortunately, this does not show in the colllated texinfo document. Run this example stand-alone to see the effect.





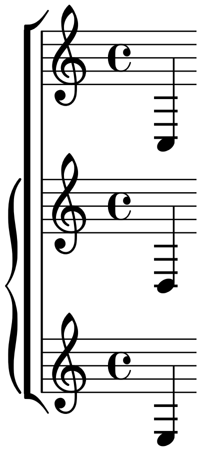
‘`system-separator.ly`’

System separators maybe defined as markups in the `systemSeparator` field of the bookpaper block. They are centered between the boundary staffs of each system.


3
5

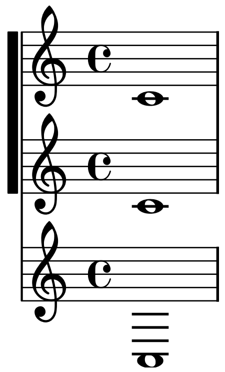
‘`system-start-bracket.ly`’

The piano brace should be shifted horizontally if it is enclosed in a bracket.



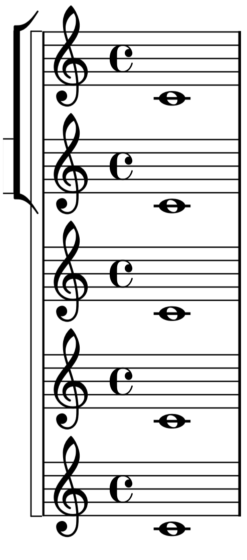
‘`system-start-heavy-bar.ly`’

A heavy-bar system start delimiter may be created by tuning the `SystemStartBar` grob.



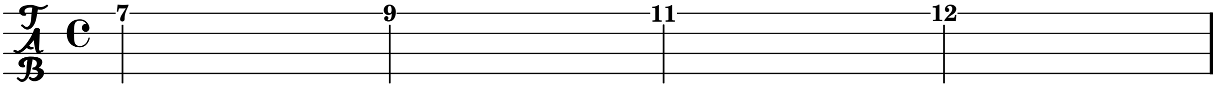
‘`system-start-nesting.ly`’

Deeply nested system braces/brackets/etc. may be created with the `Nested_system_start_delimiter_engraver`



‘`tablature-string-tunings.ly`’

For other tunings, it is sufficient to set `stringTunings`. The number of staff lines is adjusted accordingly.

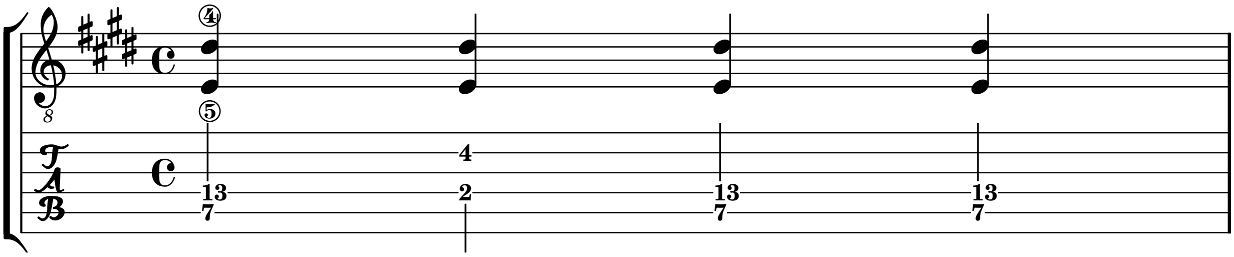


‘`tablature.ly`’

A sample tablature, with both normal staff and tab.

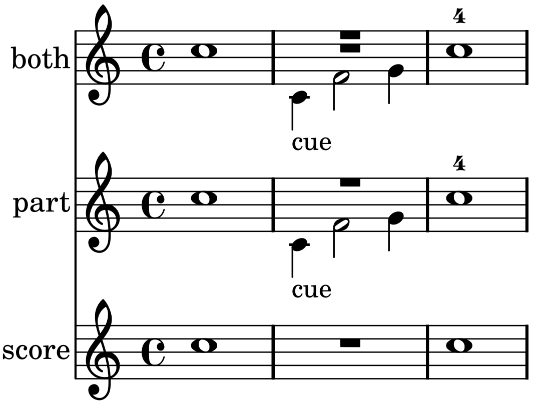
Tablature is done by overriding the note-head formatting function, and putting it on a 6-line staff. A special engraver takes care of going from string-number + pitch to number.

String numbers can be entered as note articulations (inside a chord) and chord articulations (outside a chord)



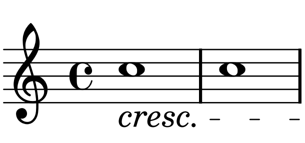
‘`tag-filter.ly`’

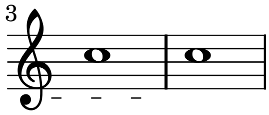
The `\tag` command marks music expressions with a name. These tagged expressions can be filtered out later. This mechanism can be used to make different versions of the same music. In this example, the top stave displays the music expression with all tags included. The bottom two staves are filtered: the part has cue notes and fingerings, but the score has not.



‘`text-spanner.ly`’

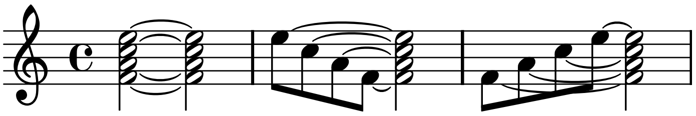
Text spanners should not repeat start text when broken.





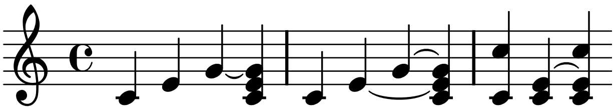
‘`tie-arpeggio-collision.ly`’

Advanced tie chord formatting also works with arpegiated ties. Due to arpeggios, tie directions may be changed relative to the unarpegiated case.



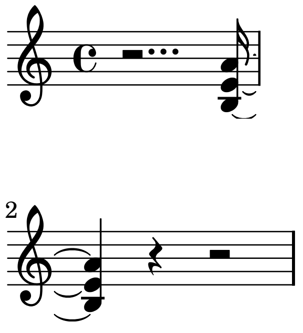
‘`tie-arpeggio.ly`’

when `tieWaitForNote` is set, the right-tied note does not have to follow the lef-tied note directly. When `tieWaitForNote` is set to false, any tie will erase all pending ties.



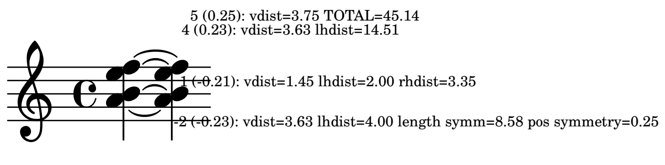
‘`tie-broken.ly`’

Ties behave properly at line breaks.



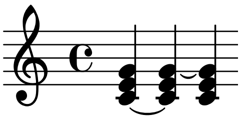
‘`tie-chord-debug.ly`’

Switching on debug-tie-scoring annotates the tie scoring decisions made.


5 (0.25): vdist=3.75 TOTAL=45.14
4 (0.23): vdist=3.63 lhdist=14.51
1 (-0.21): vdist=1.45 lhdist=2.00 rhdist=3.35
-2 (-0.23): vdist=3.63 lhdist=4.00 length symm=8.58 pos symmetry=0.25

‘`tie-chord-partial.ly`’

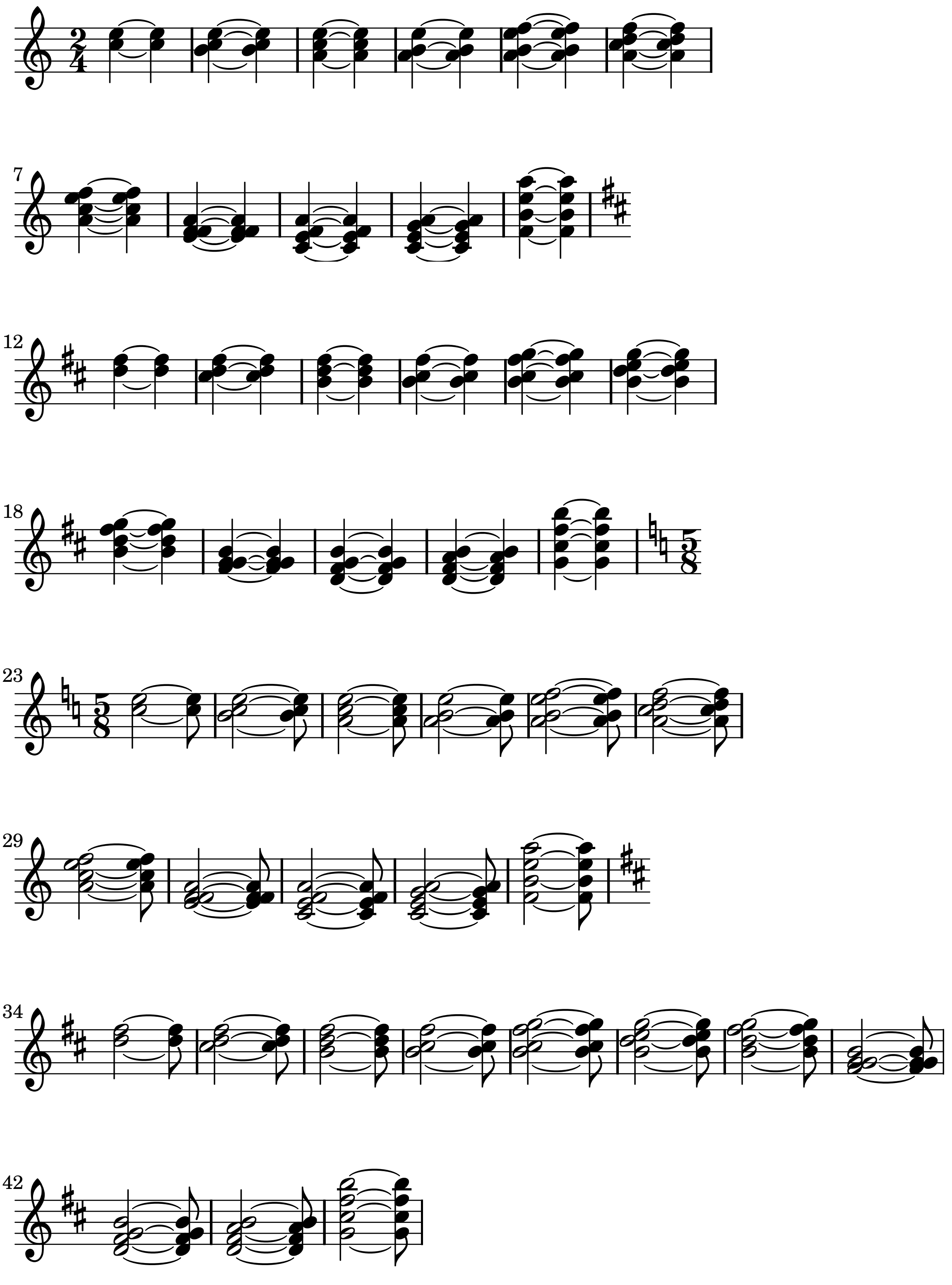
Individual chord notes can also be tied



'`tie-chord.ly`'

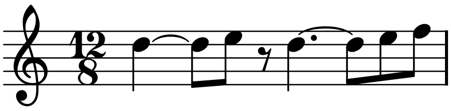
In chords, ties keep closer to the note head vertically, but never collide with heads or stems. Seconds are formatted up/down; the rest of the ties are positioned according to their vertical position.

The code does not handle all cases. Sometimes ties will printed on top of or very close to each other. This happens in the last chords of each system.



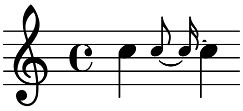
‘`tie-dot.ly`’

Ties avoid collisions with dots.



‘`tie-grace.ly`’

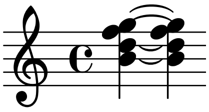
Tieing a grace to the to a following grace or main note works.



‘`tie-manual.ly`’

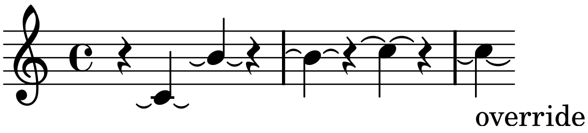
Tie formatting may be adjusted manually, by setting the `tie-configuration` property. The override should be placed at the second note of the chord.

You can leave a Tie alone by introducing a non-pair value (eg. `#t`) in the `tie-configuration` list.



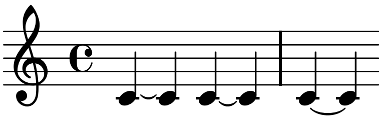
‘`tie-semi-single.ly`’

Like normal ties, single semities (LaissezVibrerTie or RepeatTie) get their direction from the stem direction, and may be tweaked with `#'direction`.



‘`tie-single-manual.ly`’

Individual ties may be formatted manually by specifying their `direction` and/or `staff-position`.

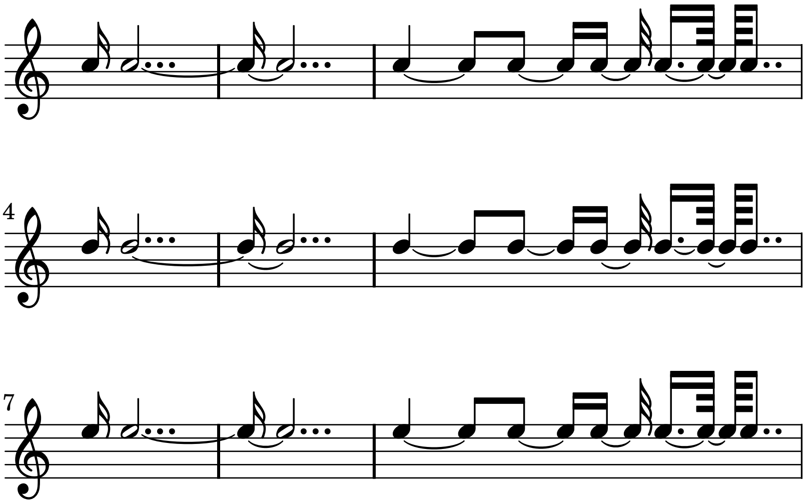


‘`tie-single.ly`’

Formatting for isolated ties.

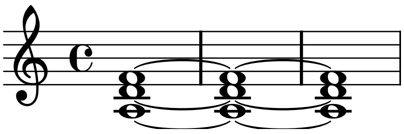
- short ties are in spaces
- long ties cross staff lines
- ties avoid flags of left stems.
- ties avoid dots of left notes.

- short ties are vertically centered in the space, as well those that otherwise don't fit in a space
- extremely short ties are put over the noteheads, instead of inbetween.



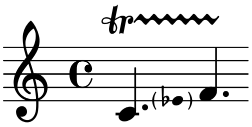
'`tie-whole.ly`'

For whole notes, the inside ties do not cross the center of the note head, horizontally.



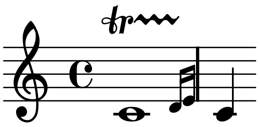
'`trill-spanner-pitched.ly`'

Pitched trills are denoted by a small note head in parentheses following the main note. This note head is properly ledgered, and parentheses include the accidental.



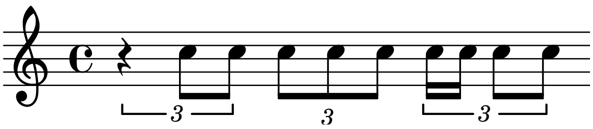
'`trill-spanner.ly`'

Trill spanner



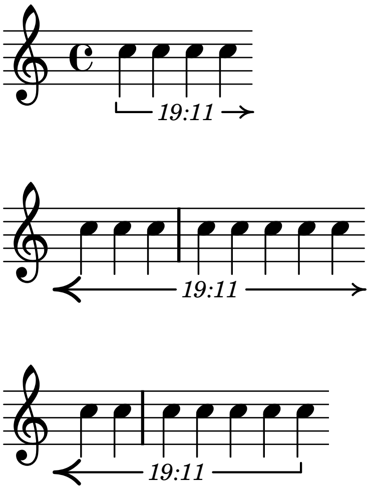
'`tuplet-beam.ly`'

In combination with a beam, the bracket of the tuplet bracket is removed. This only happens if there is one beam, as long as the bracket.



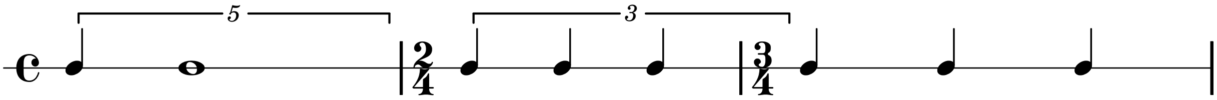
‘`tuplet-broken.ly`’

Broken tuplets are adorned with little arrows. The arrows come from the `edge-text` property, and thus be replaced with larger glyphs or other text.



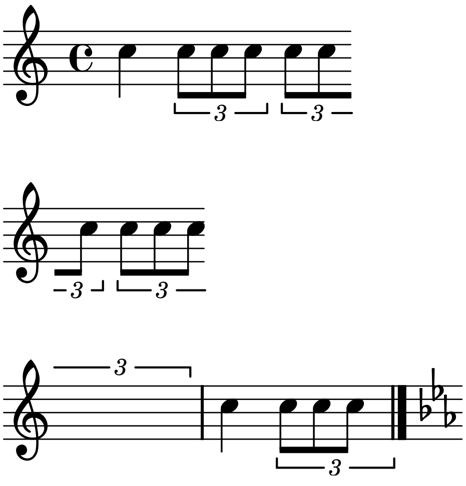
‘`tuplet-full-length-note.ly`’

tuplet can be made to run to prefatory matter or the next note, by setting `tupletFullLengthNote`.



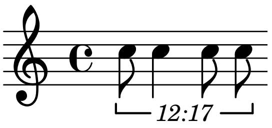
‘`tuplet-full-length.ly`’

If `tupletFullLength` is set, tuplets end at the start of the next non-tuplet note.



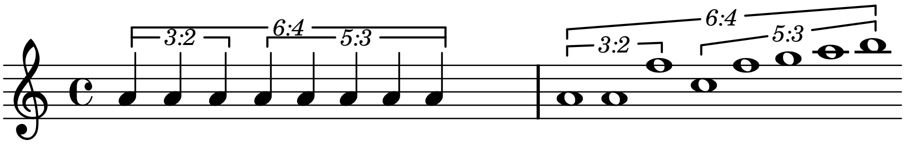
‘`tuplet-gap.ly`’

The size of the tuplet bracket gap is adjusted to the width of the text.



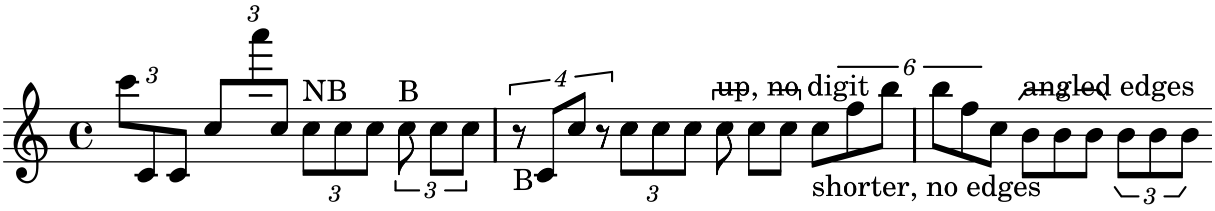
'`tuplet-nest.ly`'

Tuplets may be nested.



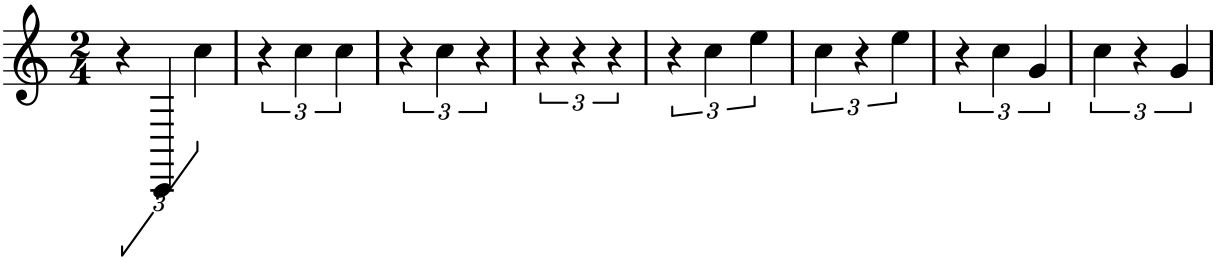
'`tuplet-properties.ly`'

Tuplet bracket formatting supports numerous options, for instance, bracketed (B) and non-bracketed (NB).



'`tuplet-rest.ly`'

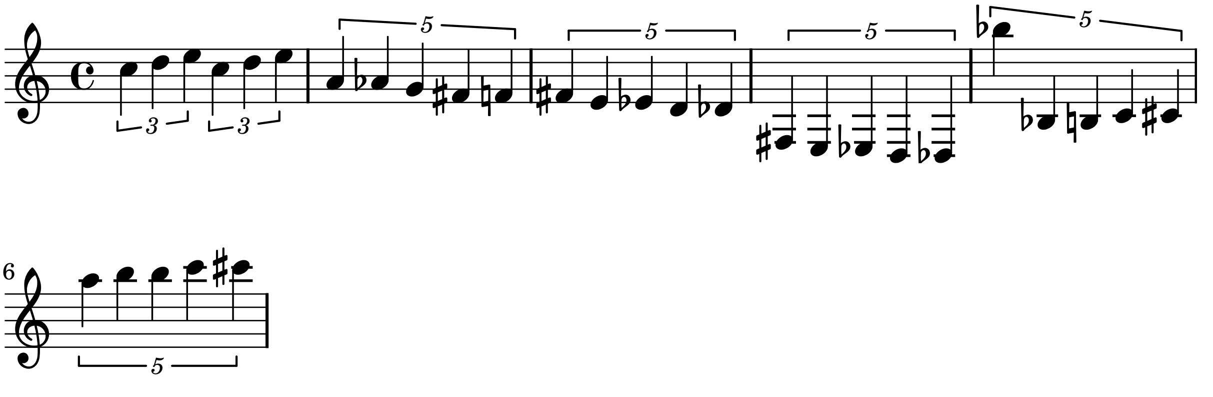
Tuplets may contain rests.



'`tuplet-slope.ly`'

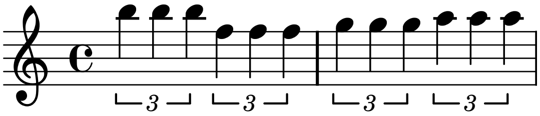
Tuplet brackets stay clear of the staff. The slope is determined by the graphical characteristic of the notes, but if the musical pattern does not follow graphical slope, then the bracket is horizontal

The bracket direction is determined by the dominating stem direction.



`tuplet-staffline-collision.ly`

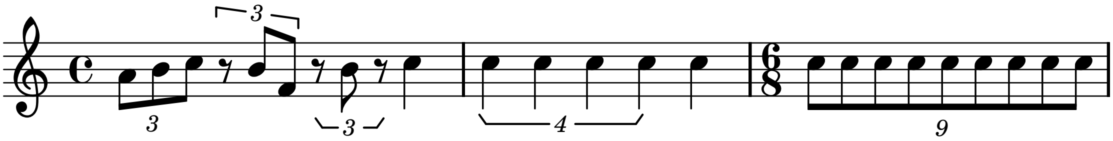
Horizontal tuplet brackets are shifted vertically to avoid staff line collisions.



`tuplets.ly`

Tuplets are indicated by a bracket with a number. There should be no bracket if there is a beam exactly matching the length of the tuplet. The bracket does not interfere with the stafflines, and the number is centered in the gap in the bracket.

The bracket stops at the end of the stems, if the stems have the same direction as the bracket. The endings can be adjusted with `bracket-flare`.



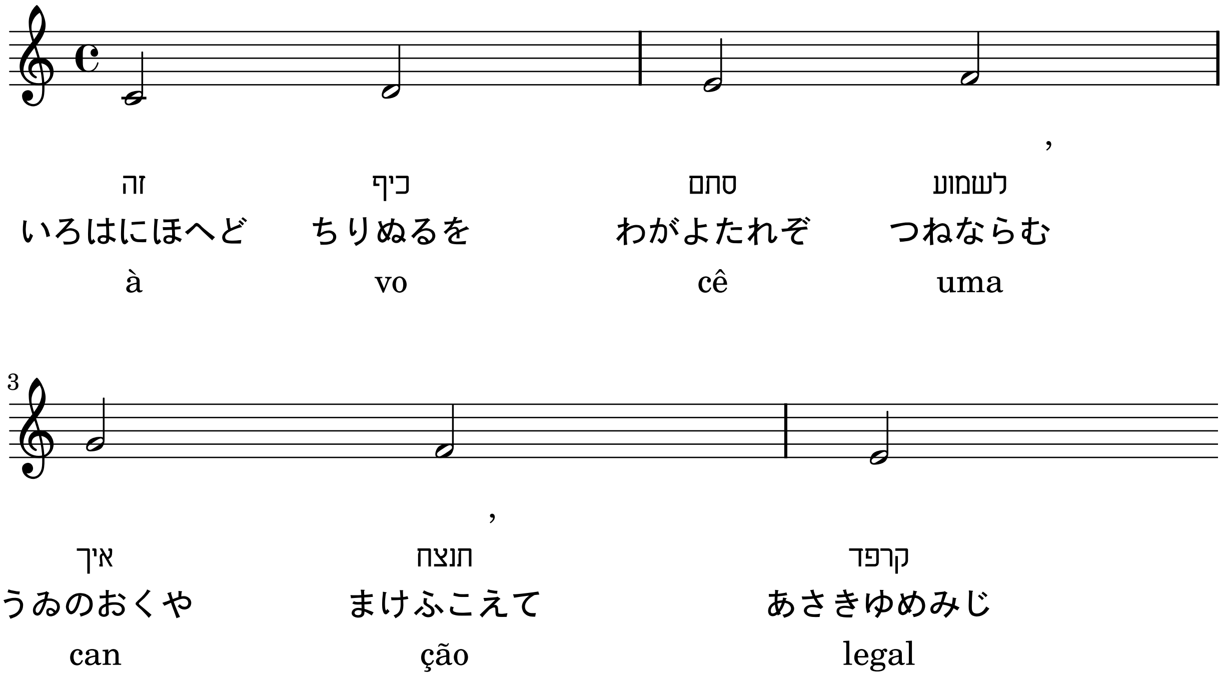
`utf-8-mixed-text.ly`

words in mixed font in a single string are separated by spaces as in the input string. Here a Russian word followed by a roman word.

Hallo

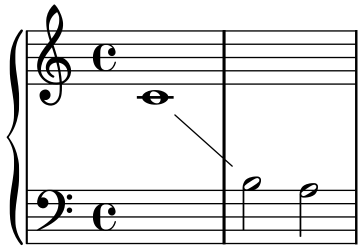
‘`utf-8.ly`’

Various scripts may be used for texts (like titles and lyrics) introduced by entering them in UTF-8 encoding, and using a Pango based backend. Depending on the fonts installed, this fragment will render Bulgarian (Cyrillic), Hebrew, Japanese and Portuguese.



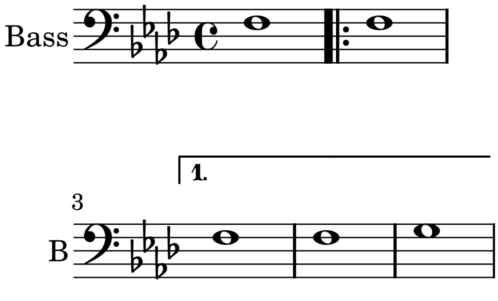
‘`voice-follower.ly`’

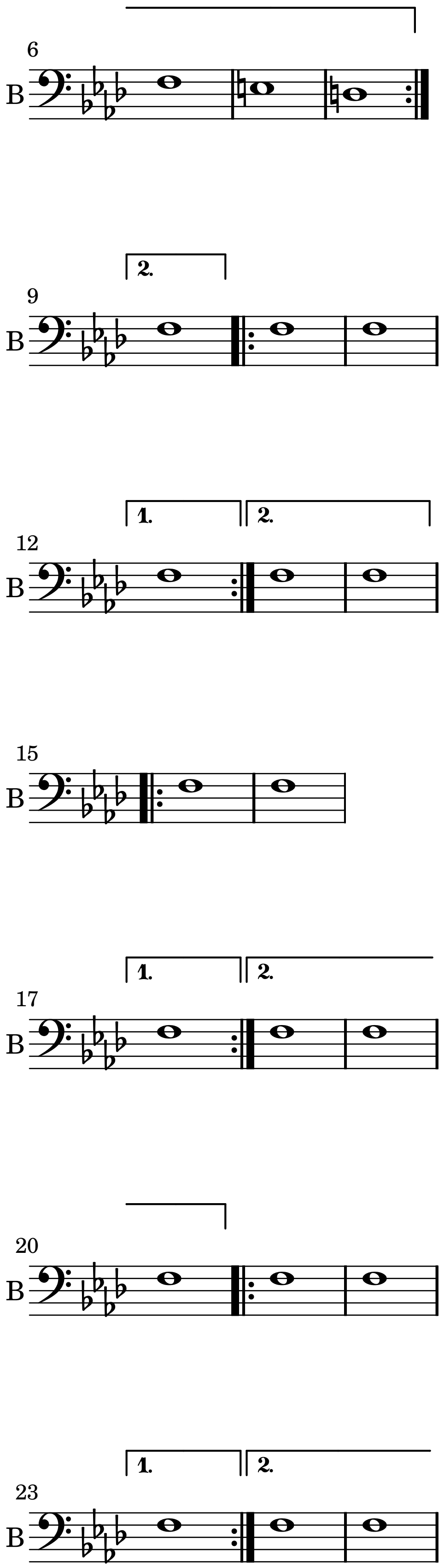
Whenever a voice switches to another staff a line connecting the notes can be printed automatically. This is enabled if the property `followVoice` is set to true.



‘`volta-broken-left-edge.ly`’

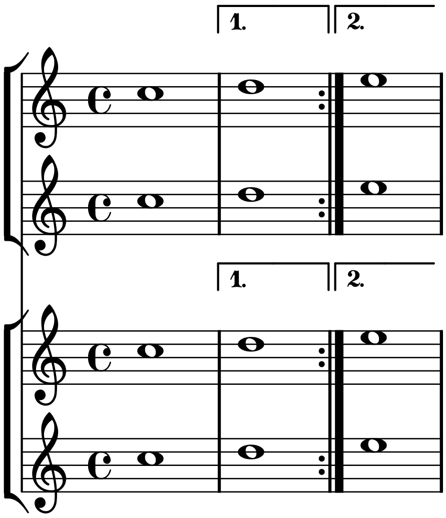
Broken volta spanners behave correctly at their left edge in all cases.





‘`volta-multi-staff.ly`’

By setting `voltaOnThisStaff`, repeats can be put also over other staves than the topmost one in a score.



‘`whiteout.ly`’

The whiteout command underlays a white box under a markup. The whitening effect only is only guaranteed for staff lines, since staff lines are in a different layer.