LilyPond Regression Tests

Introduction

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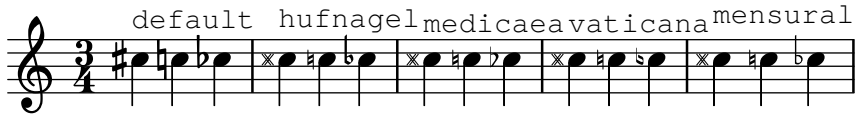
In the web version of this document, you can click on the file name or figure for each example to see the corresponding input file.

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

Regression test cases

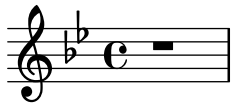
Accidentals are available in different ancient styles, which all are collected here.
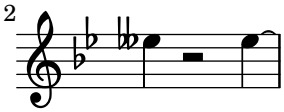
`accidental-ancient.ly`
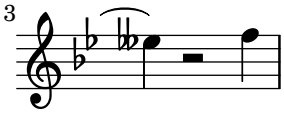


When a tie is broken, the spacing engine must consider the accidental after the line break. The second and third lines should have the same note spacing.

`accidental-broken-tie-spacing.ly`
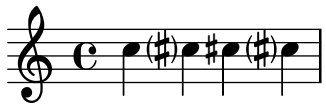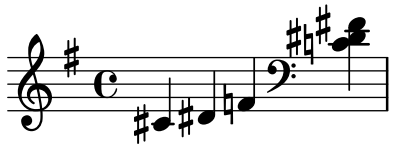






Cautionary accidentals may be indicated using either parentheses (default) or smaller accidentals.

`accidental-cautionary.ly`
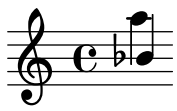


Accidentals are invalidated at clef changes.

`accidental-clef-change.ly`



accidentals avoid stems of other notes too.

`accidental-collision.ly`
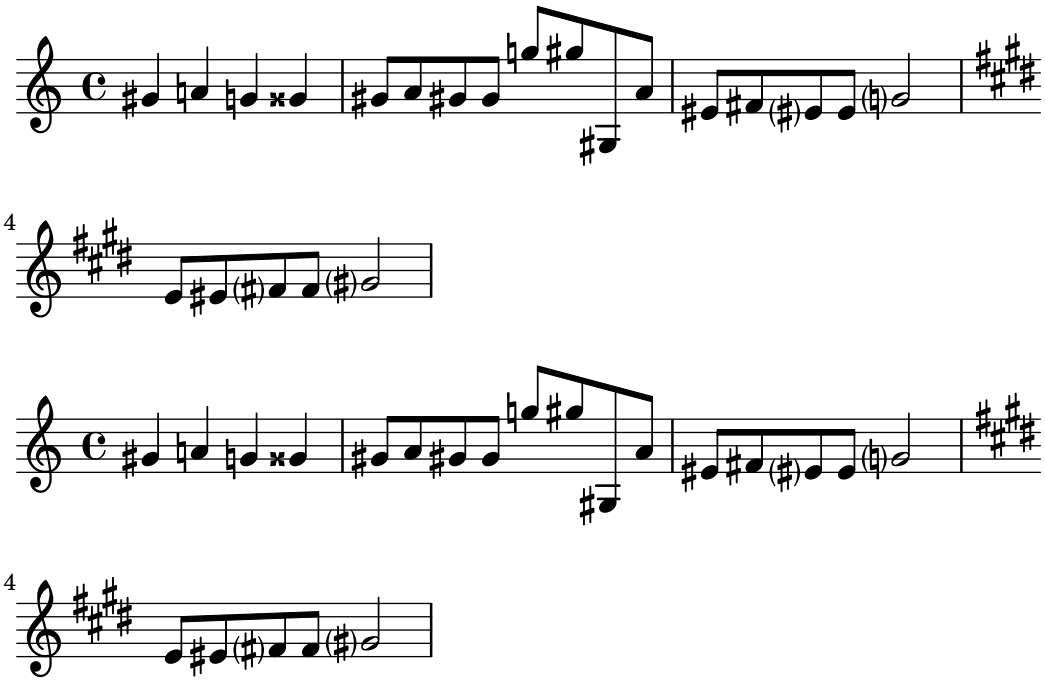


Several automatic accidental rules aim to reproduce contemporary music notation practices:

- `'dodecaphonic` style prints accidentals on every note (including naturals)

- `'neo-modern style` prints accidentals on every note (not including naturals), except when a note is immediately repeated
- `'neo-modern-cautionary` style acts like neo-modern, adding cautionary parentheses around accidentals.
- `'teaching` prints accidentals normally, but adds cautionary accidentals when an accidental is already included in the key signature.
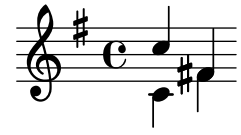
Both scores should show the same accidentals.

`accidental-contemporary.ly`



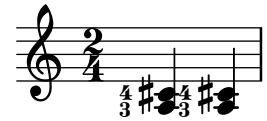
If two forced accidentals happen at the same time, only one sharp sign is printed.

`accidental-double.ly`
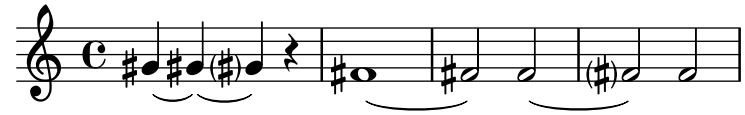


Horizontal `Fingering` grobs should not collide with accidentals.

`accidental-fingering-collision.ly`



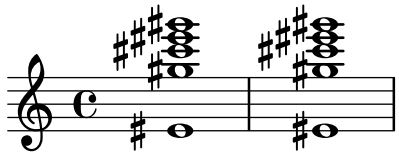
Accidentals can be forced with ! and ? even if the notes are tied. Cautionary accidentals applied to tied notes after a bar line are valid for the whole measure.

`accidental-forced-tie.ly`



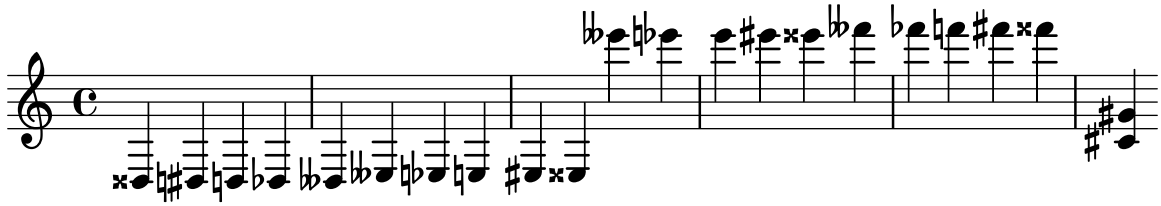
By setting `accidentalGrouping` to `'voice`, LilyPond will horizontally stagger the accidentals of octaves in different voices as seen in this test's E-sharp.

`accidental-grouping.ly`
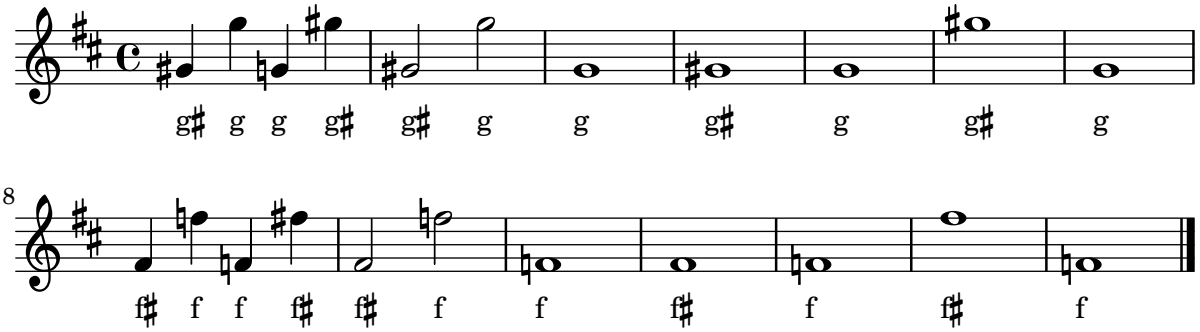


Ledger lines are shortened when there are accidentals. This happens only for the single ledger line close to the note head, and only if the accidental is horizontally close to the head.

`accidental-ledger.ly`
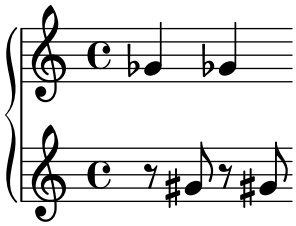


This shows how accidentals in different octaves are handled. The note names are also automatically printed but the octavation has been dropped out.

`accidental-octave.ly`



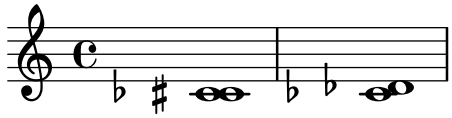
In piano accidental style, notes in both staves influence each other. In this example, each note should have an accidental.

`accidental-piano.ly`
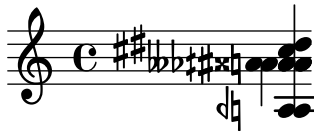


Accidental padding works for all accidentals, including those modifying the same pitch.

`accidental-placement-padding.ly`
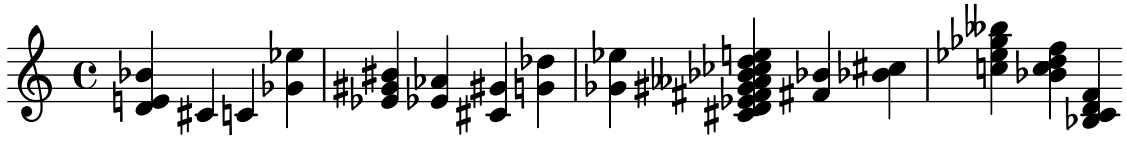


When two (or more) accidentals modify the same pitch, they are printed adjacent to one another unless they represent the same alteration, in which case they are printed in exactly the same position as one another. In either case, collisions with accidentals of different pitches are correctly computed.

`accidental-placement-samepitch.ly`
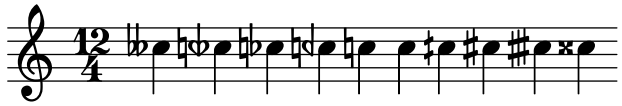


Accidentals are placed as closely as possible. Accidentals in corresponding octaves are aligned. The top accidental should be nearest to the chord. The flats in a sixth should be staggered.

`accidental-placement.ly`
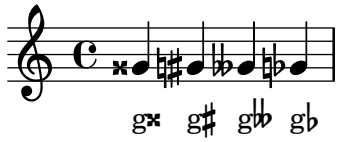


Quarter tone notation is supported, including threequarters flat.

`accidental-quarter.ly`



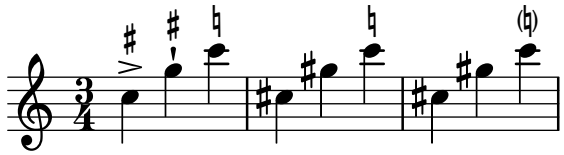
A sharp sign after a double sharp sign, as well as a flat sign after a double flat sign is automatically prepended with a natural sign.

`accidental-single-double.ly`



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

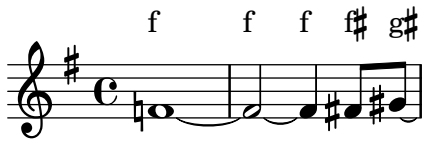
`accidental-suggestions.ly`

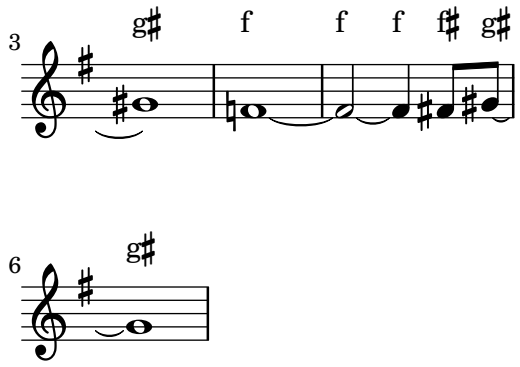


The second and third notes should not get accidentals, because they are tied to a note. However, an accidental is present if the line is broken at the tie, which happens for the G sharp.

The presence of an accidental after a broken tie can be overridden.

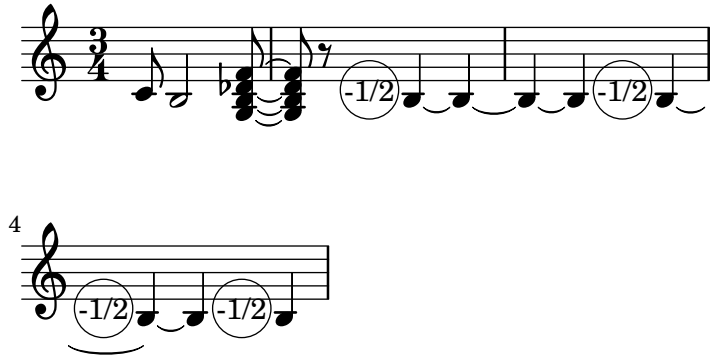
`accidental-tie.ly`





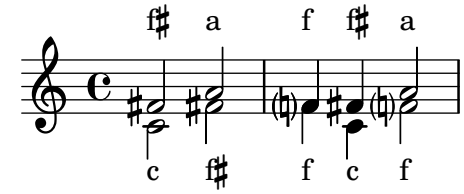
Space is allowed for the actual size of accidentals on tied notes.

`accidental-unbroken-tie-spacing.ly`
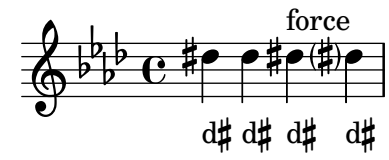


This shows how modern cross voice auto cautionary accidentals are handled. The first two fisses get accidentals because they belong to different voices. The first f gets cautionary natural because of previous measure. The last f gets cautionary natural because fis was only in the other voice.

`accidental-voice.ly`



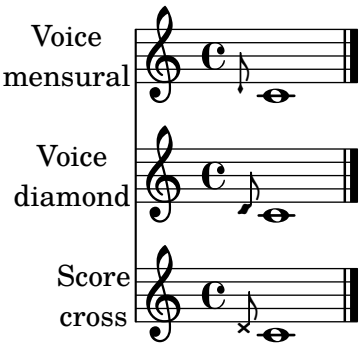
Accidentals work: the second note does not get a sharp. The third and fourth show forced and cautionary accidentals.

`accidental.ly`
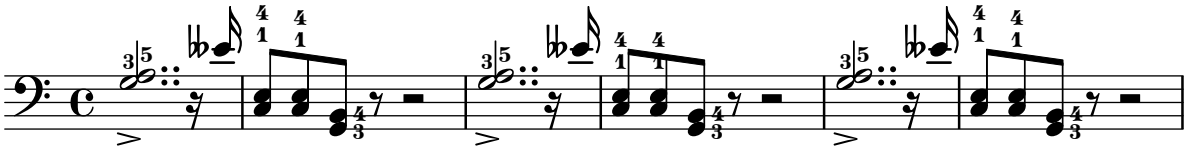


`\add-grace-property` can be used at various context levels in order to override grace properties. Overrides in different parallel contexts are independent.

`add-grace-property.ly`



`add-stem-support` can be removed or implemented only for beamed notes.

`add-stem-support.ly`

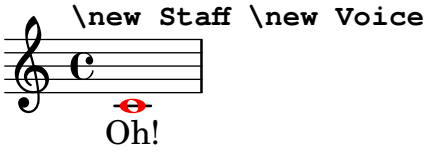


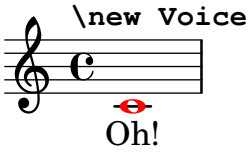
This is a test of combining post-events with various constructs. Problems are reported on the stderr of this run; there are no images produced.

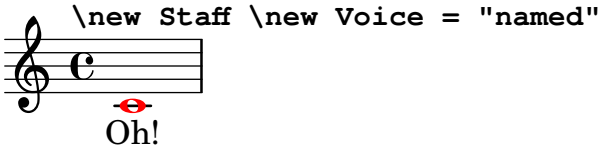
`added-post-event-test.ly`

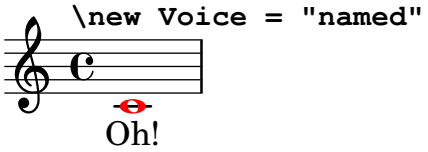
`\addlyrics` should be able to attach itself to named and unnamed `Voice` constructs. For all tests where this succeeds, the noteheads will be red.

`addlyrics-existing-context.ly`



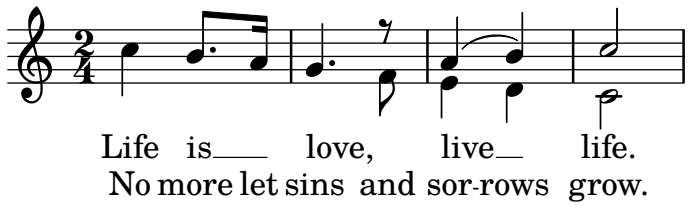






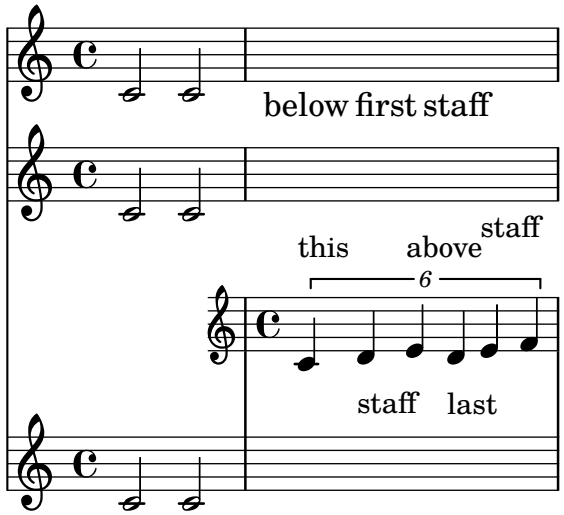
`\addlyrics` may get used on a `Staff` context and will then consider all note events created below it for synchronization.

addlyrics-to-staff-context.ly



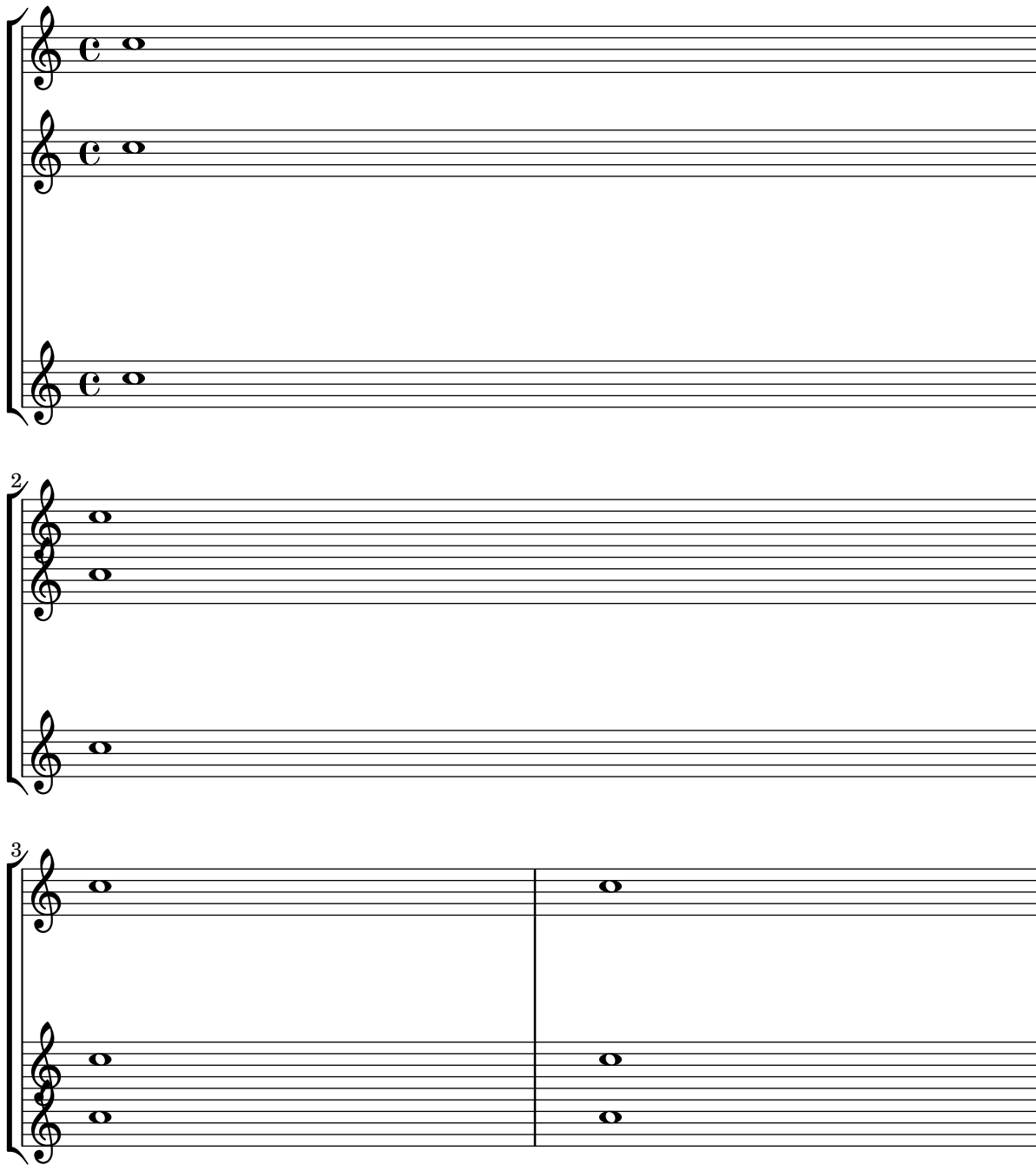
Newly created contexts can be inserted anywhere in the vertical alignment.

alignment-order.ly



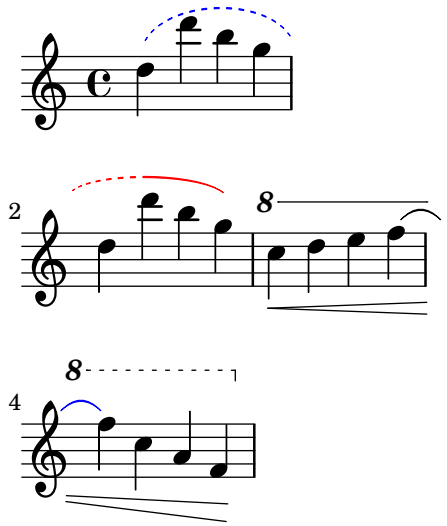
Alignments may be changed per system by setting `alignment-distances` in the `line-break-system-details` property

alignment-vertical-manual-setting.ly



The command `\alterBroken` may be used to override the pieces of a broken spanner independently. The following example demonstrates its usage with a variety of data types.

`alter-broken.ly`
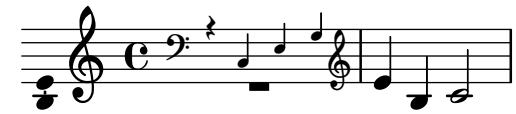


Ambitus for pieces beginning with `\cueDuringWithClef`.

Cues are often used at or near the beginning of a piece. Furthermore, a cue is frequently in a different clef, so the `\cueDuringWithClef` command is handy. Using this command at the beginning of a piece should leave the ambitus displayed based on the main clef.
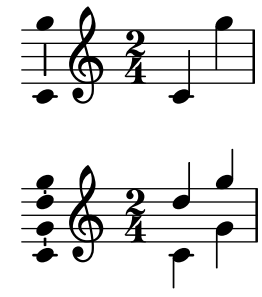
An `Ambitus_engraver` should ignore notes in `CueVoice` contexts.

`ambitus-cue.ly`



The gaps between an `AmbitusLine` and its note heads are set by the `gap` property. By default, `gap` is a function that reduces the gap for small intervals (e.g. a fourth), so that the line remains visible.
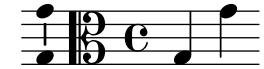
`ambitus-gap.ly`
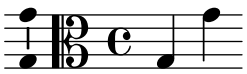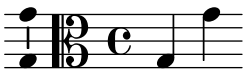


Ambitus engraver should obey middleCOffset, middleCPosition, and the staffLineLayoutFunction.

All three staves should look the same.

`ambitus-middleC.ly`

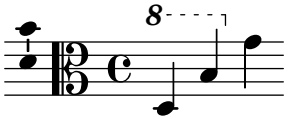






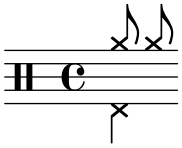
A voice starting with \ottava shouldn't confuse ambitus.

`ambitus-ottava.ly`
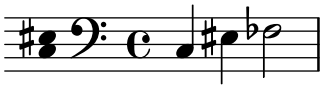


Adding ambitus to percussion contexts does not cause crashes, since the `Ambitus_engraver` will only acknowledge pitched note heads.

`ambitus-percussion-staves.ly`
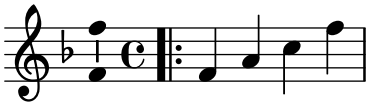


Ambitus use actual pitch not lexicographic ordering.
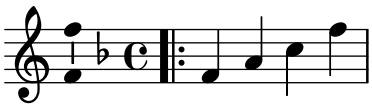
`ambitus-pitch-ordering.ly`
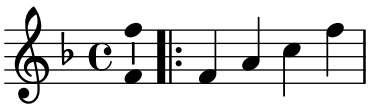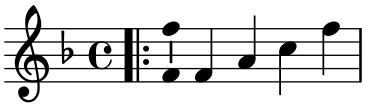


Ambitus can be moved to various positions with correct horizontal spacing in all cases.

`ambitus-position.ly`
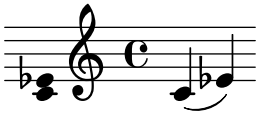








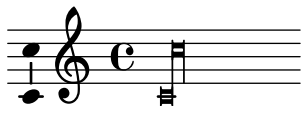
Ambitus accidentals (whether present or not) are ignored by the slur engravers.

`ambitus-slur.ly`



A `\Voice` should be able to contain both an `Ambitus_engraver` and a `Mensural_ligature_engraver` without segfaulting.
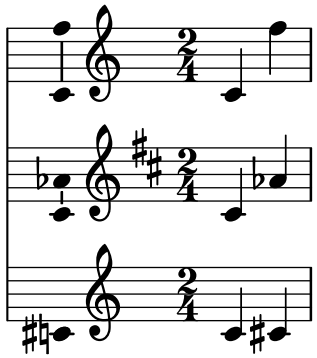
`ambitus-with-ligature.ly`



Ambitus indicate pitch ranges for voices.

Accidentals only show up if they're not part of key signature. `AmbitusNoteHead` grobs also have ledger lines. The noteheads are printed in overstrike, so there's only one visible; the accidentals are prevented from colliding.
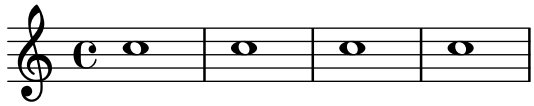
`ambitus.ly`



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

This example prints a bar-number during processing on stdout.
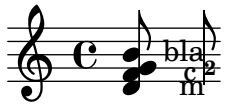
`apply-context.ly`



The `\applyOutput` expression is the most flexible way to tune properties for individual grobs.
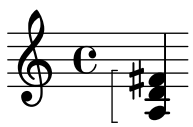
Here, the layout of a note head is changed depending on its vertical position.

`apply-output.ly`



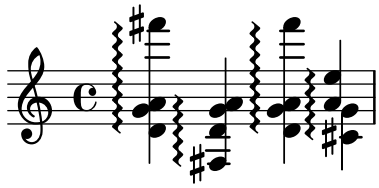
A square bracket on the left indicates that the player should not arpeggiate the chord.

`arpeggio-bracket.ly`



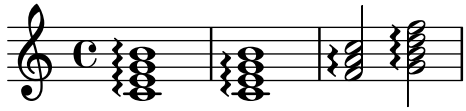
Arpeggio stays clear of accidentals and flipped note heads.

`arpeggio-collision.ly`



Arpeggios do not overshoot the highest note head. The first chord in this example simulates overshoot using `'positions` for comparison with the correct behaviour.

`arpeggio-no-overshoot.ly`



Arpeggios stil work in the absence of a staff-symbol.

`arpeggio-no-staff-symbol.ly`



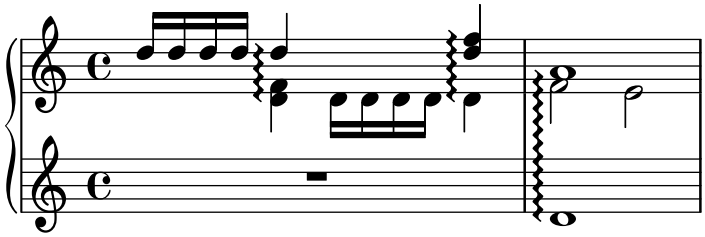
There is a variant of the arpeggio sign that uses a ‘vertical slur’ instead of the wiggle.

`arpeggio-parenthesis.ly`



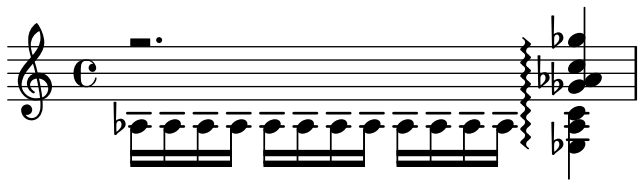
Cross-staff or -voice arpeggios which include single note heads as anchors do not collide with previous note heads or prefatory material.

`arpeggio-span-collision.ly`



Span arpeggios that are not cross-staff do not have horizontal spacing problems.

`arpeggio-span-one-staff-collision.ly`
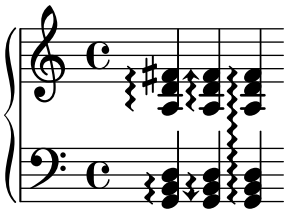


Span arpeggios within one staff also work

`arpeggio-span-one-staff.ly`
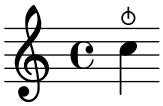


Arpeggios are supported, both cross-staff and broken single staff.

`arpeggio.ly`
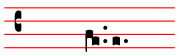


The snappizzicato articulation adds a snappizzicato sign to the note.

`articulation-snappizzicato.ly`
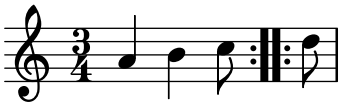


Augmentum dots are accounted for in horizontal spacing.

`augmentum.ly`
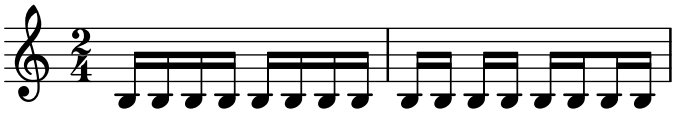


No auto beams will be put over (manual) repeat bars.

`auto-beam-bar.ly`
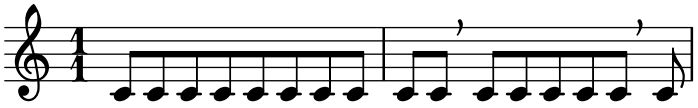


Autobeamer remembers `subdivideBeams` and other beaming pattern related functions at the start of an autobeam.

`auto-beam-beaming-override.ly`
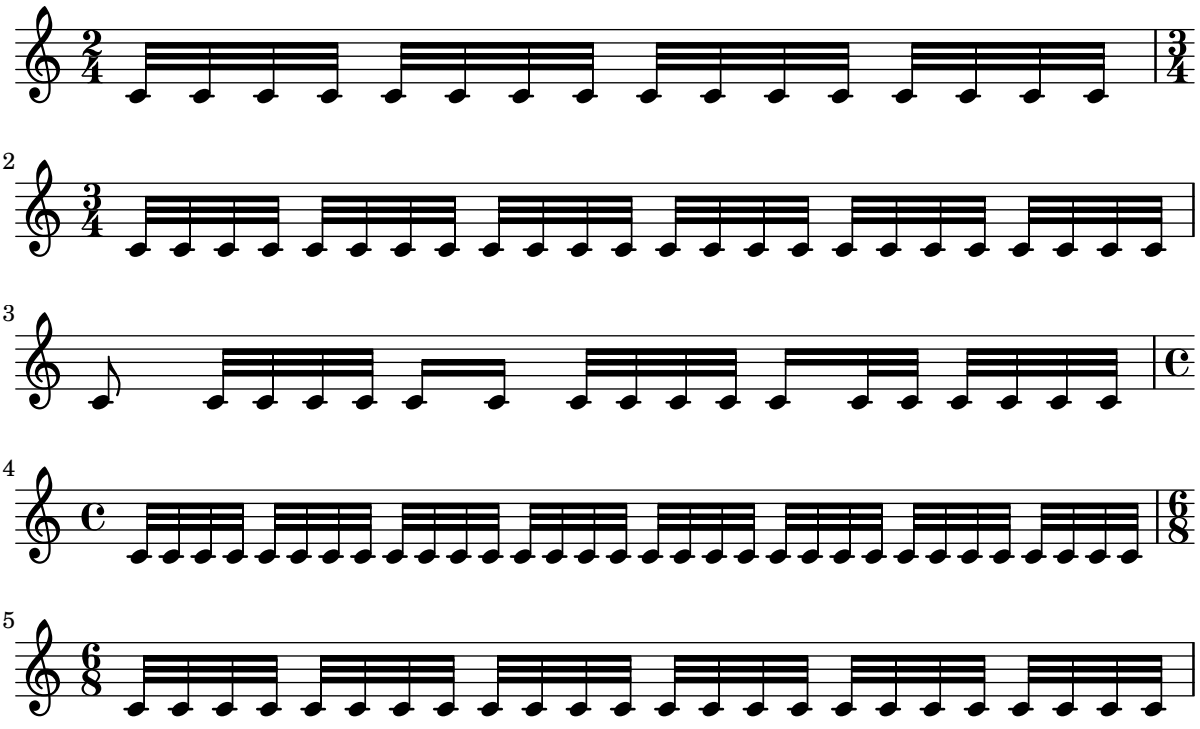


Automatic beams are ended early if a breathing sign is encountered.

`auto-beam-breathe.ly`
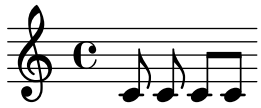


`beamExceptions` is used to modify the automatic beaming for certain durations; the expected grouping is given after the note duration.

`auto-beam-exceptions.ly`
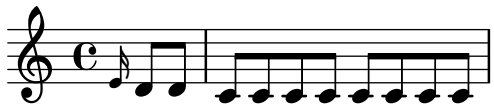


The autobeamer may be switched off for a single note with `\noBeam`.

`auto-beam-no-beam.ly`
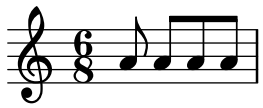


Grace notes at the start of a partial measure do not break autobeaming.

`auto-beam-partial-grace.ly`
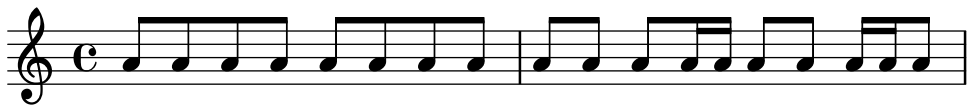


Autobeaming works properly in partial measures.
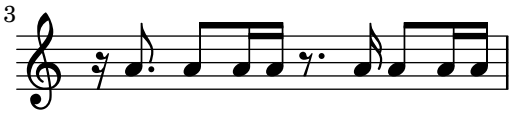
`auto-beam-partial.ly`



In 4/4 time, the first and second and third and fourth beats should be beamed together if only eighth notes are involved. If any shorter notes are included, each beat should be beamed separately.

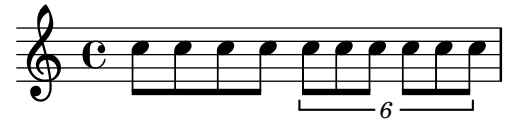
`auto-beam-recheck.ly`





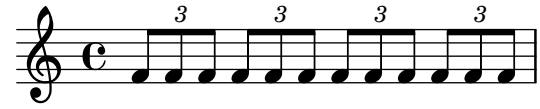
Automatic beaming is also done on tuplets.

`auto-beam-triplet.ly`



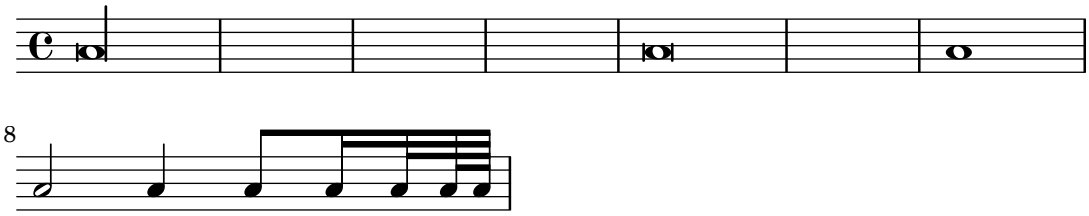
Tuplet-spanner should not put (visible) brackets on beams even if they're auto generated.

`auto-beam-tuplets.ly`
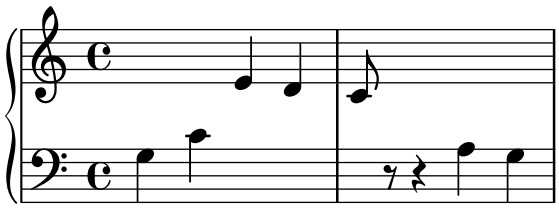


Beams are placed automatically; the last measure should have a single beam.

`auto-beam.ly`



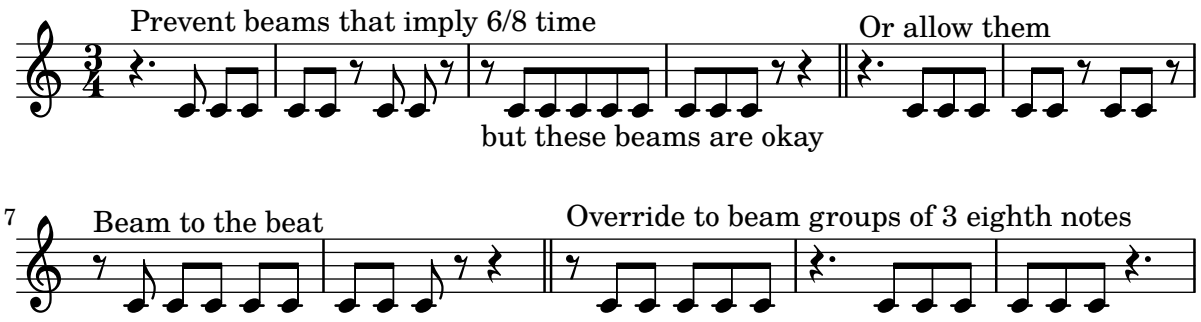
Auto change piano staff switches voices between up and down staves automatically; rests are switched along with the coming note. When central C is reached, staff is not yet switched (by default).

`auto-change.ly`
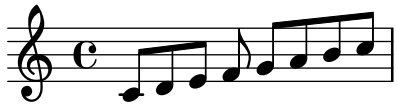


Beaming in 3/4 time has special treatment. By default six eighth notes are beamed in one. Beams that would imply 6/8 time may be avoided with `beamHalfMeasure = ##f`. When the beaming is changed, beams should start at the beginning of the measure.

`autobeam-3-4-rules.ly`
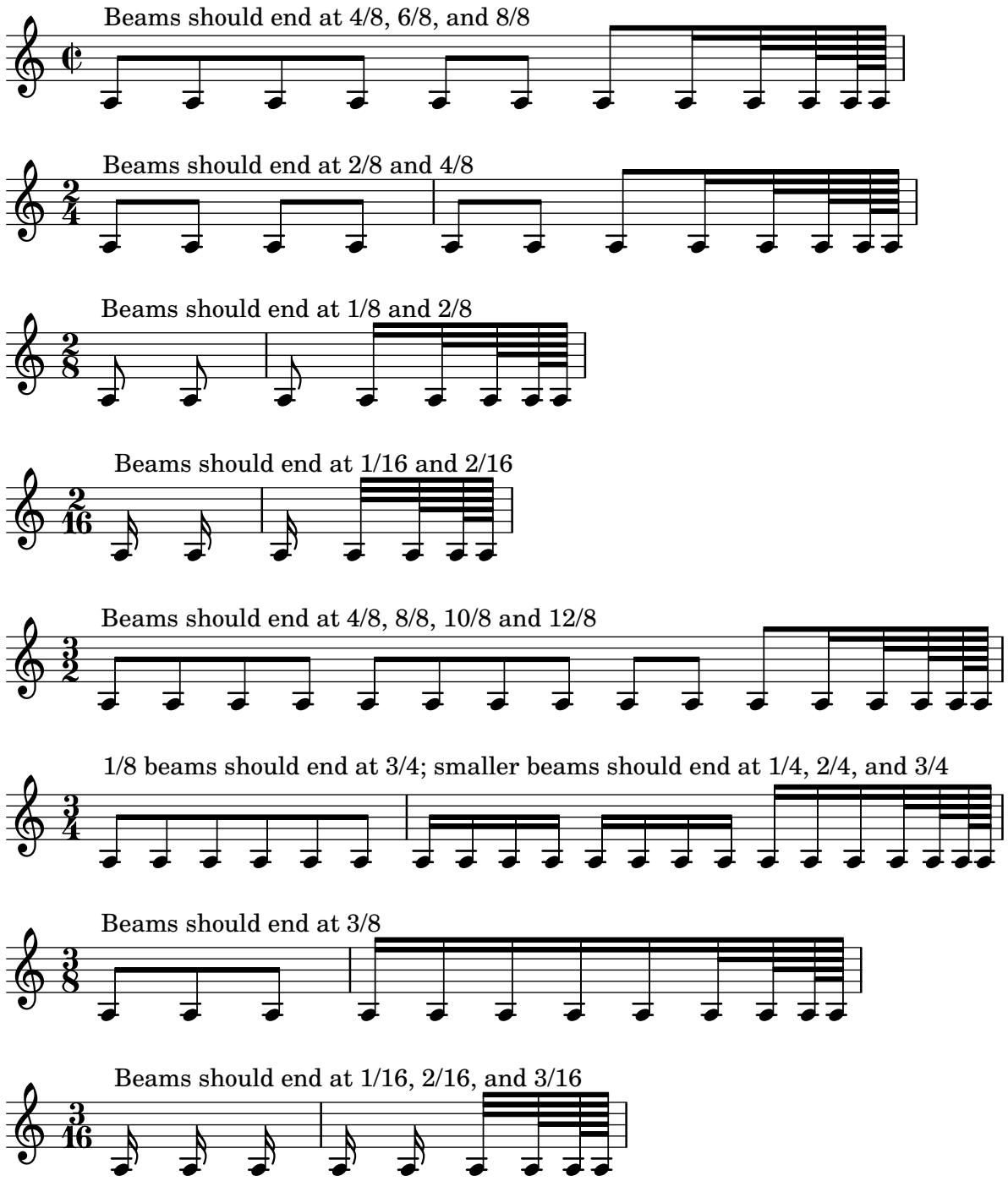


`\noBeam` should terminate an autobeam, even if it's not a recommended place for stopping a beam. In this example, the first three eighth notes should be beamed.
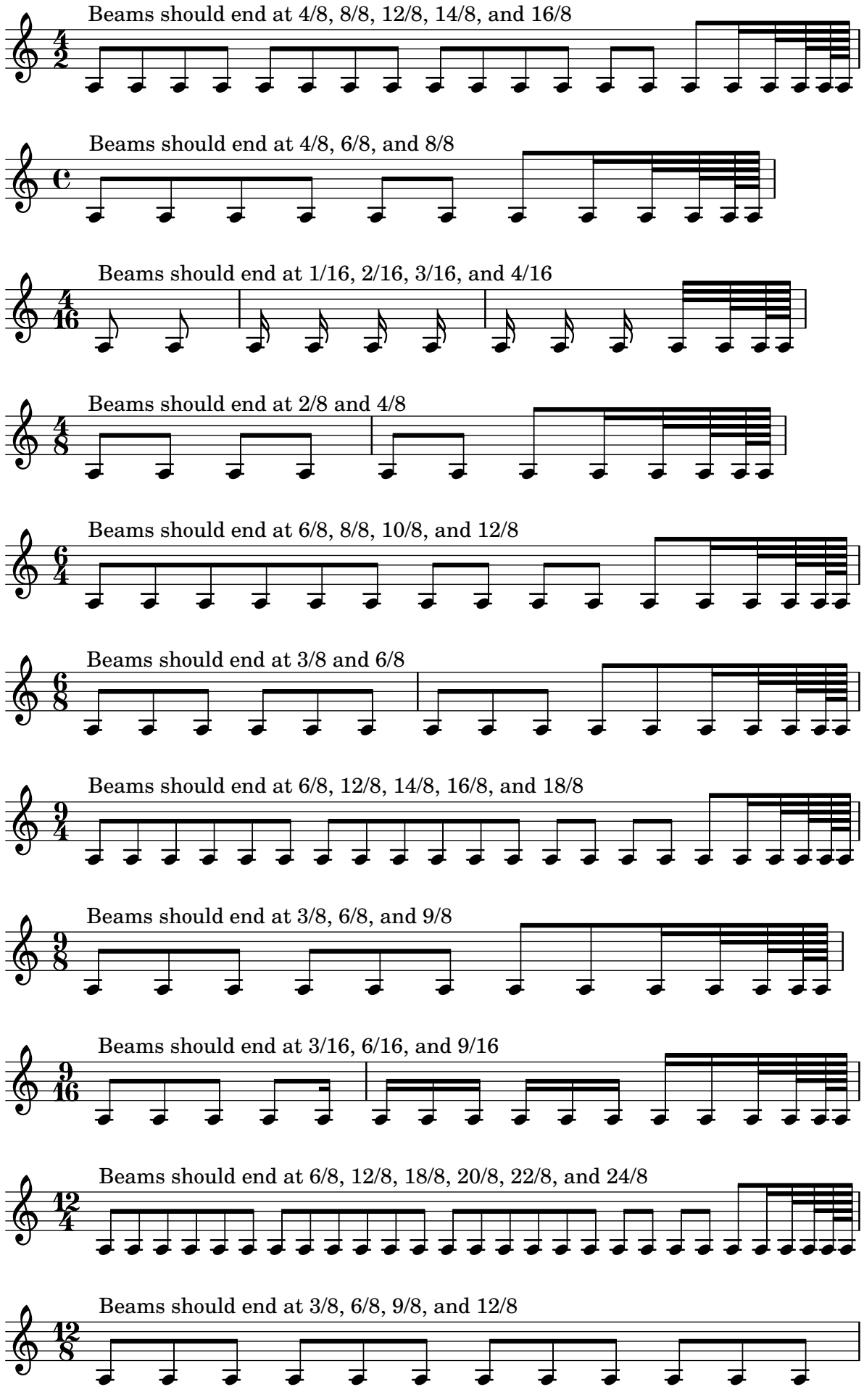
`autobeam-nobeam.ly`

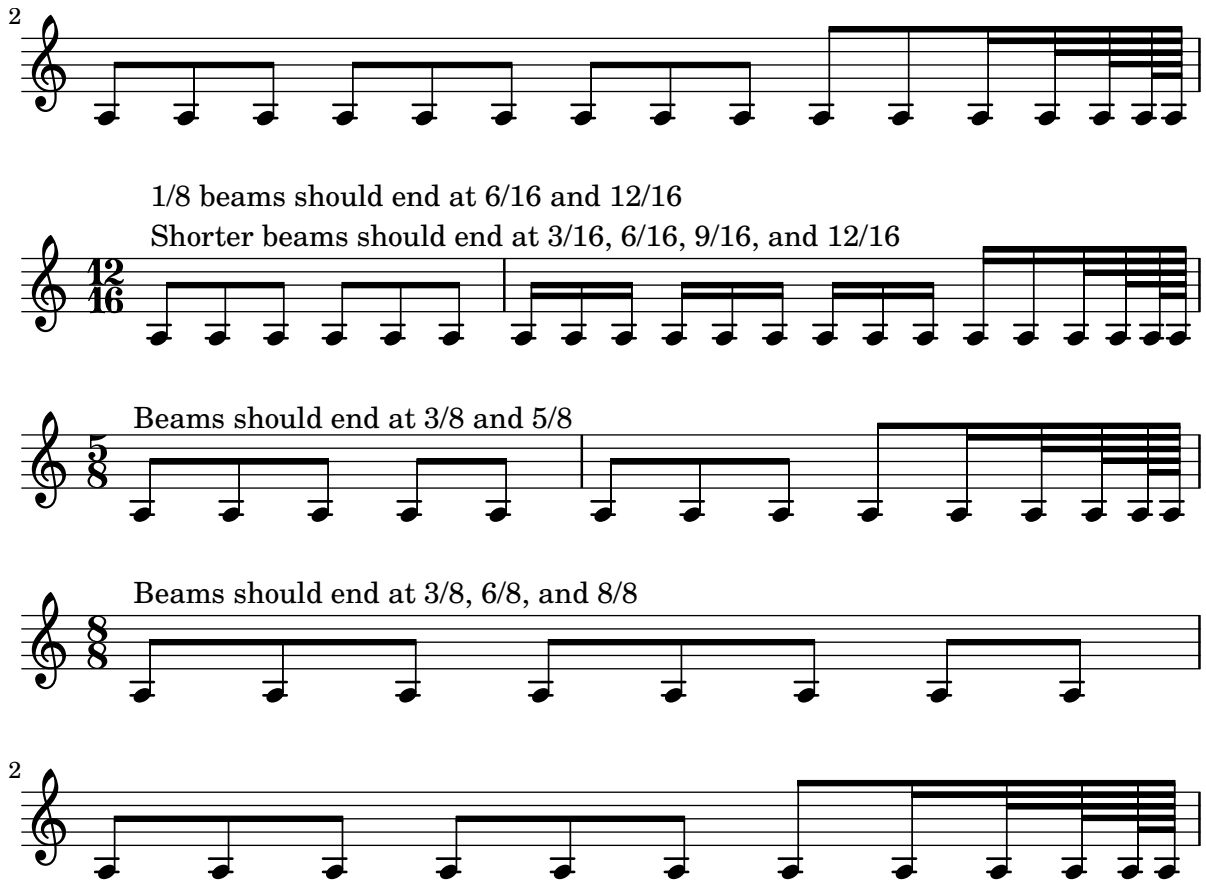


Default autobeam settings have been set for a number of time signatures. Each score shows the desired beaming

`autobeam-show-defaults.ly`
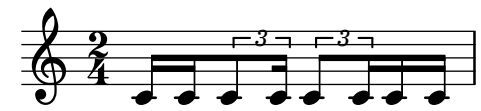



Beams should end at 4/8, 8/8, 12/8, 14/8, and 16/8
Beams should end at 4/8, 6/8, and 8/8
Beams should end at 1/16, 2/16, 3/16, and 4/16
Beams should end at 2/8 and 4/8
Beams should end at 6/8, 8/8, 10/8, and 12/8
Beams should end at 3/8 and 6/8
Beams should end at 6/8, 12/8, 14/8, 16/8, and 18/8
Beams should end at 3/8, 6/8, and 9/8
Beams should end at 3/16, 6/16, and 9/16
Beams should end at 6/8, 12/8, 18/8, 20/8, 22/8, and 24/8
Beams should end at 3/8, 6/8, 9/8, and 12/8
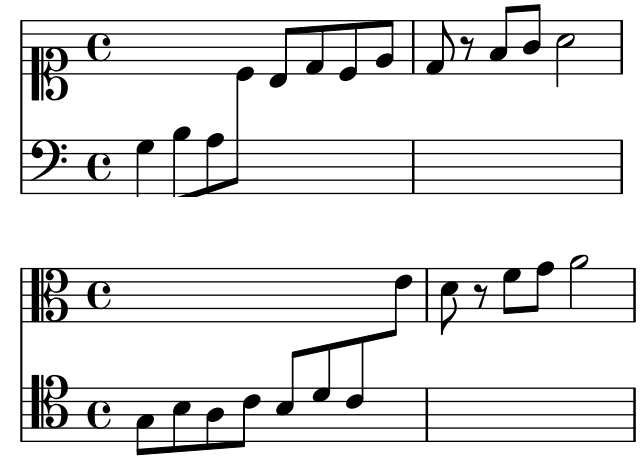


Autobeam rechecking works properly with tuplets. In the example, the first beat should be beamed completely together.

`autobeam-tuplet-recheck.ly`



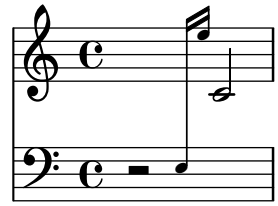
Other clefs for the autoChanger may be set. This works for implicitly created staves only. The first example should turn at b with soprano-clef in the upper Staff. The second example should turn at d' with alto-clef in the upper and tenor-clef in the lower Staff.

`autochange-clefs.ly`



Grace notes are placed on the appropriate staff.
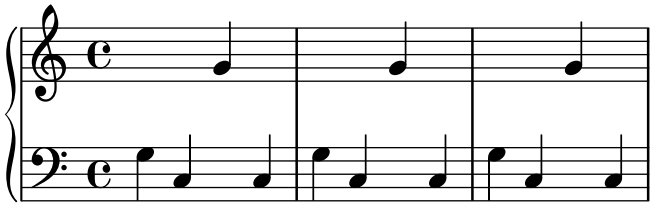
`autochange-inside-grace.ly`



`\autoChange` needs to be given pitches in their final octaves, so if `\relative` is used it must be applied inside `\autoChange`. The pitches in `\autoChange` are unaffected by an outer `\relative`, so that the printed output shows the pitches that `\autoChange` used.

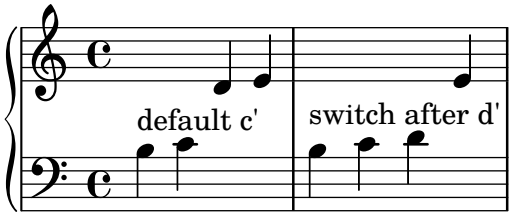
The expected output of this test is three identical measures.

`autochange-relative.ly`



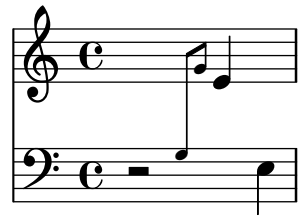
Other turning points for the autoChanger are possible.

`autochange-turning-pitch.ly`
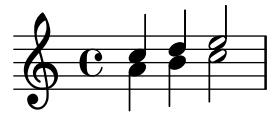


Grace notes are placed on the appropriate staff.

`autochange-with-grace.ly`



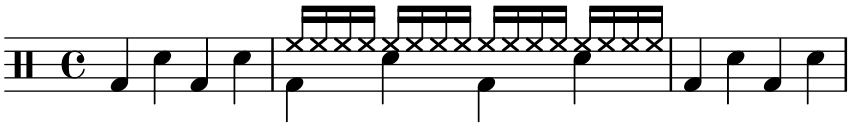
The bottom-level contexts in polyphony shorthand are allocated a context id in order of creation, starting with `"1"`. This snippet will fail to compile if either voice has an invalid `context-id` string.

`automatic-polyphony-context-id.ly`



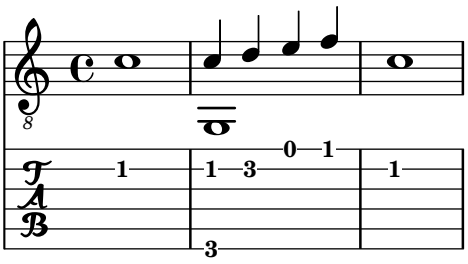
In a DrumStaff, automatic polyphony can be used without explicitly initializing separate voices.

`automatic-polyphony-drumstaff.ly`



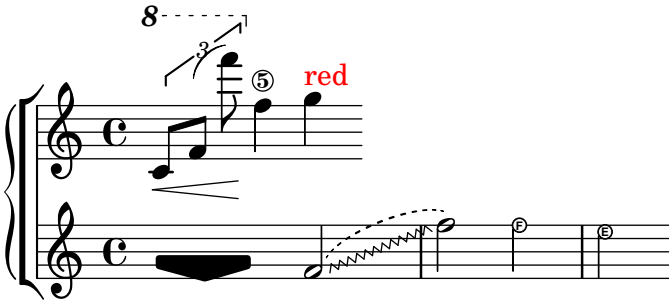
In a TabStaff, automatic polyphony can be used without explicitly initializing separate voices.

`automatic-polyphony-tabstaff.ly`



Exercise all output functions

`backend-excercise.ly`



`backend-svg.ly`

The Bärenreiter edition of the Cello Suites is the most beautifully typeset piece of music in our collection of music (we both own one. It is also lovely on French Horn). This piece does not include articulation, but it does follows the same beaming and linebreaking as the printed edition. This is done in order to benchmark the quality of the LilyPond output.

As of lilypond 1.5.42, the spacing and beam quanting is almost identical.

There are two tweaks in this file: a line-break was forced before measure 25, we get back the linebreaking of Bärenreiter. The stem direction is forced in measure 24. The last beam of that measure is up in Bärenreiter because of context. We don't detect that yet.
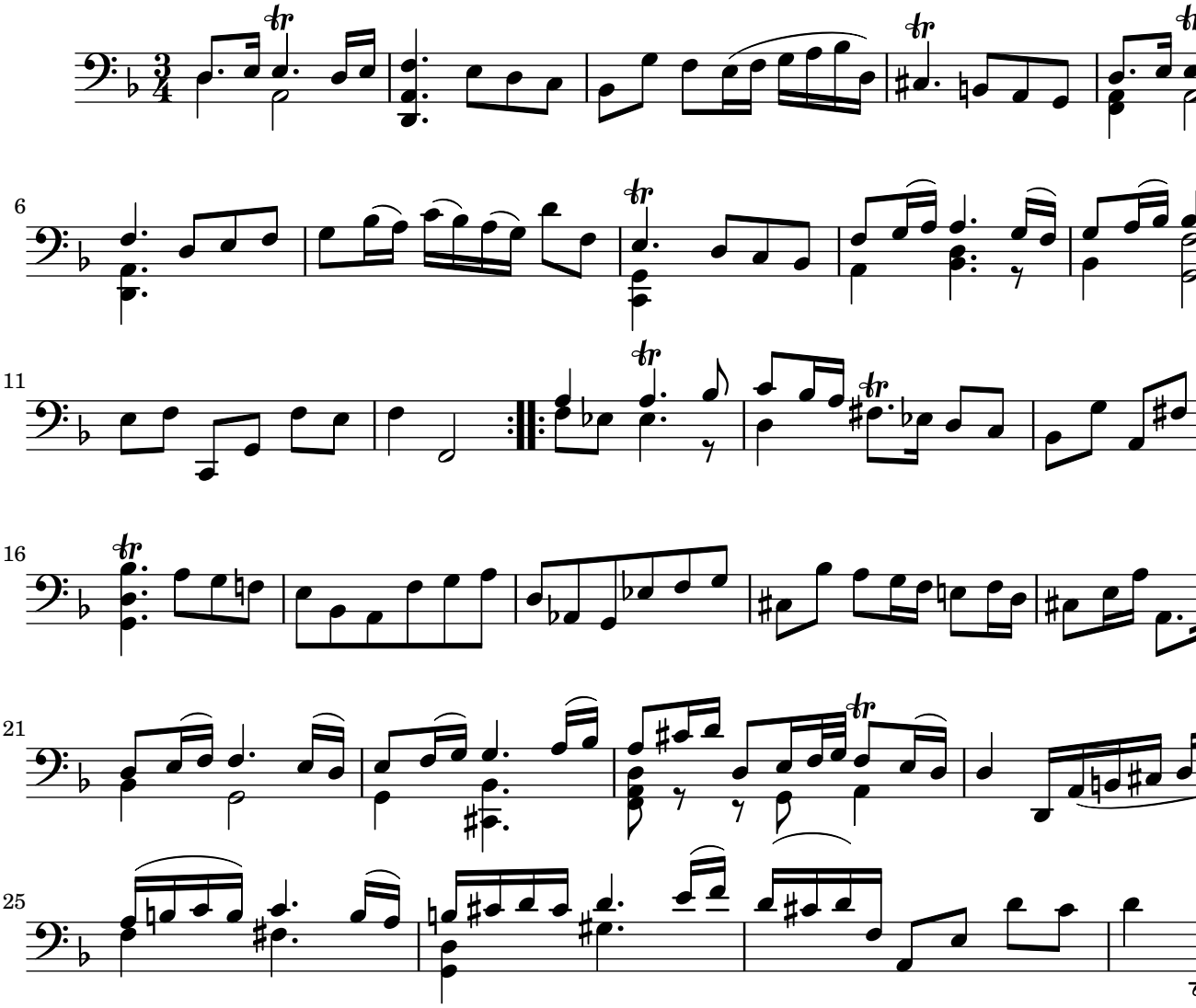
Note that the Bärenreiter edition contains a few engraving mistakes. The second line begins with measure 6 (but prints 5). The |: half way in measure 13 has been forgotten.

baerenreiter-sarabande.ly

Solo Cello Suite II

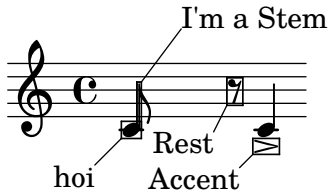
Johann Sebastian Bach (16

Sarabande
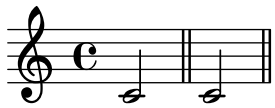


With balloon texts, objects in the output can be marked, with lines and explanatory text added.

`balloon.ly`



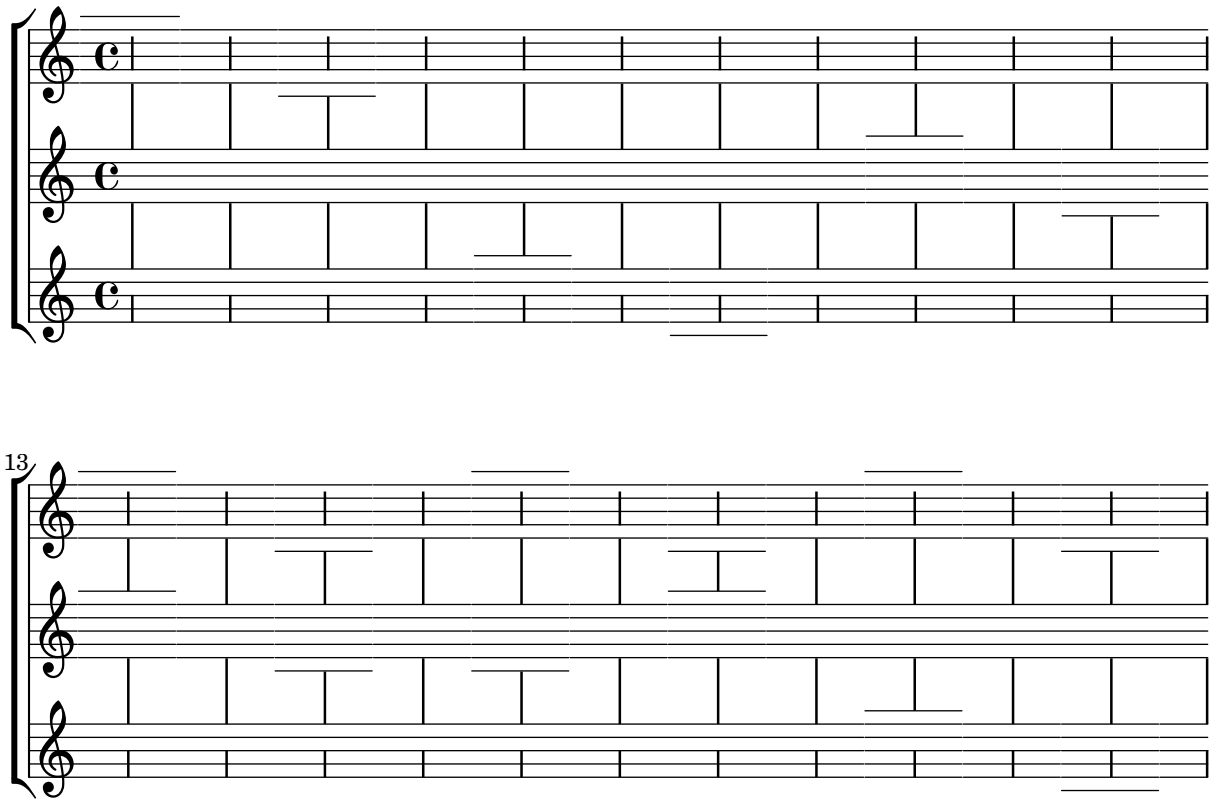
The meaning of `|` is stored in the identifier `"|"`.

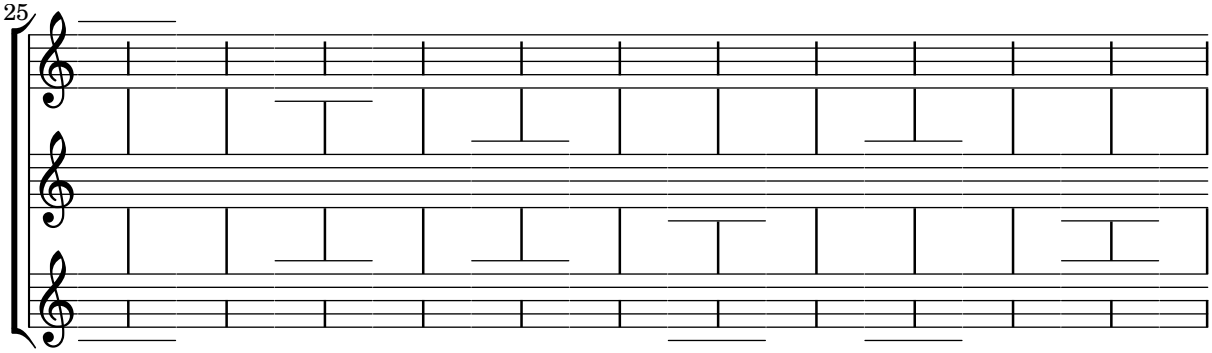
`bar-check-redefine.ly`



Bar line extent can be customised and the customised value must be respected when staff symbol is changed temporarily (e.g. to simulate ledger lines of renaissance prints and manuscripts); moreover, span bars should not enter the staves.

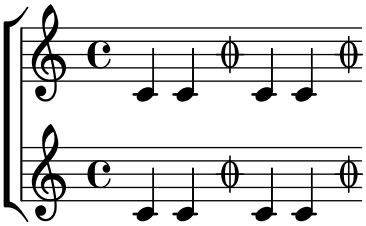
`bar-extent.ly`

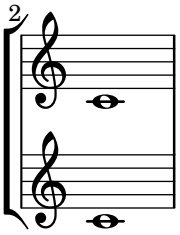




New bar line glyphs can be defined in Scheme.

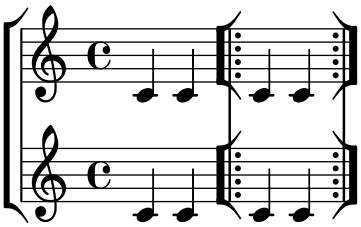
`bar-line-define-bar-glyph.ly`

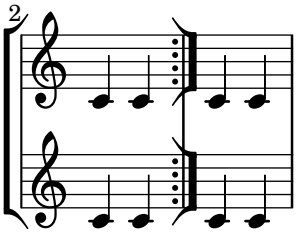


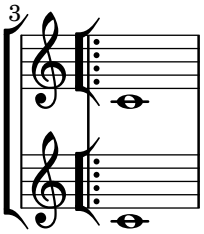


New bar line styles can be defined by `\defineBarLine`.

`bar-line-define-bar-line.ly`

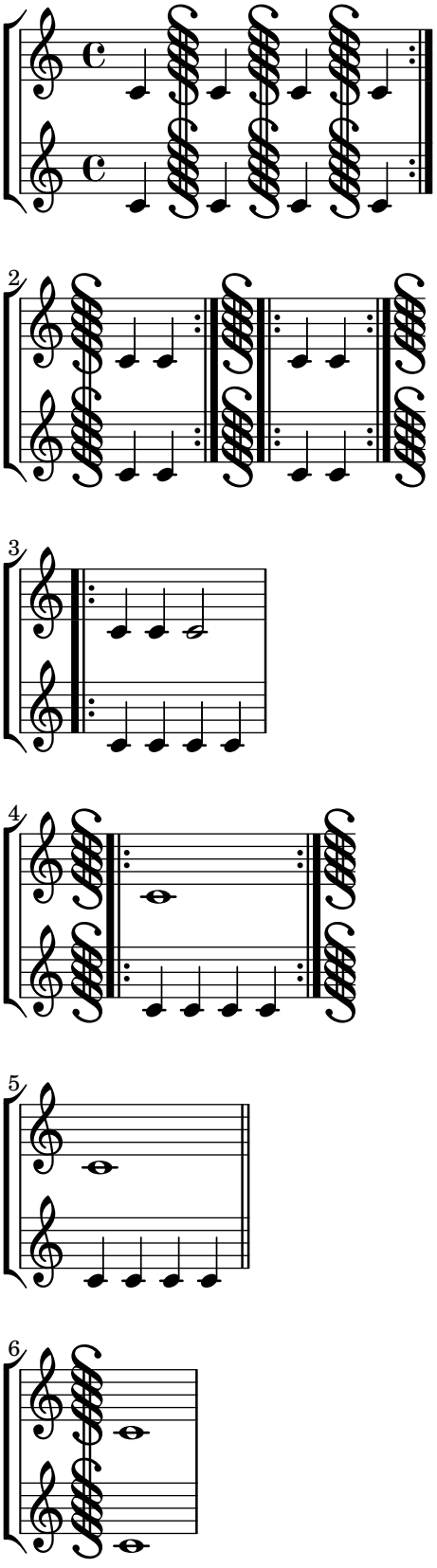






Segno bar lines can be used to mark the begin and the end of a segno part.

`bar-line-segno.ly`



Various types of bar lines can be drawn.

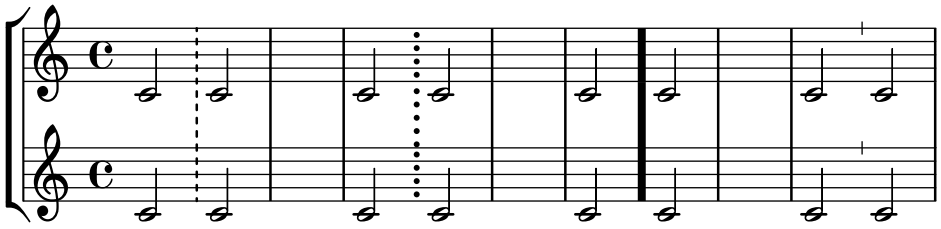
The dashes in a dashed bar line covers staff lines exactly. Dashed barlines between staves start and end on a half dash precisely.

The dots in a dotted bar line are in spaces.

A thick bar line is created by \bar ".", which is consistent with e.g. \bar "|."

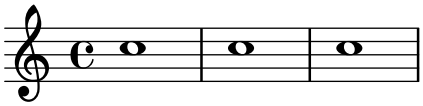
A ticked bar line is a short line of the same length as a staff space, centered on the top-most barline.

`bar-lines.ly`



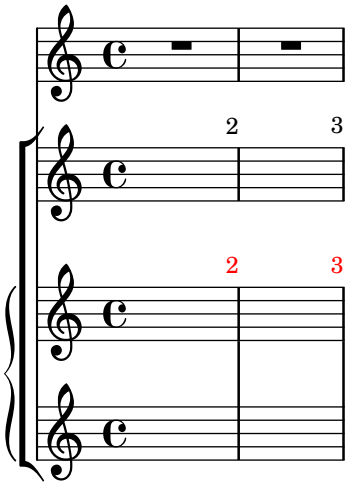
`\barNumberCheck` may be inserted to check whether the current bar number is correct. Checking is enabled by default for layout and disabled by default for MIDI.

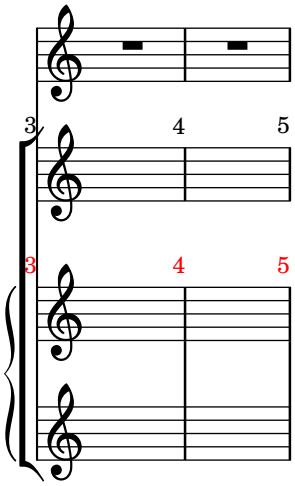
`bar-number-check-warning.ly`



Bar_number_engraver may be moved to staff-group contexts. Bar numbers should appear in black above the second staff from the top. The same numbers should appear in red above the third staff from the top.

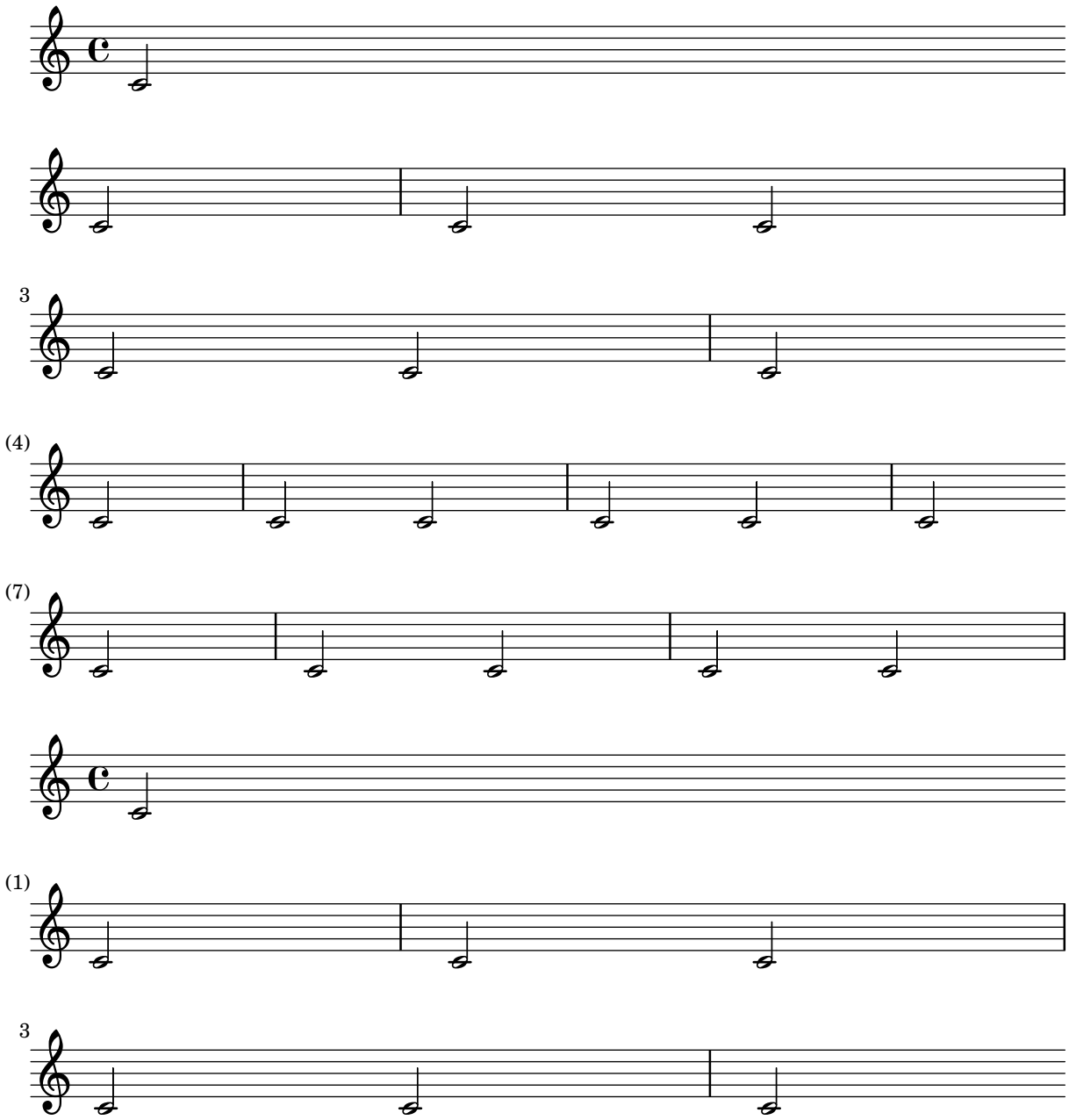
`bar-number-staff-group-context.ly`

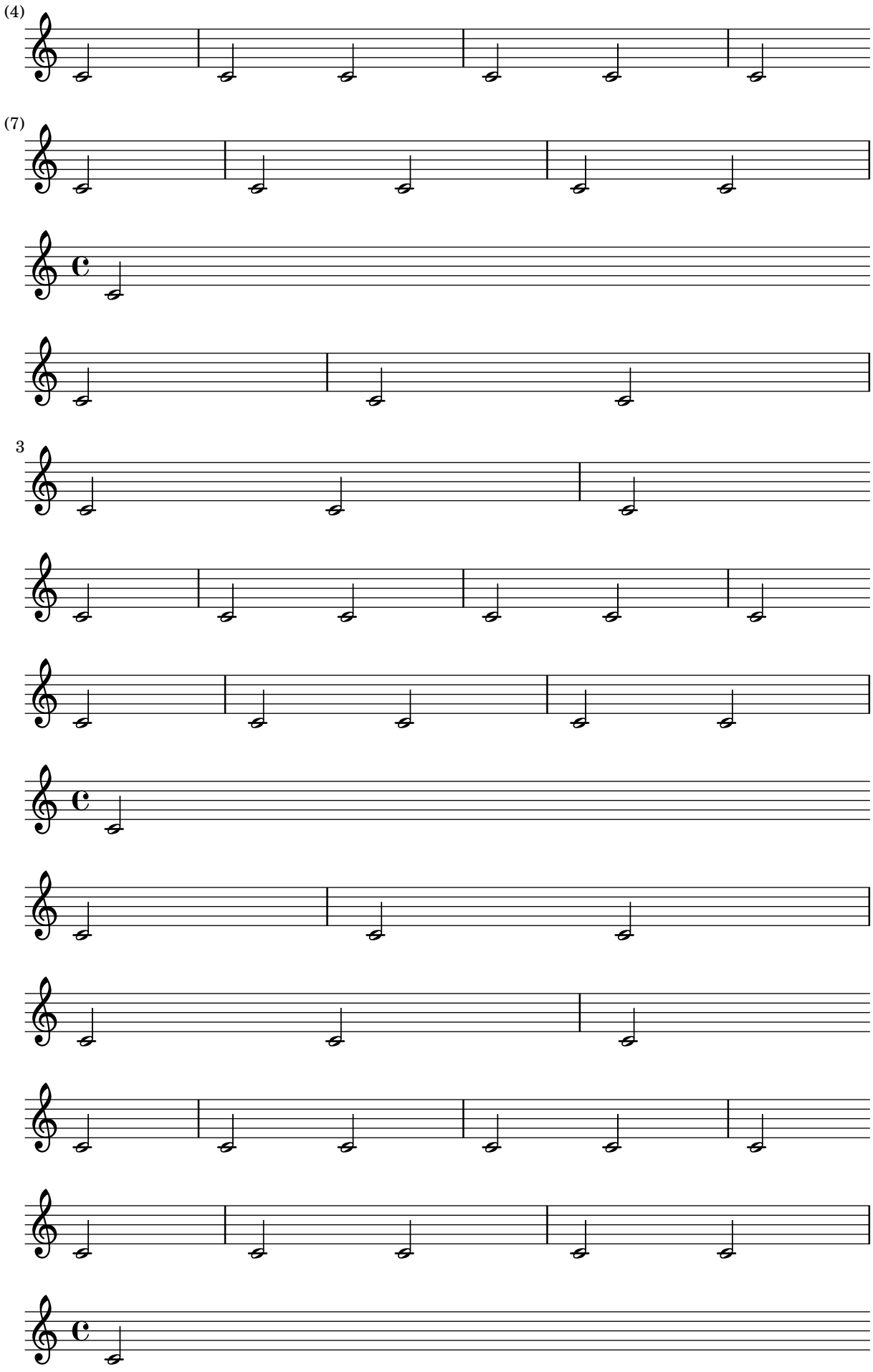


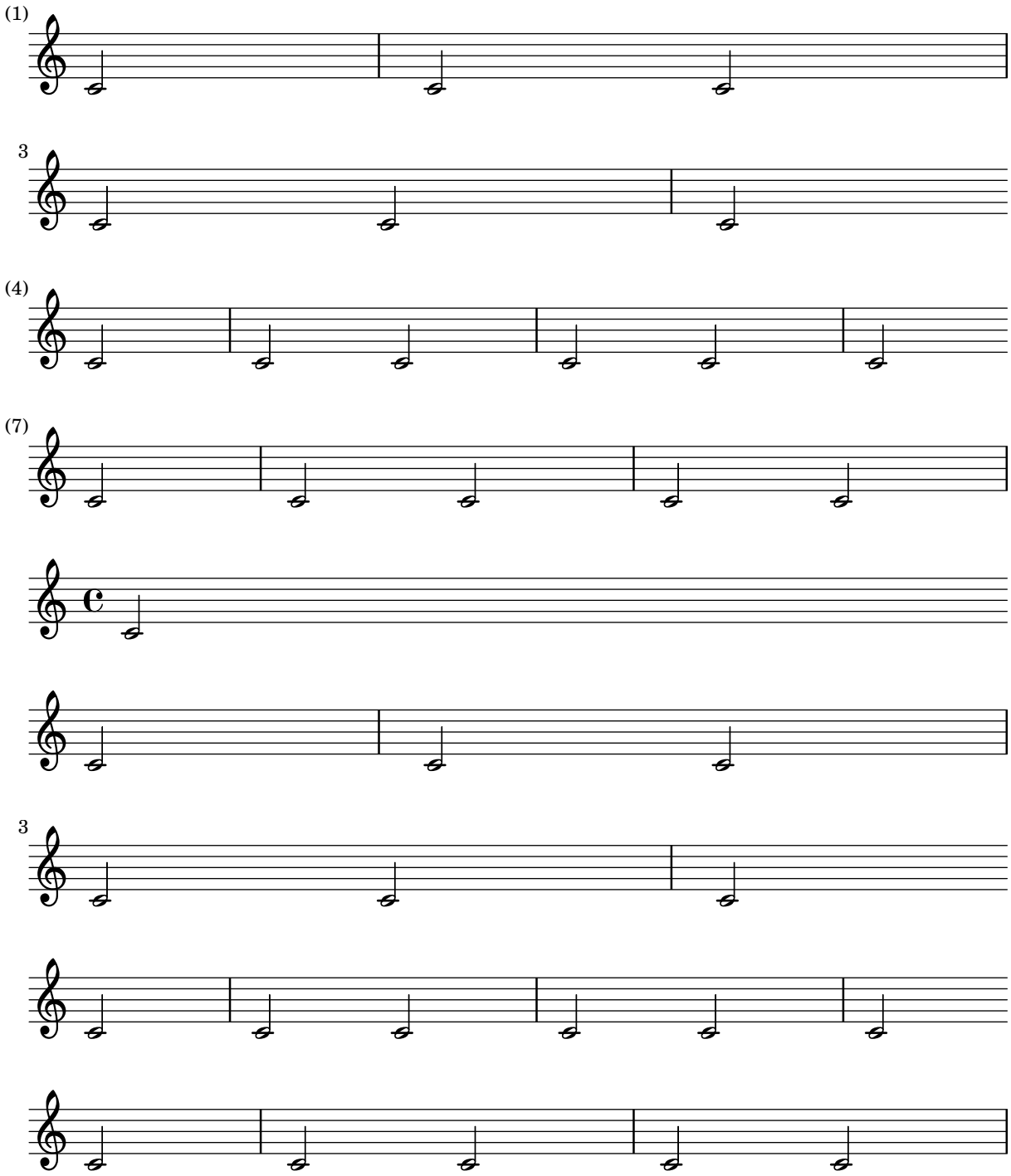


The `barNumberVisibility` property controls at what intervals bar numbers are printed.

`bar-number-visibility.ly`

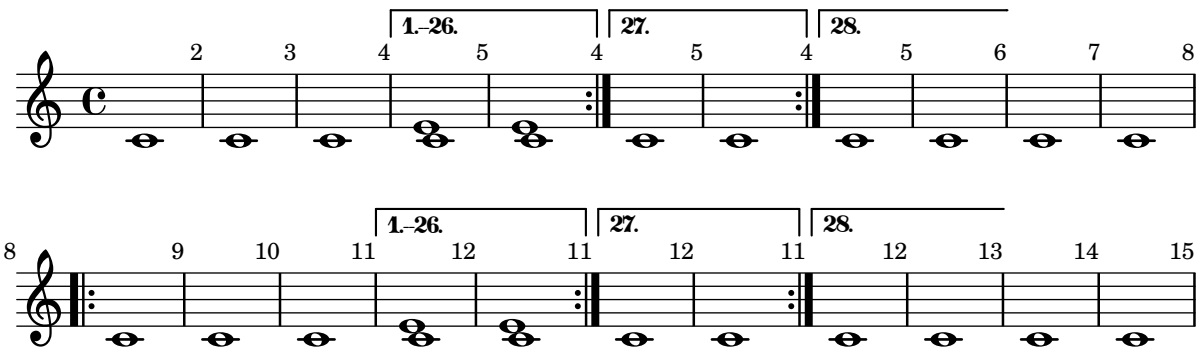


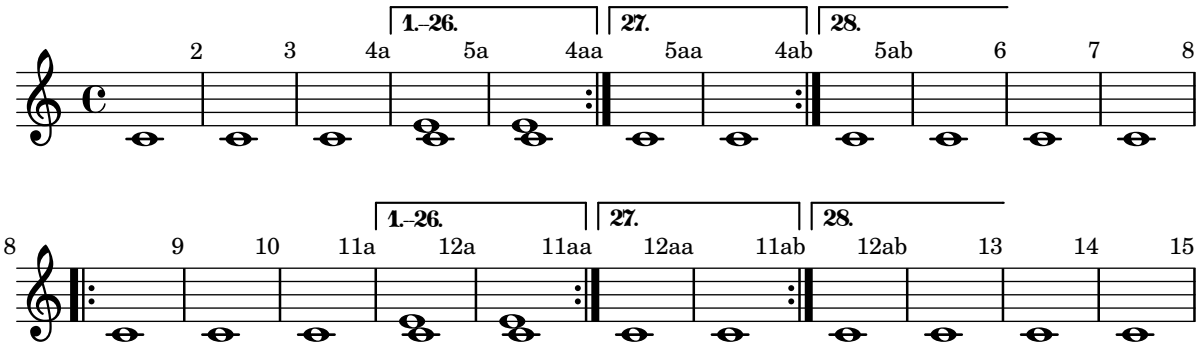

(4)
(7)
3



Bar numbers can automatically reset at volta repeats.

`bar-number-volta-repeat.ly`

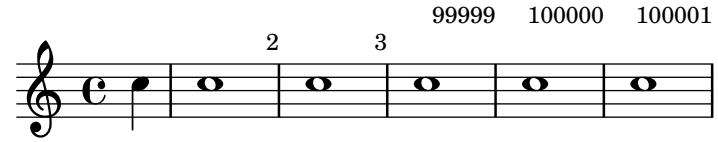




Bar numbers may be set and their padding adjusted individually. The counting of bar numbers is started after the anacrusis.

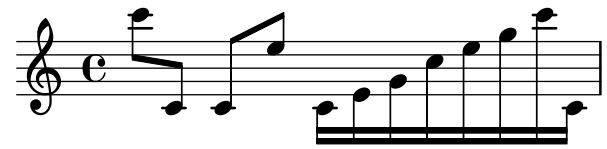
To prevent clashes at the beginning of a line, the padding may have to be increased.

`bar-number.ly`



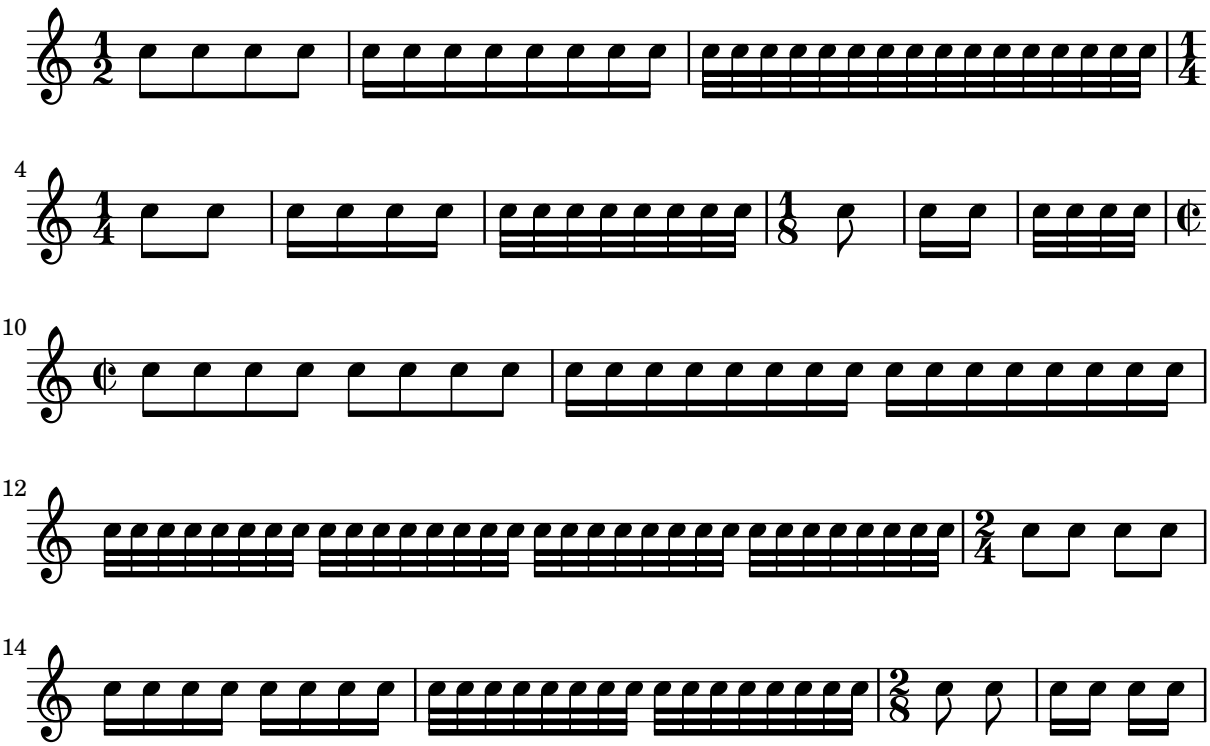
A knee is made automatically when a horizontal beam fits in a gap between note heads that is larger than a predefined threshold.

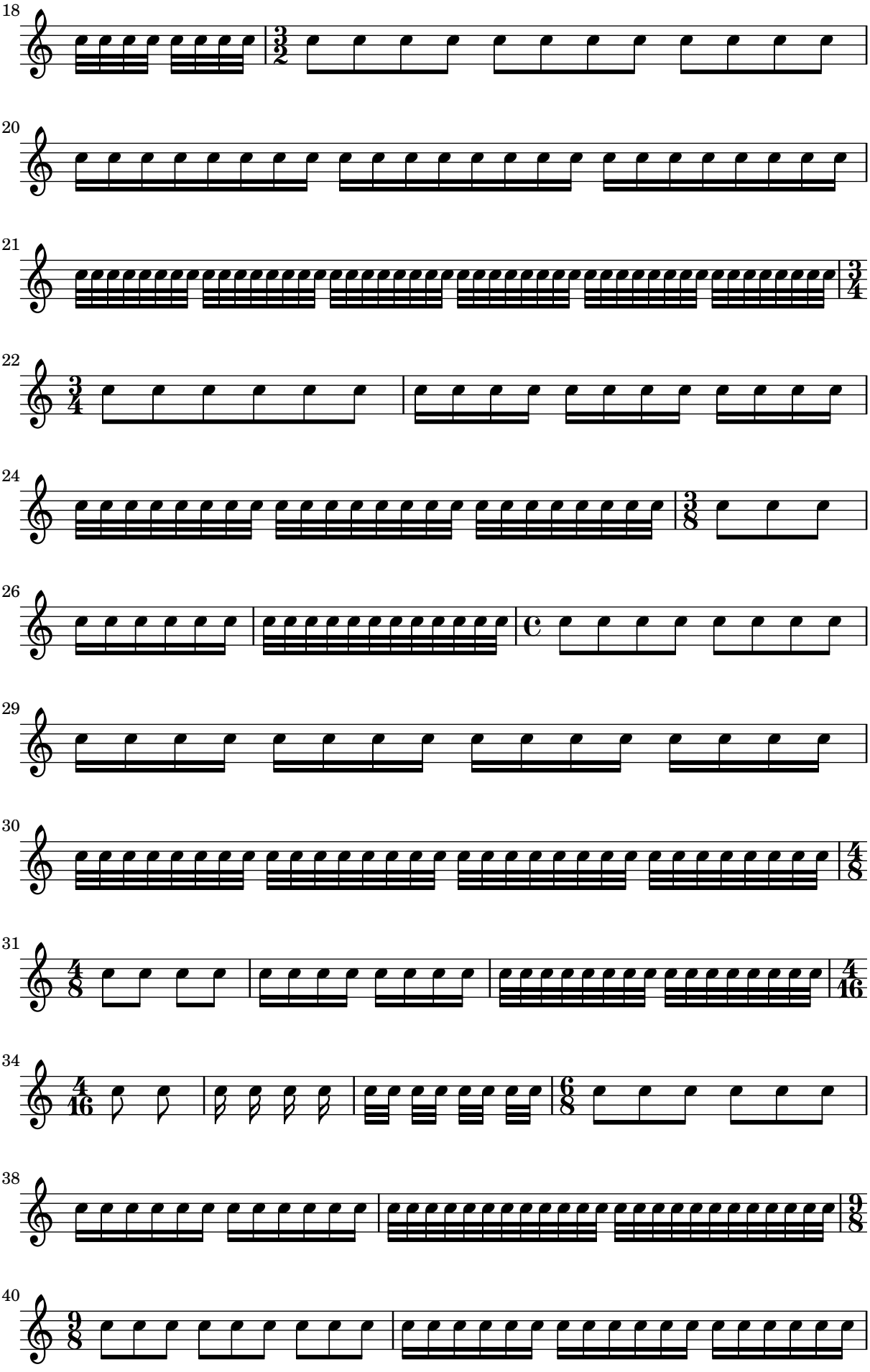
`beam-auto-knee.ly`

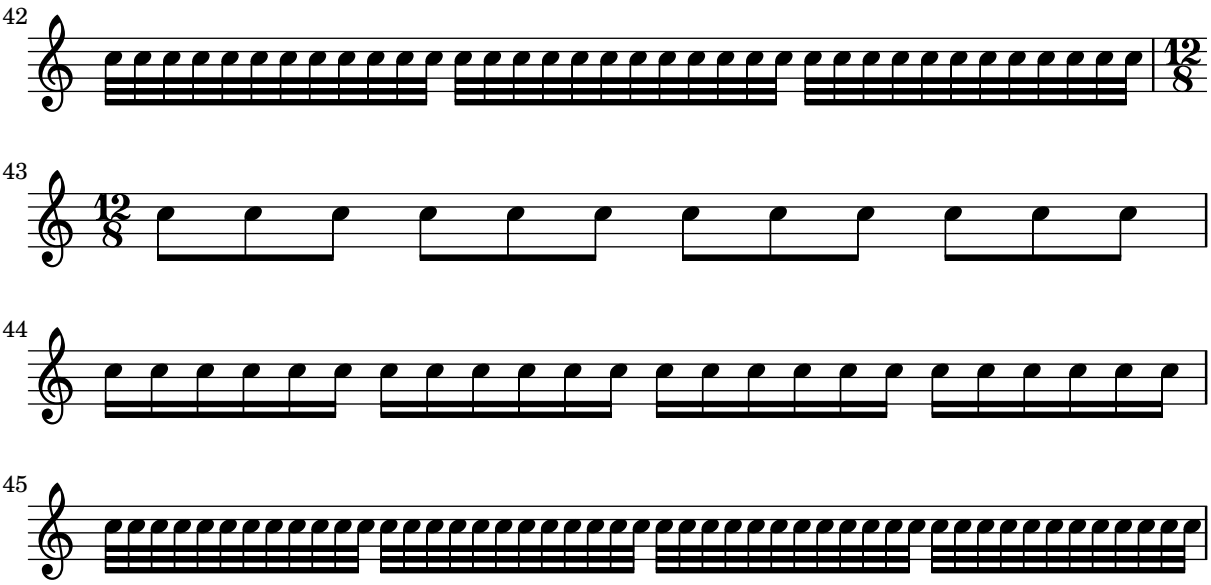


There are presets for the `auto-beam` engraver in the case of common time signatures.

`beam-auto.ly`

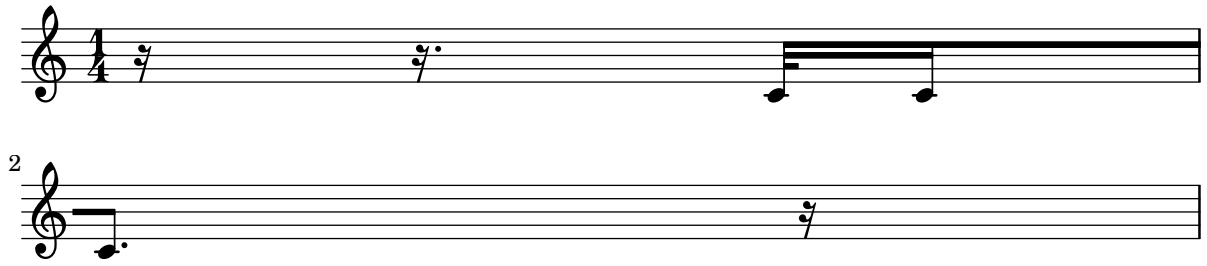






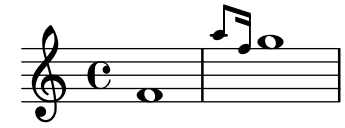
beamlets don’t run to end of line if there are no other beamlets on the same height.

`beam-beamlet-break.ly`
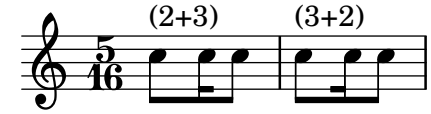


Beamlets in grace notes remain readable.

`beam-beamlet-grace.ly`



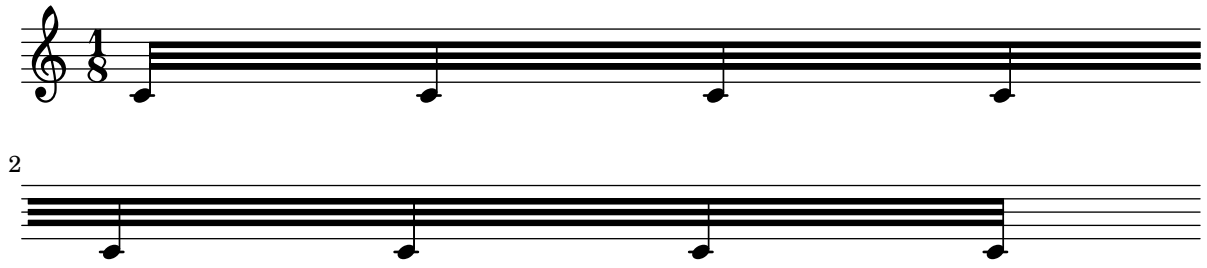
Default beaming patterns can be set for the current time signature.

`beam-beat-grouping.ly`



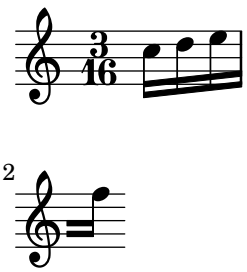
Broken beams have sane endings even if grobs are not present at the broken end.

`beam-break-no-bar.ly`



Beams can be printed across line breaks, if forced.

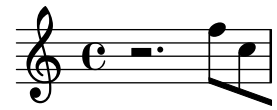
beam-break.ly
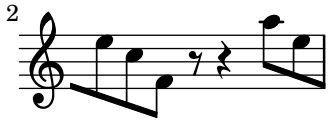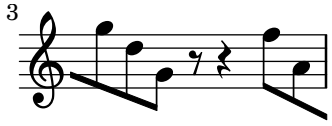


Some classic examples of broken beams, all taken from Scriabin Op. 11, No. 1.
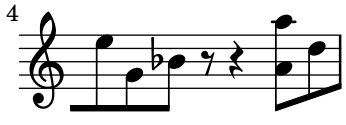
beam-broken-classic.ly
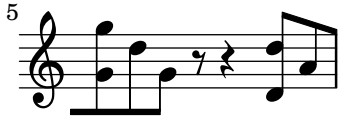
\override Beam.positions = #beam::place-broken-parts-individually (default)
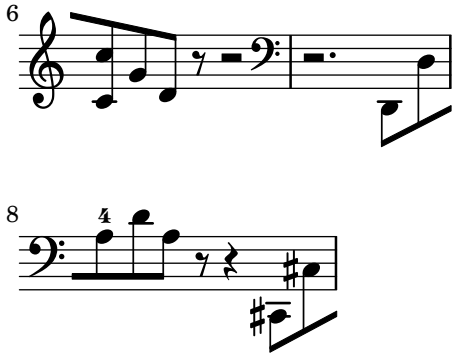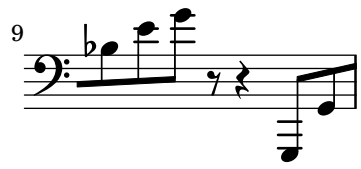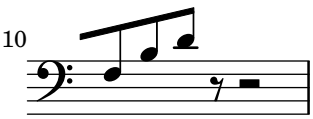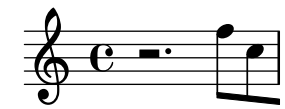












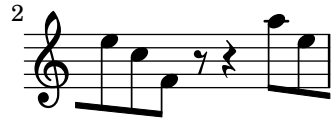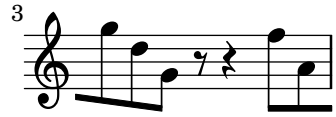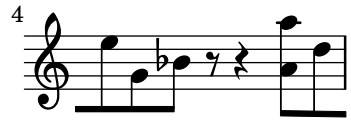




\override Beam.positions = #beam::align-with-broken-parts

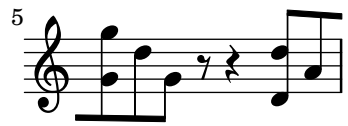
Returns y-positions at the ends of the beam such that beams align-across-breaks.
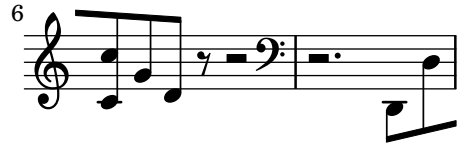


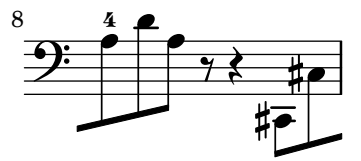


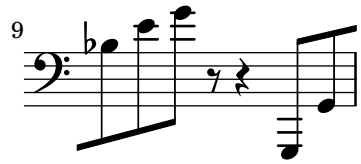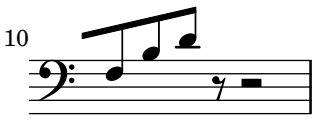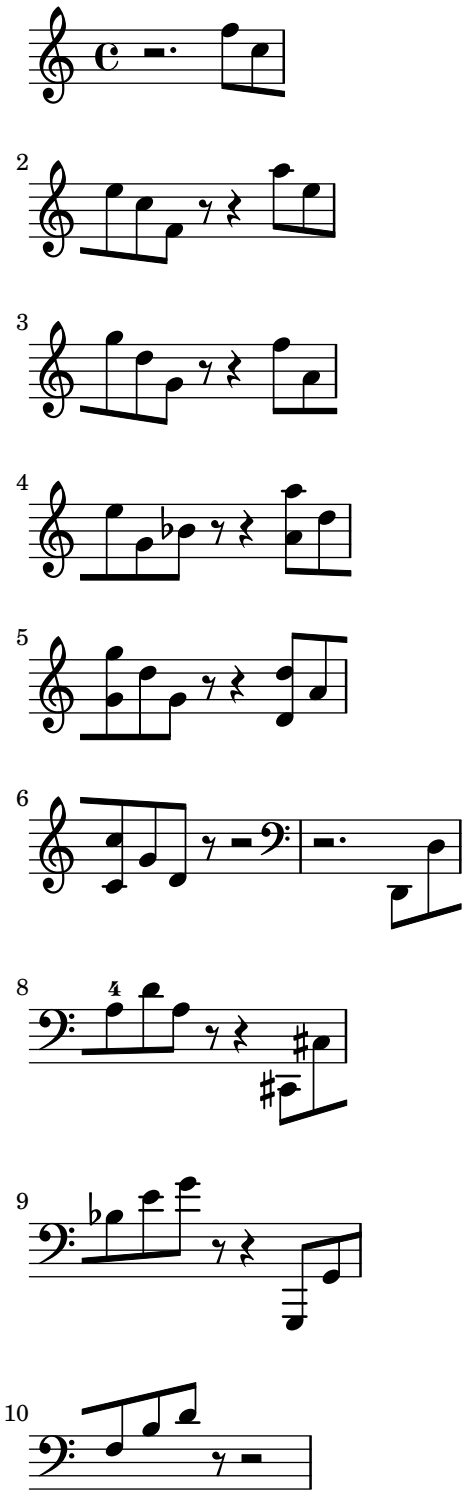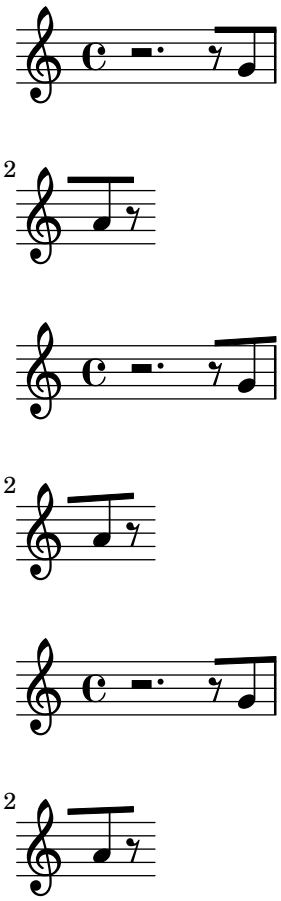














\override Beam.positions = #beam::slope-like-broken-parts

Approximates broken beam positioning in turn-of-the-century Editions Peters scores.



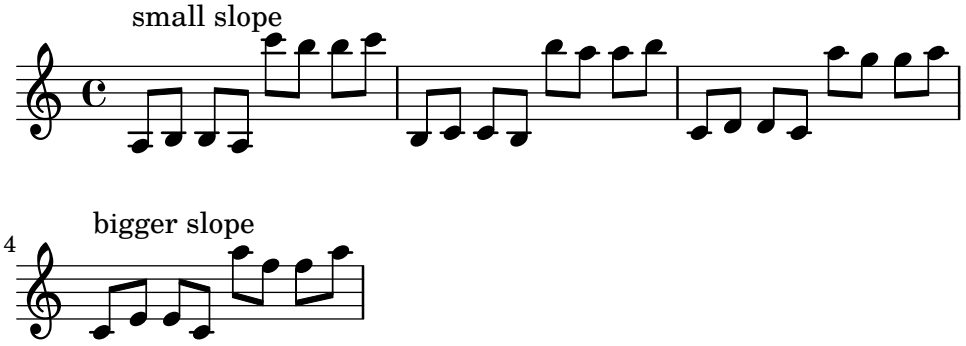
The functions passed to the `positions` property should handle complicated cases in the same manner that they handle more normal cases.

`beam-broken-difficult.ly`
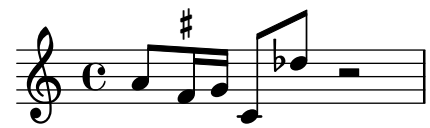


Simple beams on middle staffline are allowed to be slightly sloped, even if the notes have ledgers. Beams reaching beyond middle line can have bigger slope.

`beam-center-slope.ly`
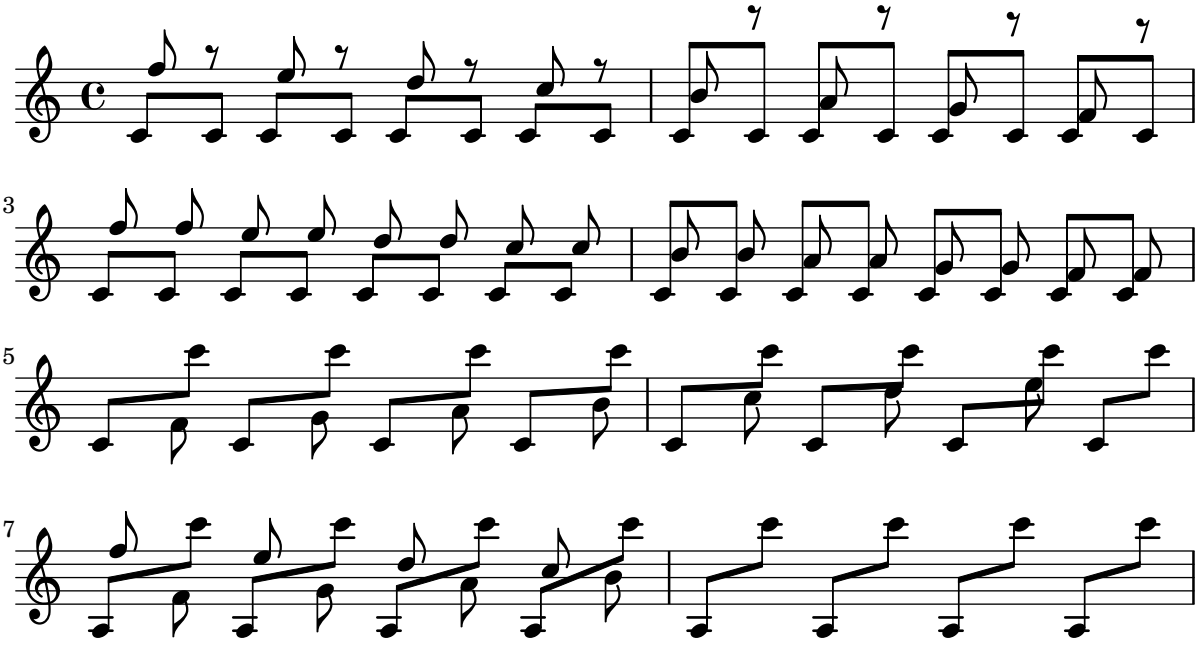


Beams only check for collisions with in-line accidentals.

`beam-collision-accidentals.ly`
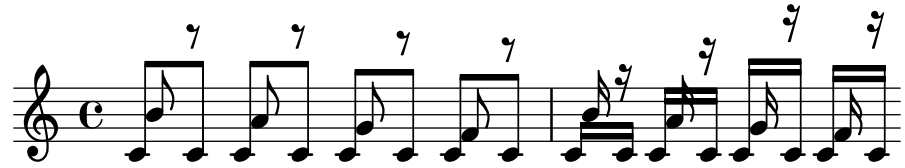


Manual beams do not collide with notes.

`beam-collision-basic.ly`
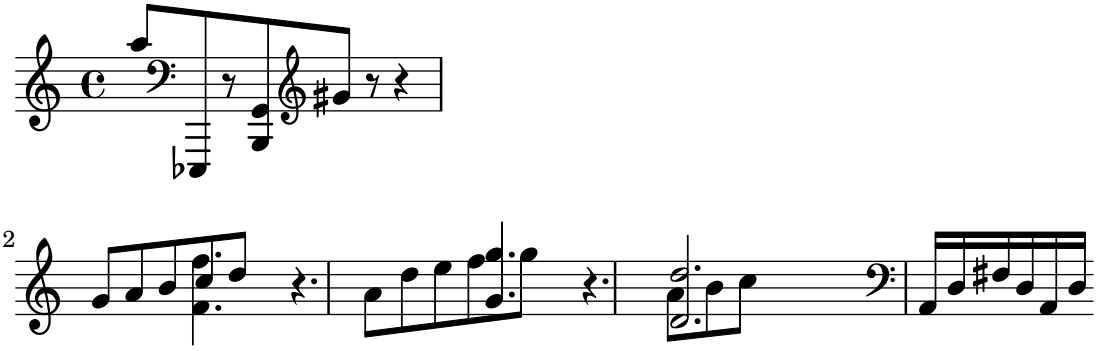


Manual beams do not collide with notes.

`beam-collision-beamcount.ly`



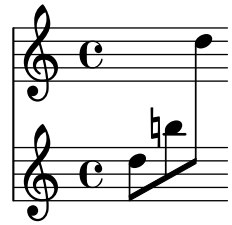
Beam collisions from modern works

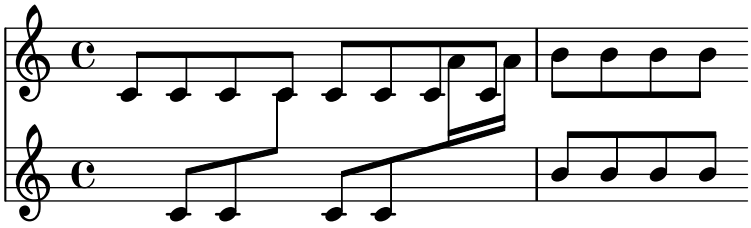
`beam-collision-classic.ly`



cross staff beams work with collisions.

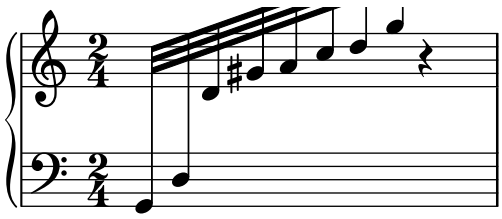
`beam-collision-cross-staff.ly`





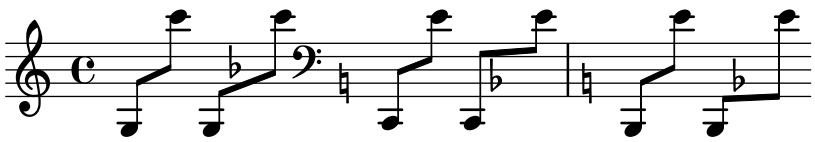
Cross staff beams do collision avoidance.

`beam-collision-cross-staff2.ly`
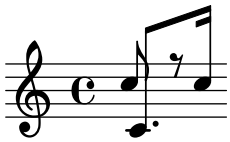


A rough guess for collisions is taken into account when choosing initial beam configurations; the initial position may be chosen to be either above or below large collisions.

`beam-collision-feasible-region.ly`
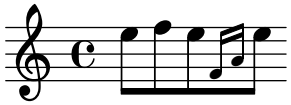


Beams do not collide with flags.

`beam-collision-flag.ly`



The beaming algorithm handles collisions between beams and grace notes too.

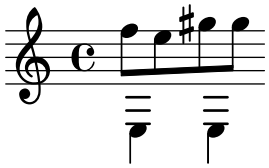
`beam-collision-grace.ly`
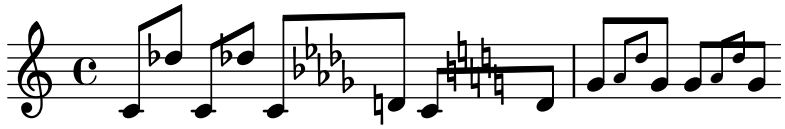




Behave sensibly in the presence of large collisions.

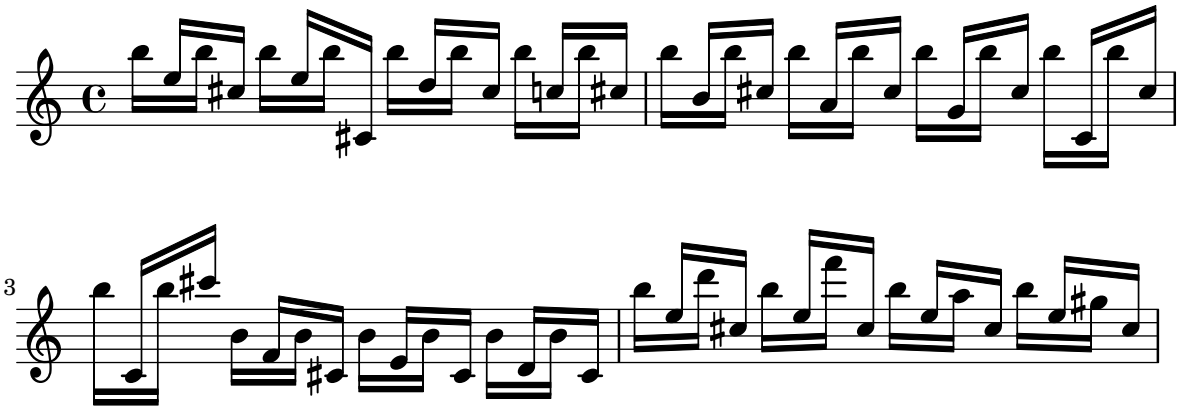
`beam-collision-large-object.ly`
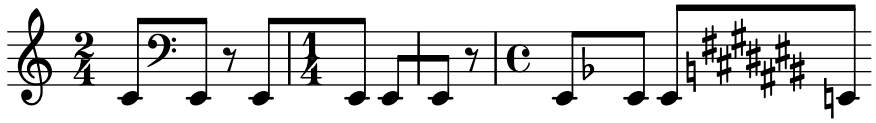


Beams can be allowed to collide with grobs by overriding the collision-interfaces property.
`beam-collision-off.ly`
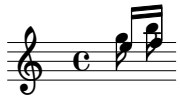


Meshing stems in oppositely directed beams are handled correctly.
`beam-collision-opposite-stem.ly`
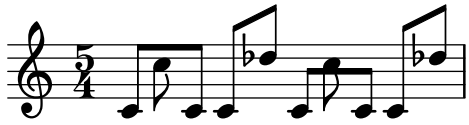


Beams do not collide with clefs, key signatures, time signatures
`beam-collision-prefatory-matter.ly`



Beam collisions are resistant to scaled down staves.
`beam-collision-scaled-staff.ly`
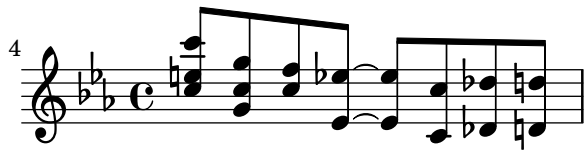


Beam collision can be tweaked to only apply to the grobs within the beam's original voice.
`beam-collision-voice-only.ly`
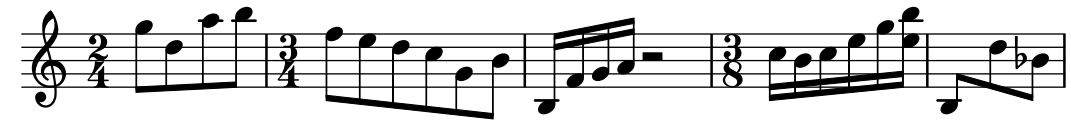


Concave beaming works for chords as well as monophonic music.
`beam-concave-chord.ly`





Beams that are not strictly concave are damped according to their concaveness.
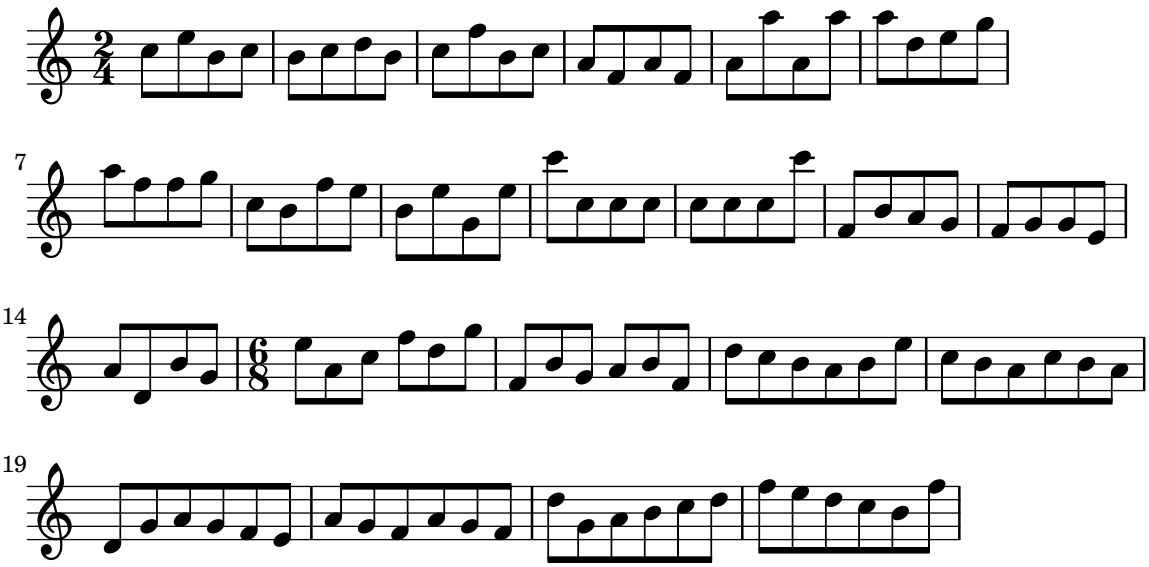
`beam-concave-damped.ly`



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.

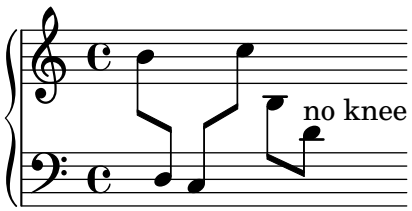
If a beam fails a test, the desired slope is printed next to it.

`beam-concave.ly`



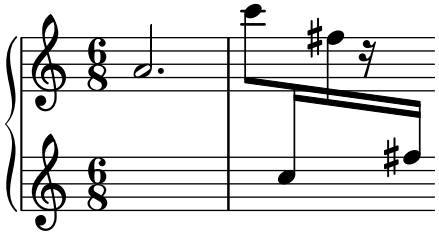
Automatic cross-staff knees work also (here they were produced with explicit staff switches).

`beam-cross-staff-auto-knee.ly`



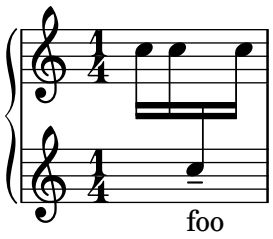
Placement of beamed cross staff rests should be reasonably close to beam.

`beam-cross-staff-rest.ly`



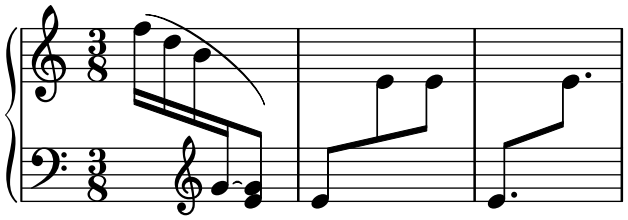
scripts don't trigger beam formatting. If this does happen, we can have a cyclic dependency on Y-positions of staves.

`beam-cross-staff-script.ly`



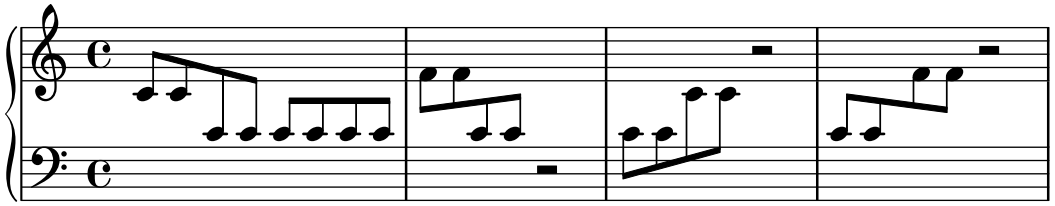
Cross staff (kneed) beams do not cause extreme slopes.

`beam-cross-staff-slope.ly`
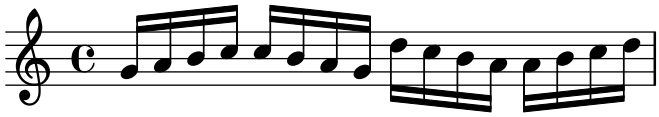


Beams can be typeset over fixed distance aligned staves, beam beautification does not really work, but knees do. Beams should be behave well, wherever the switching point is.

`beam-cross-staff.ly`
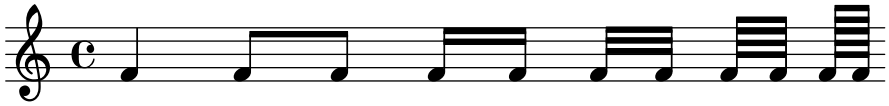


Beams are less steep than the notes they encompass.

`beam-damp.ly`



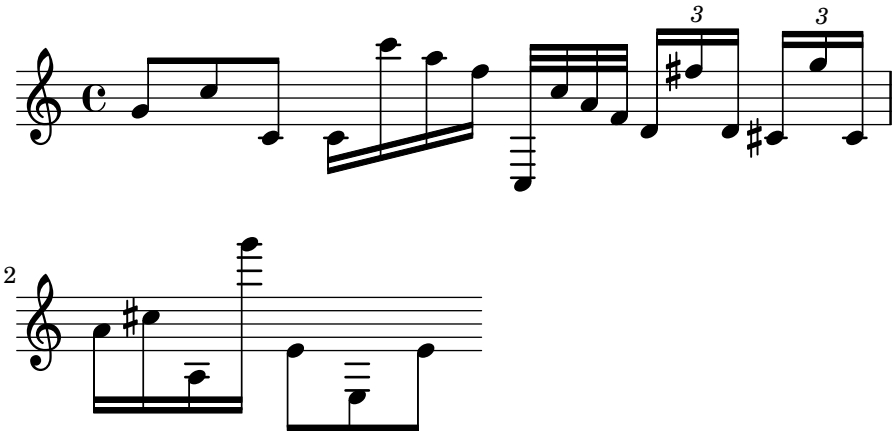
Beamed stems have standard lengths if possible. Quantization is switched off in this example.

`beam-default-lengths.ly`



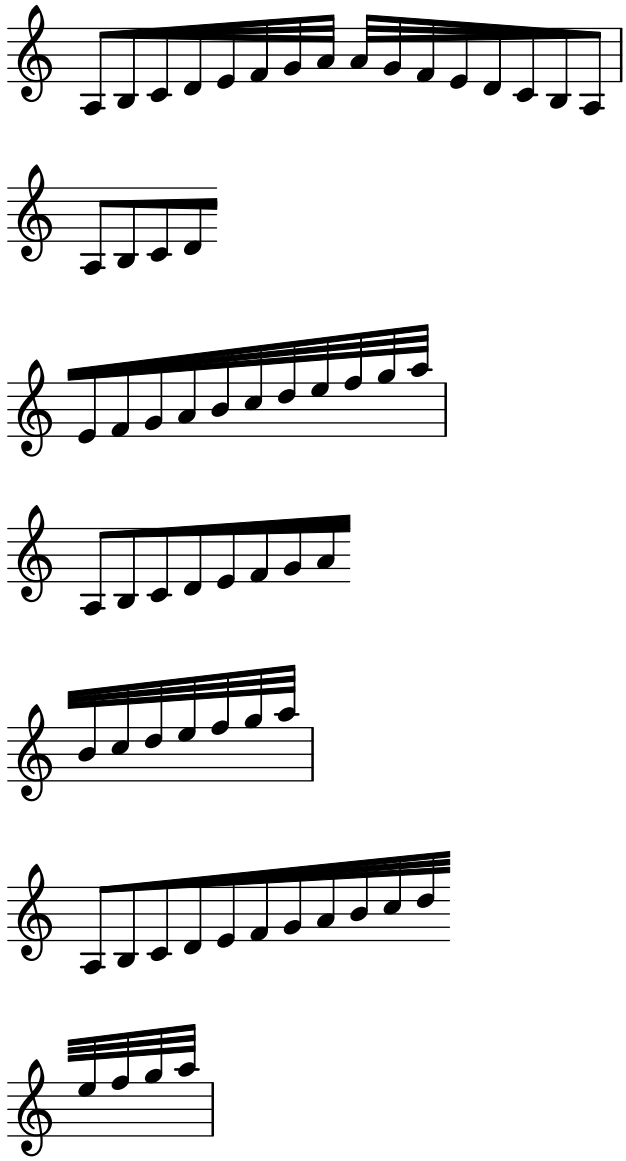
Beams should behave reasonably well, even under extreme circumstances. Stems may be short, but noteheads should never touch the beam. Note that under normal circumstances, these beams would get knees. Here `Beam.auto-knee-gap` was set to false.
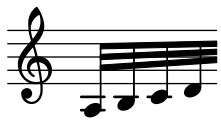
beam-extreme.ly

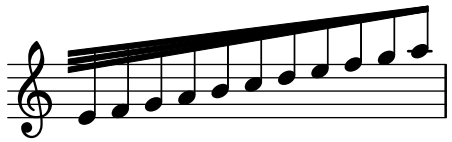


Feathered beams should have the same progress of their feathering at the end of a line break as they do at the beginning of the next line.

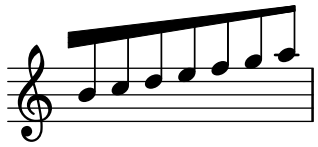
beam-feather-breaking.ly



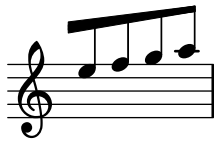


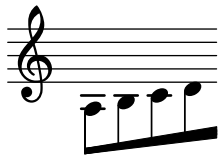


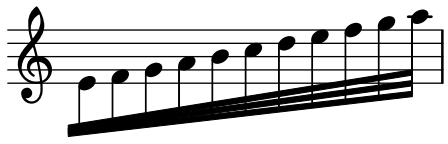


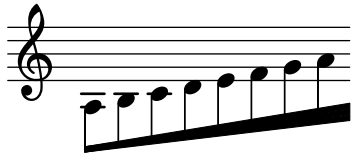




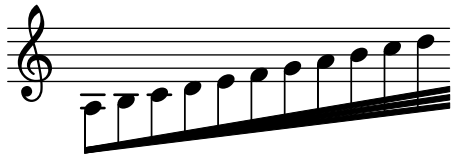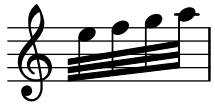




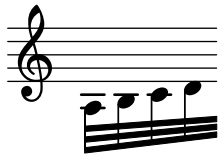


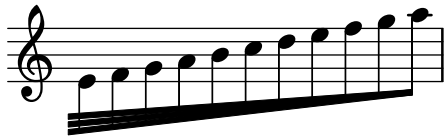


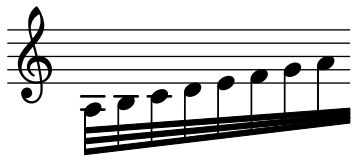




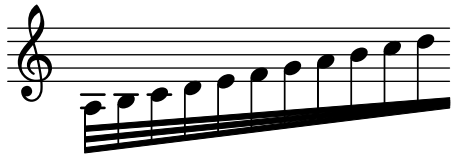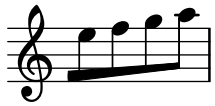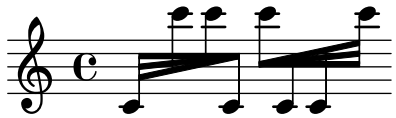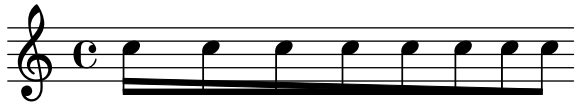














In feathered beams, stems in knees reach up to the feathered part correctly.

`beam-feather-knee-stem-length.ly`
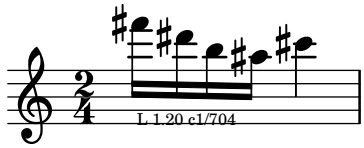


Specifying `grow-direction` on a beam, will cause feathered beaming. The `\featherDurations` function can be used to adjust note durations.

`beam-feather.ly`



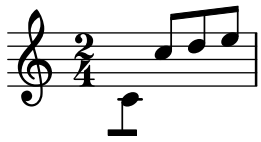
Even very flat but slanted patterns should give slanted beams.

`beam-flat-retain-direction.ly`
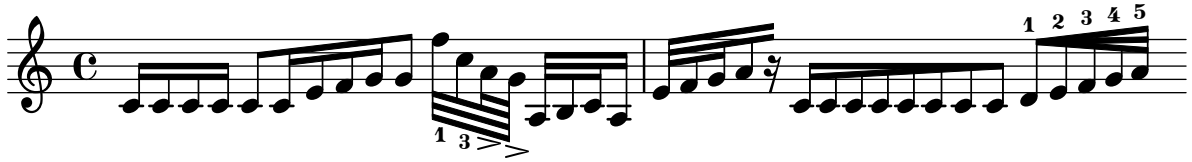


The direction of manual beams can be forced using `_` and `^`.

`beam-forced-direction.ly`
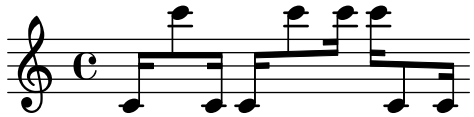


In French style beaming, the stems do not go between beams.

`beam-french.ly`



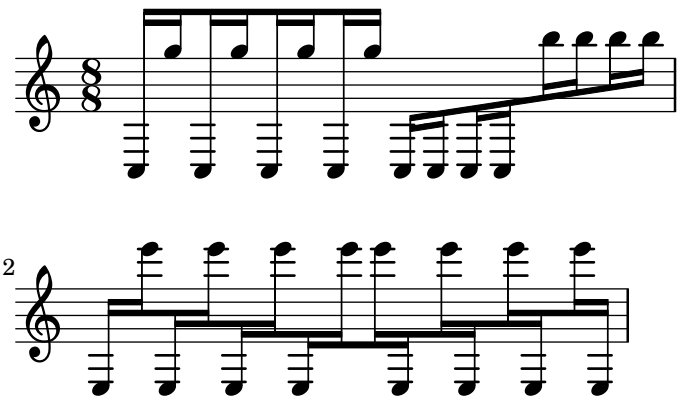
Funky kneed beams with beamlets also work. The beamlets should be pointing to the note head.

`beam-funky-beamlet.ly`



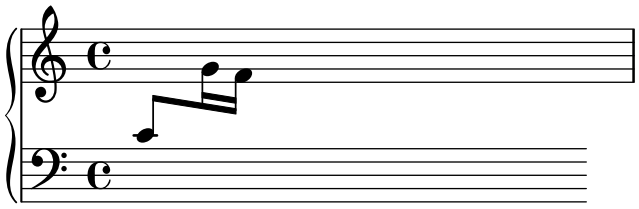
In complex configurations of knee beaming, according to Paul Roberts, the first stem of a beam determines the direction of the beam, and as such the way that following (kneed) stems attach to the beam. This is in disagreement with the current algorithm.

`beam-funky.ly`
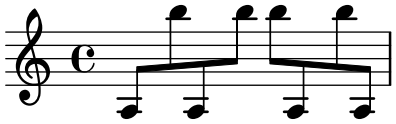


Beams can be placed across a `PianoStaff`.

`beam-isknee.ly`
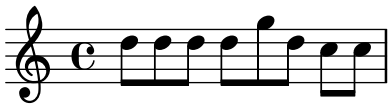


Point-symmetric beams should receive the same quanting. There is no up/down bias in the quanting code.

`beam-knee-symmetry.ly`
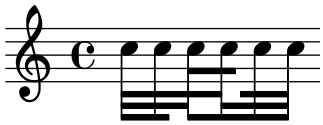


Beams should look the same.

`beam-length.ly`
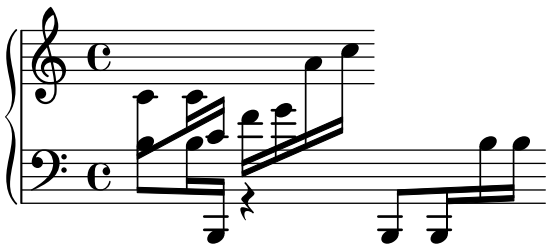


Beaming can be overidden for individual stems.

`beam-manual-beaming.ly`
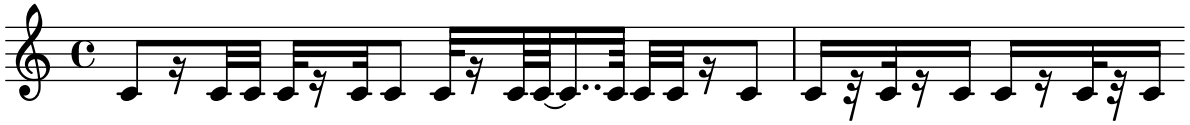


Kneed beams (often happens with cross-staff beams) should look good when there are multiple beams: all the beams should go on continuously at the staff change. Stems in both staves reach up to the last beam.

`beam-multiple-cross-staff.ly`
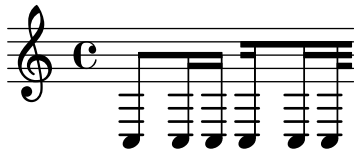


When a beam goes over a rest, beamlets should be as necessary to show the beat structure.

`beam-multiplicity-over-rests.ly`
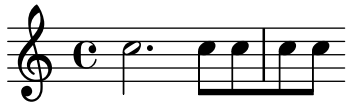


Beams may overshoot stems. This is also controlled with `break-overshoot`.

`beam-outside-beamlets.ly`
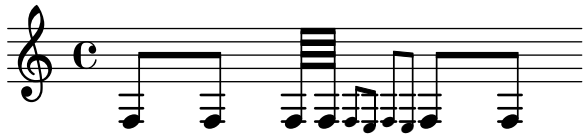


Explicit beams may cross barlines.

`beam-over-barline.ly`
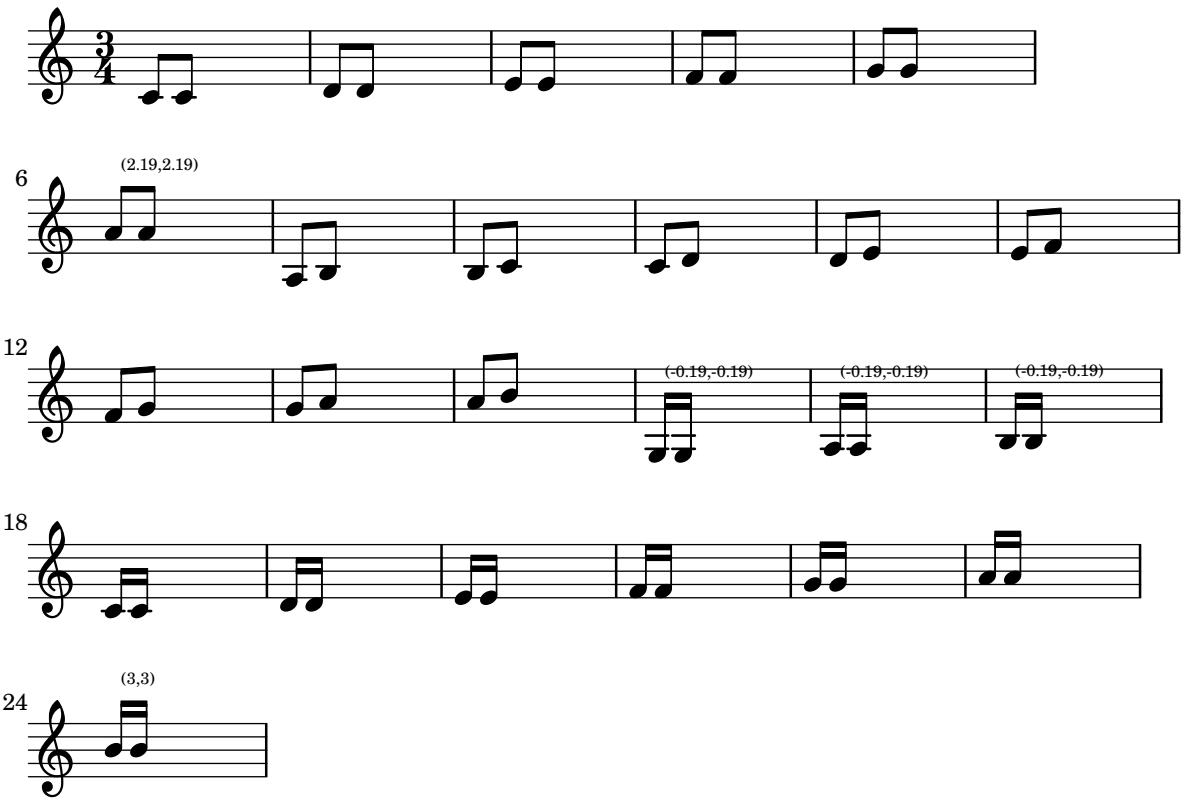


Beams on ledgered notes should always reach the middle staff line. The second beam, counting from the note head side, should never be lower than the second staff line. This does not hold for grace note beams. Override with `no-stem-extend`.

`beam-position.ly`
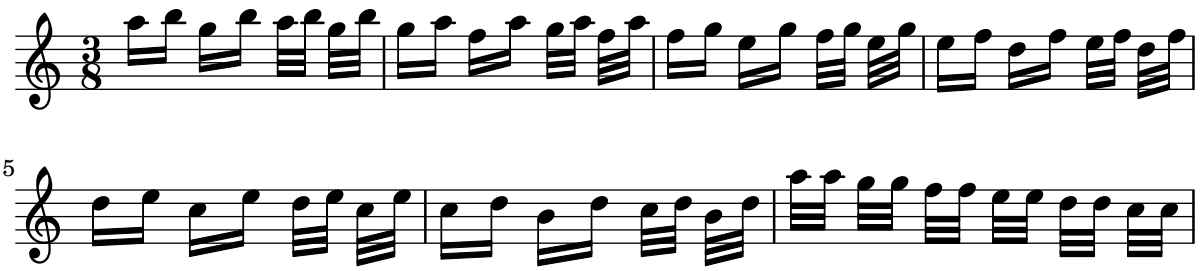


This file tests a few standard beam quants, taken from Ted Ross' book. If LilyPond finds another quant, the correct quant is printed over the beam.

`beam-quant-standard.ly`



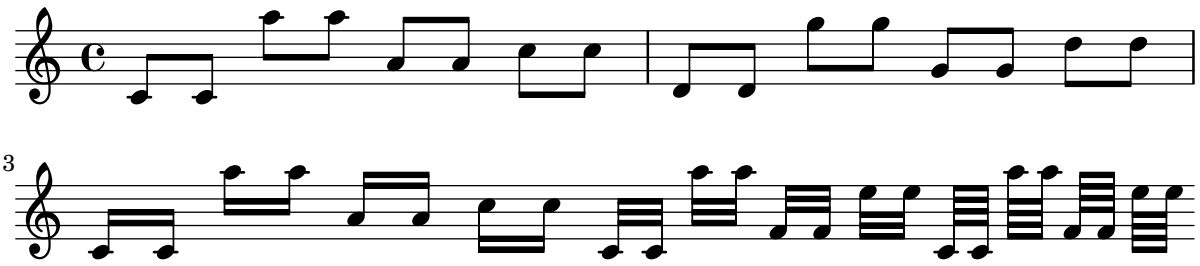
Stem lengths take precedence over beam quants: ‘forbidden’ quants are only avoided for 32nd beams when they are outside of the staff. However, that leads to very long stems, which is even worse.

`beam-quanting-32nd.ly`
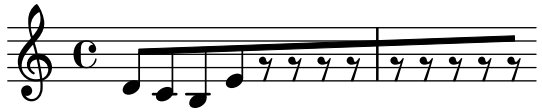


In this test for beam quant positions for horizontal beams, staff lines should be covered in all cases. For 32nd beams, the free stem lengths are between 2 and 1.5.

`beam-quanting-horizontal.ly`
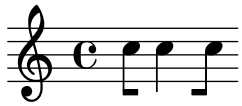


Beam quanting accounts for beam overhang. A beam ending above rests should always fall on a viable quant (straddle, sit, inter, or hang).

`beam-quanting-overhang.ly`



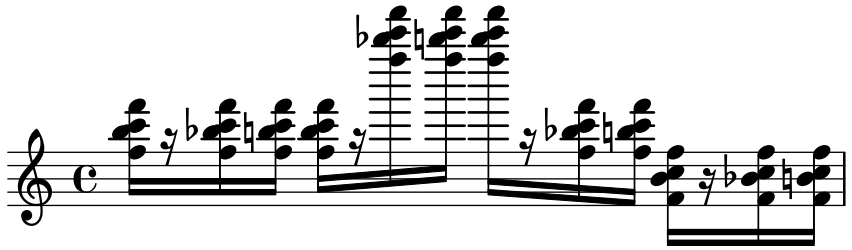
Quarter notes may be beamed: the beam is halted momentarily.

`beam-quarter.ly`
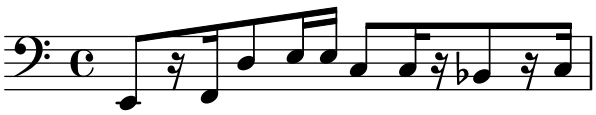


Beamed rests are given a pure height approximation that gets their spacing correct in the majority of circumstances.

`beam-rest-extreme.ly`
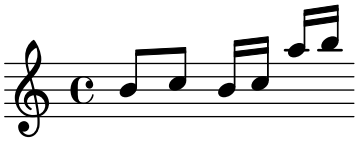


The number of beams does not change on a rest.

`beam-rest.ly`



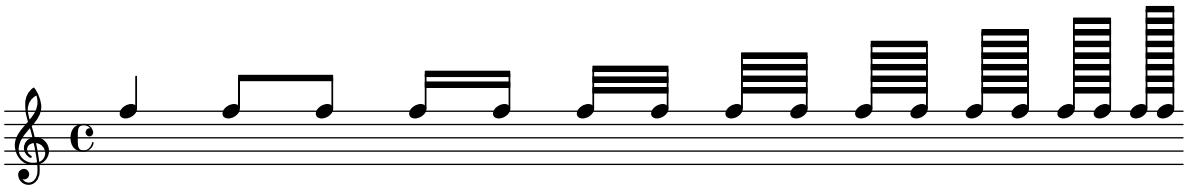
Engraving second intervals is tricky. We used to have problems with seconds being too steep, or getting too long stems. In a file like this, showing seconds, you'll spot something fishy very quickly.

`beam-second.ly`
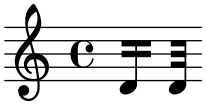


Beams in unnatural direction, have shortened stems, but do not look too short.

`beam-shortened-lengths.ly`
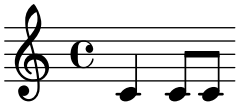


Single stem beams are also allowed. For such beams, clip-edges is switched off automatically.

`beam-single-stem.ly`
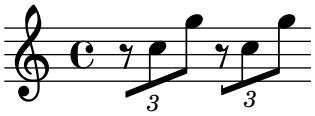


Beams over skips do not cause a segfault.

`beam-skip.ly`



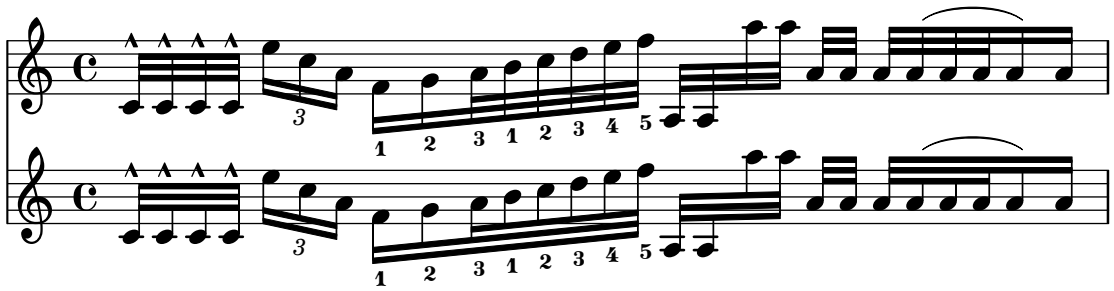
For slope calculations, stemlets are treated as invisible stems.

`beam-slope-stemlet.ly`
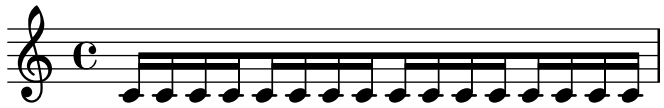


Beam positioning and placement of articulations, fingerings, tuplet numbers, and slurs must be identical in standard and French beaming style.

`beam-standard-french-compare.ly`



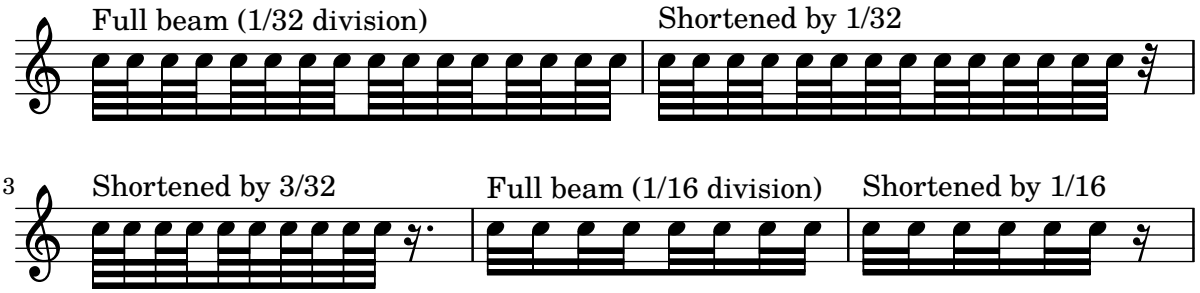
Beam count at subdivisions should match the location of the current subdivision. However, if the groups are equal or longer than quarter notes, one beam should always be left.

`beam-subdivide-quarter-notes.ly`
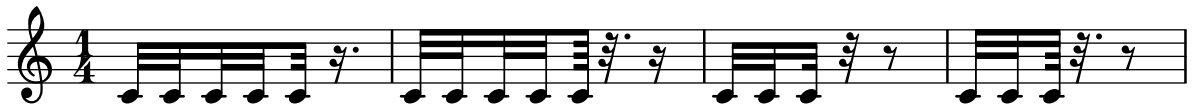


Beam count at subdivisions should match the count corresponding to the location of the current subdivision. However, if the remainder of the beam is shorter than that the beam count should be adopted accordingly.

`beam-subdivide-shortened-beam.ly`
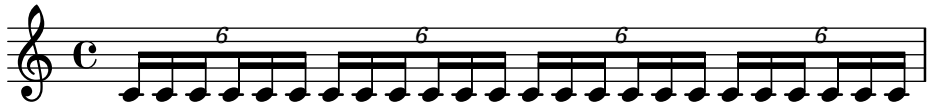


If in a subdivided beam one single stem follows a subdivision the beam count should reflect the beam count of the subdivision as usual. That is, the beam count should not be increased according to the remaining length of the beam. The appended single stem has beamlets to the left.

`beam-subdivide-trailing-stem.ly`



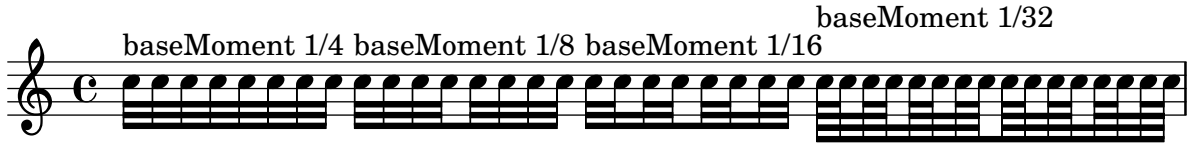
Tuplets that span more than one beat should be subdivided if subdivideBeams is #t. In this example, the beams should be subdivided every 1/8.

`beam-subdivide-tuplets.ly`
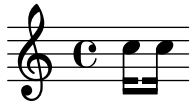


Beam count at subdivisions should match the location of the current subdivision.

`beam-subdivision.ly`



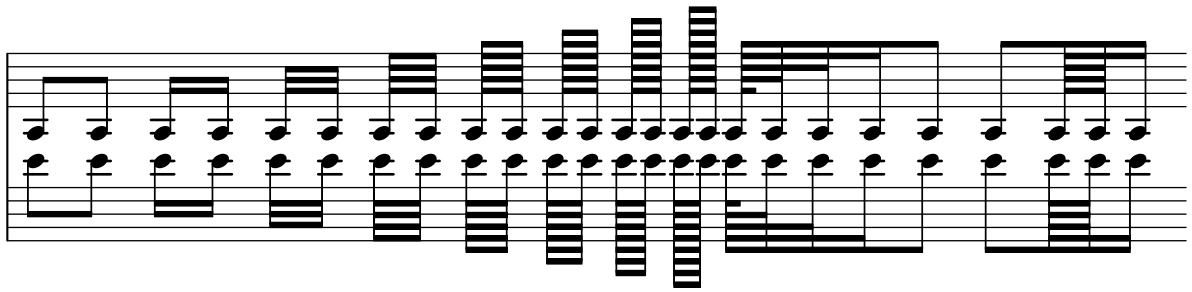
By setting `max-beam-connect`, it is possible to create pairs of unconnected beamlets.

`beam-unconnected-beamlets.ly`
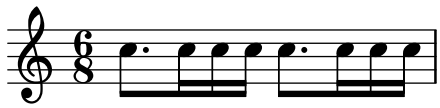


Inside-staff beams should align with staff lines (sit, straddle, hang) as smoothly as possible (standard-sized beams). The outside-staff beams do not interfere with staff lines, so the inside-staff beams are more important when it comes to beam quanting/scoring/positioning.

`beaming-more-than-4-beams-normal-size.ly`
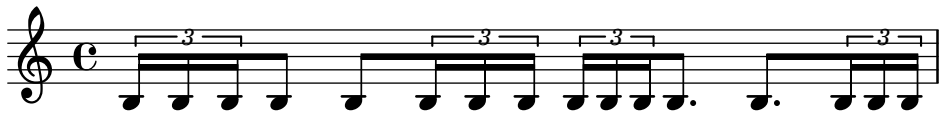


Automatic beaming works also in ternary time sigs. As desired, the measure is split in half, with beats 1-3 and 4-6 beamed together as a whole.

`beaming-ternary-metrum.ly`



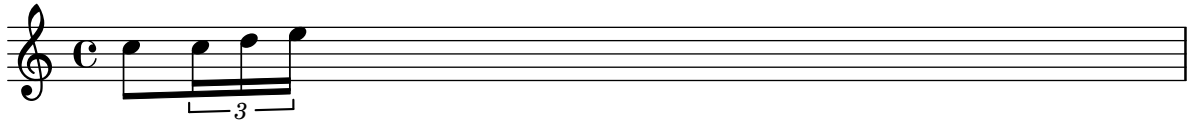
Beams in a completed tuplet should be continuous.
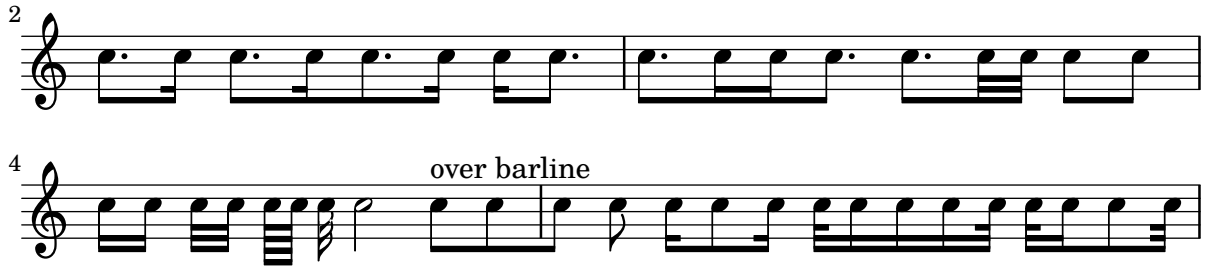
`beaming-tuplet-regular.ly`



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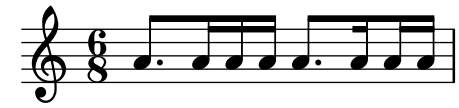
`beaming.ly`





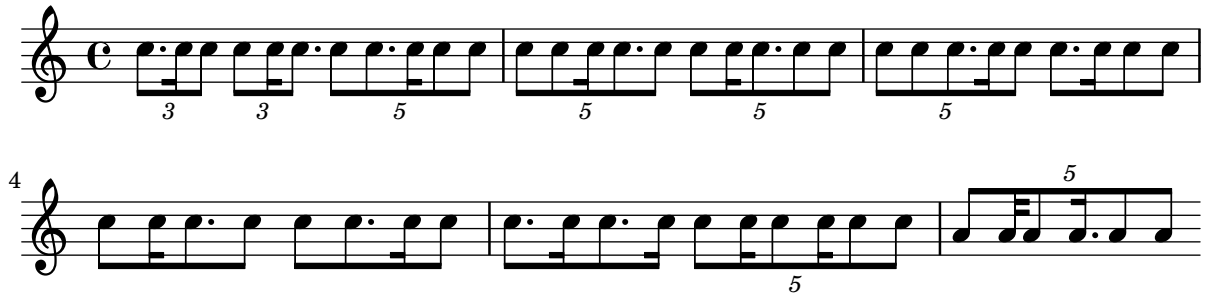
Beamlets can be set to point in the direction of the beat to which they belong. The first beam avoids sticking out flags (the default); the second beam strictly follows the beat.

`beamlet-point-toward-beat.ly`
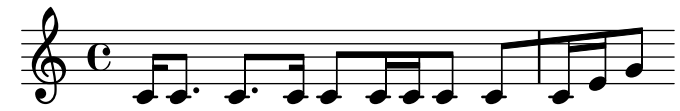


Beamlets should point away from complete beat units and toward off-beat or broken beat units. This should work in tuplets as well as in ordinary time.

`beamlet-test.ly`
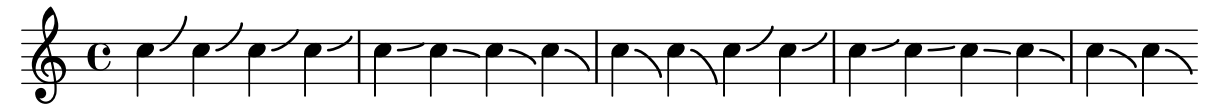


Beaming can be also given explicitly.

`beams.ly`



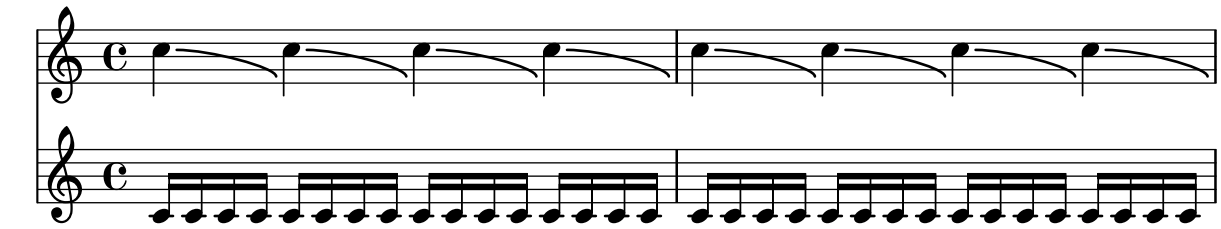
Falls and doits can be created with bendAfter. They run to the next note, or to the next barline. Microtone bends (i.e. \bendAfter #3.5) are also supported.

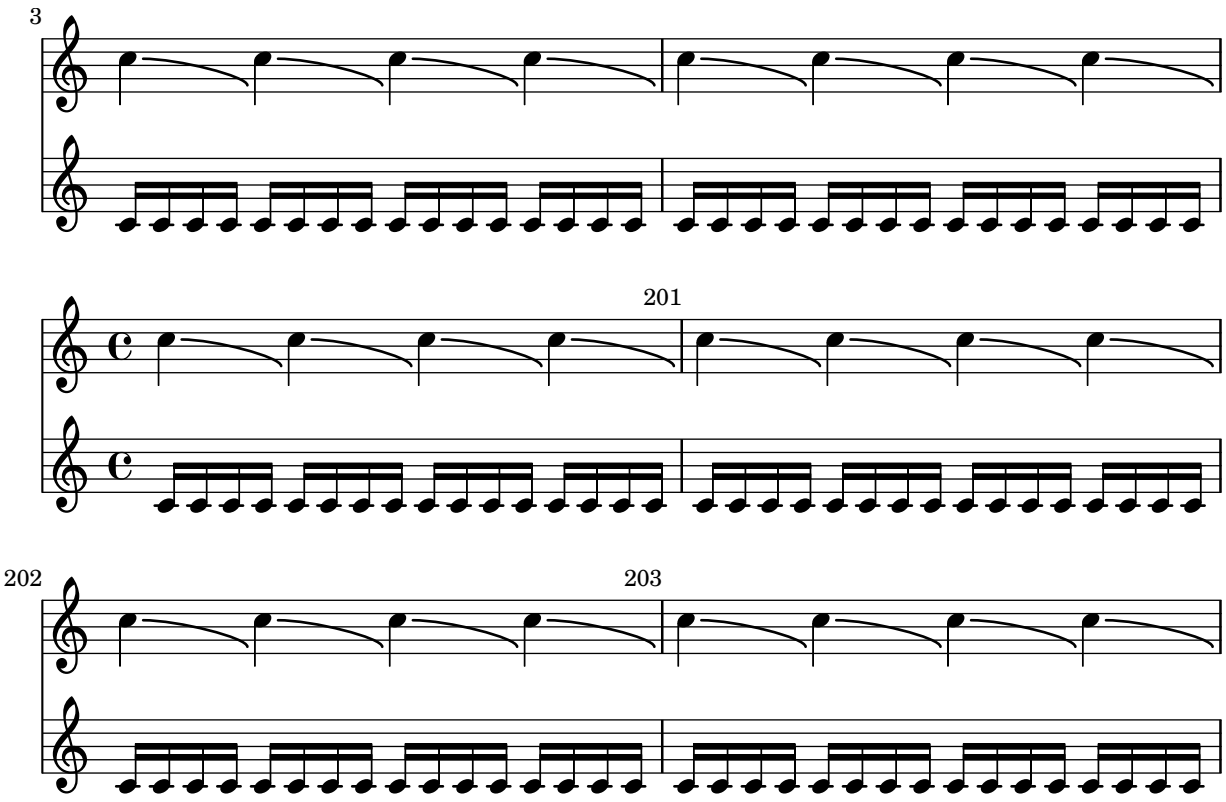
`bend-after.ly`



Bends should not be effected by the full width of a `NonMusicalPaperColumn`. The bends should have identical X spans in the two examples.
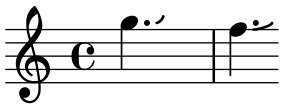
`bend-bound.ly`





Bends avoid dots, but only if necessary.

`bend-dot.ly`



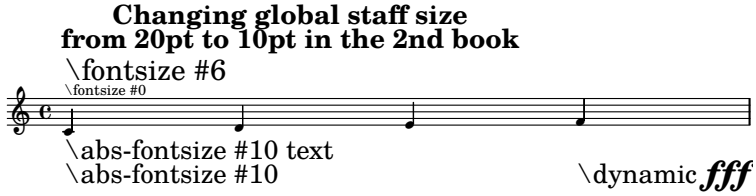
This input file contains a UTF-8 BOM not at the very beginning, but on the first line after the first byte. LilyPond should gracefully ignore this BOM as specified in RFC 3629, but print a warning.

`bom-mark.ly`



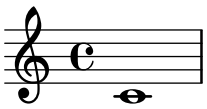
Changing `global-staff-size` between consecutive `\book`s must not impair font spacing. While the Pango fonts stay the same and may be re-used, the internal LilyPond scaling factor will not be correct any more. Not only `\abs-fontsize`, but even `\fontsize` (in extreme cases) will be affected. The following output shows a 10pt book after a standard 20pt book:

`book-change-global-staffsize-abs-fonts.ly`



A `\book` or `\bookpart` identifier can contain top-level markup and page-markers.

book-identifier-markup.ly

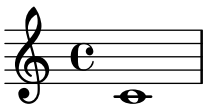


A book(part) can contain only a label without causing a segfault.

book-label-no-segfault.ly

foo

bookpart-variable.ly



A book can be split into several parts with different paper settings, using `\bookpart`.

Fonts are loaded into the top-level paper. Page labels are also collected into the top-level paper.

`bookparts.ly`

Book with several parts

First part
with default paper settings.

II SECOND PART

Book with several parts

Second part, with different margins
and page header.



3

Book with several parts

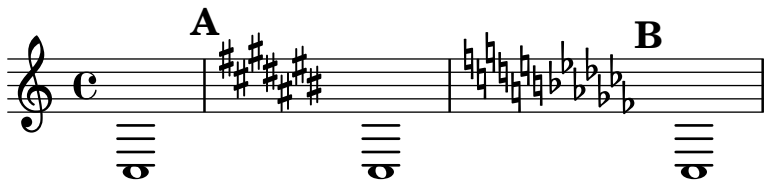
Third part

Table of Contents

First part 1
Second part 2
Third part 3

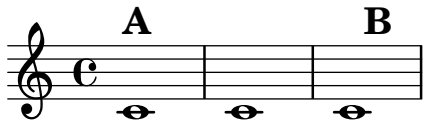
The default callback for break-align-anchor in clefs and time/key signatures reads the `break-align-anchor-aligment` property to align the anchor to the extent of the break-aligned grob.

`break-alignment-anchor-alignment.ly`



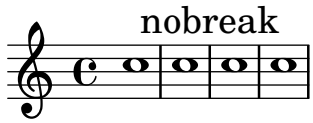
The break-align-anchor property of a break-aligned grob gives the horizontal offset at which other grobs should attach.

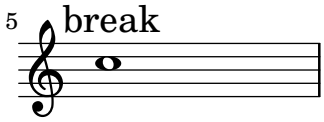
`break-alignment-anchors.ly`

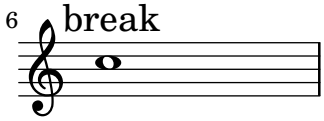


Breaks can be encouraged and discouraged using `\break` and `\noBreak`.

`break.ly`

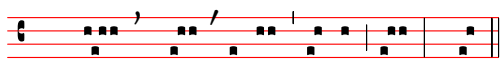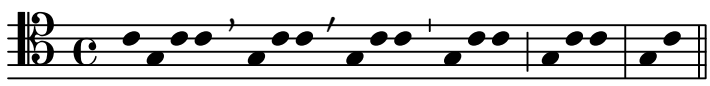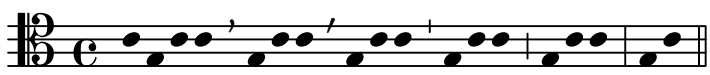






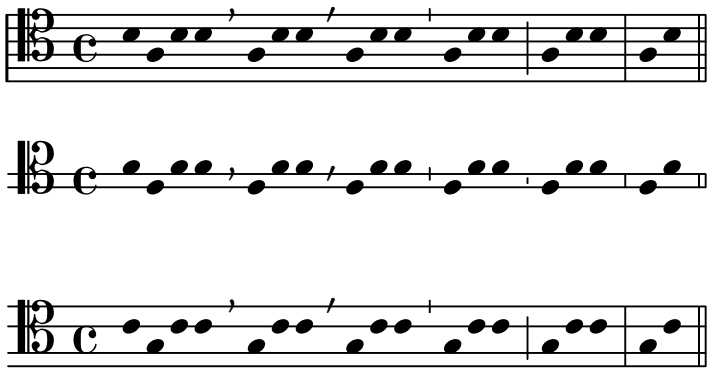
Gregorian chant notation sometimes also uses commas and ticks, but in smaller font size (we call it ‘virgula’ and ‘caesura’). However, the most common breathing signs are divisio minima/ maior/maxima and finalis, the latter three looking similar to bar glyphs.

`breathing-sign-ancient.ly`
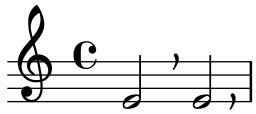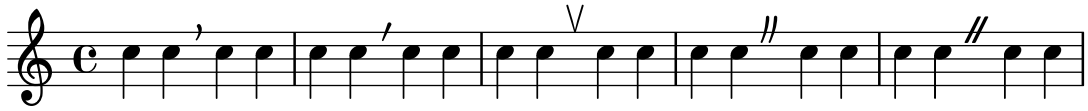








Breathing signs are positioned correctly on custom staves which use `line-positions`.

`breathing-sign-custom-staff.ly`
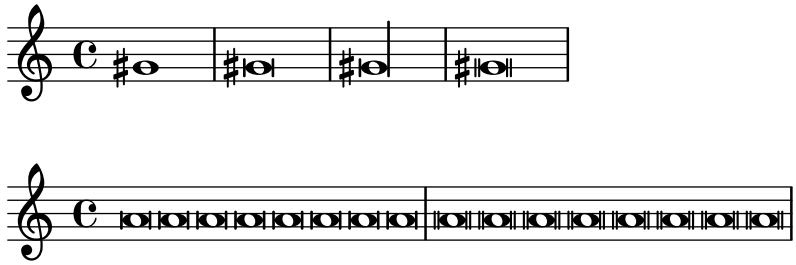


Breathing signs are available in different tastes: commas (default), ticks, vees and ‘railroad tracks’ (caesura).

`breathing-sign.ly`



LilyPond knows that breves and longas are wider than whole notes (because of vertical lines on their sides). Breves and longas don’t collide with accidentals, barlines, neighbor notes etc. The distance between accidental and note is the same for whole notes, breves and longas.

`breve-extent.ly`



A grace note after `\cadenzaOff` does not keep autobeaming from resuming properly.
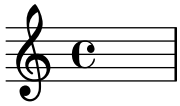
`cadenza-grace-autobeam.ly`



Long titles should be properly centered.

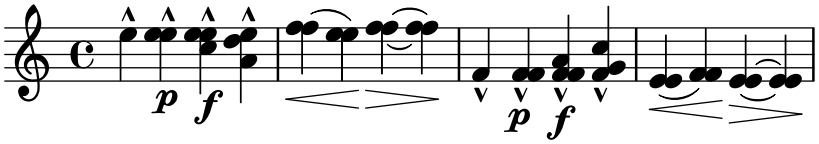
`center-title.ly`

How Razorback Jumping Frogs Level Six Piqued Gymnast



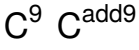
Chords containing unisons or seconds: Center articulation marks, dynamics, slurs, etc. on the notehead that is on the "correct" side of the stem.

`chord-X-align-on-main-noteheads.ly`



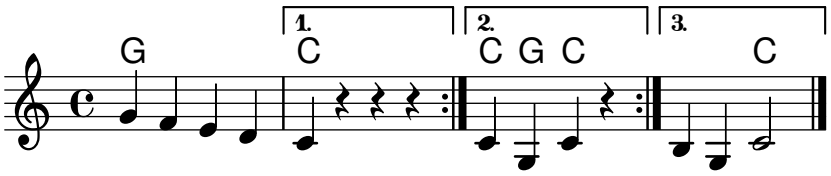
The prefix of additional chord pitches can be tuned with `additionalPitchPrefix`.

`chord-additional-pitch-prefix.ly`



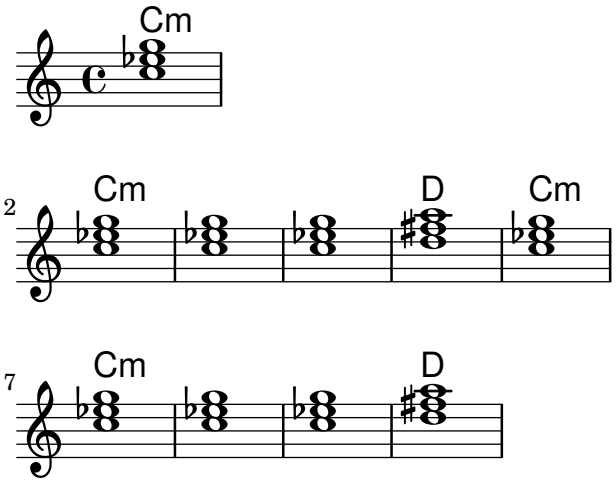
Chord change detection in repeat alternatives happens in relation to the chord active at the beginning of the first alternative.

`chord-changes-alternative.ly`



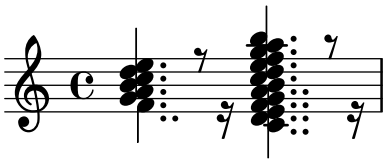
Property chordChanges: display chord names only when there's a change in the chords scheme, but always display the chord name after a line break.

`chord-changes.ly`



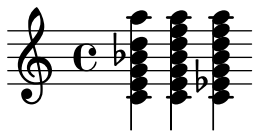
The column of dots on a chord is limited to the height of the chord plus `chord-dots-limit` staff-positions.

`chord-dots.ly`



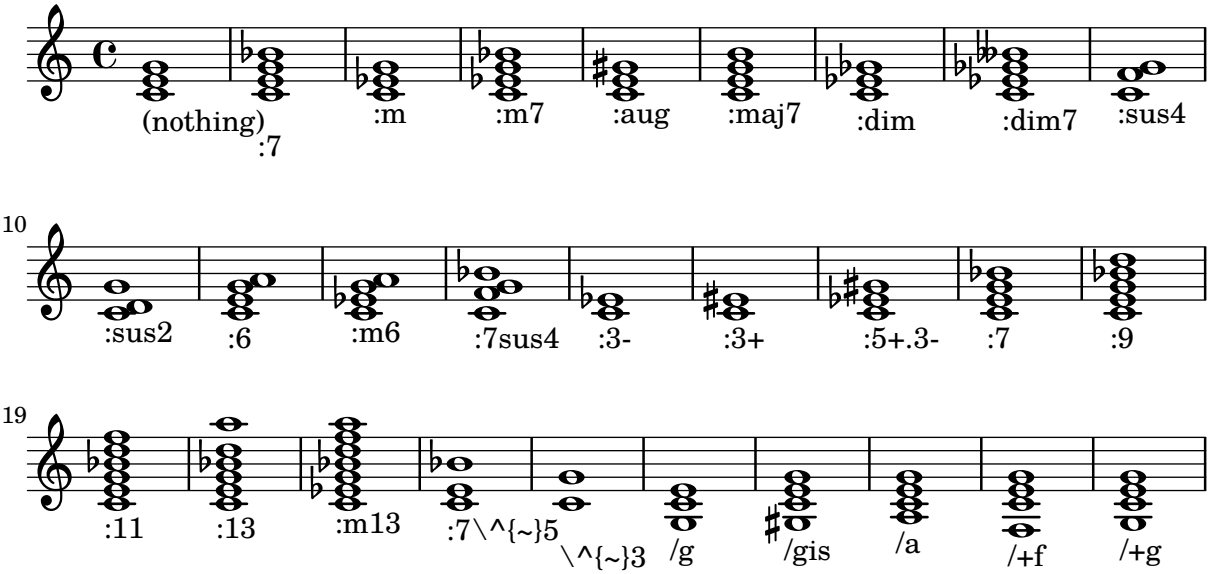
The 11 is only added to major-13 if it is mentioned explicitly.

`chord-name-entry-11.ly`



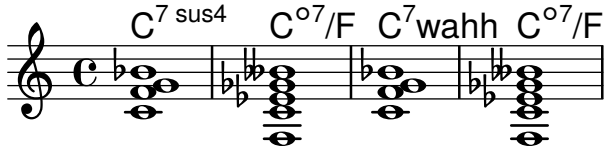
Chords can be produced with the chordname entry code (`\chordmode` mode), using a pitch and a suffix. Here, the suffixes are printed below pitches.

`chord-name-entry.ly`



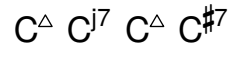
The property `chordNameExceptions` can used to store a list of special notations for specific chords.

`chord-name-exceptions.ly`



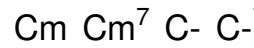
The layout of the major 7 can be tuned with `majorSevenSymbol`. It does not break if `majorSevenSymbol` is unset. One should see: triangle - j7 - triangle - #7.

`chord-name-major7.ly`



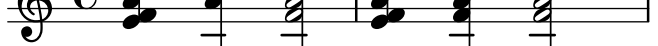
The layout of the minor chord can be tuned with `minorChordModifier`.

`chord-name-minor.ly`



Users can override the `text` property of `ChordName`.

`chord-name-override-text.ly`

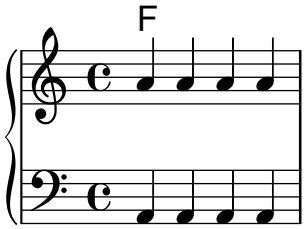


In ignatzek inversions, a note is dropped down to act as the bass note of the chord. Bass note may be also added explicitly. Above the staff: computed chord names. Below staff: entered chord name.

`chord-names-bass.ly`

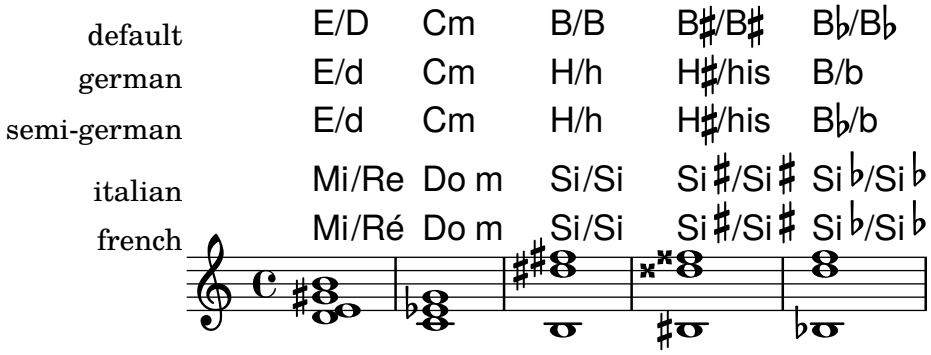
GrandStaff contexts accept chord names. The chord name in this example should be printed above the top staff.

`chord-names-in-grand-staff.ly`



The english naming of chords (default) can be changed to german (`\germanChords` replaces B and Bes to H and B), semi-german (`\semiGermanChords` replaces B and Bes to H and Bb), italian (`\italianChords` uses Do Re Mi Fa Sol La Si), or french (`\frenchChords` replaces Re to Ré).

`chord-names-languages.ly`



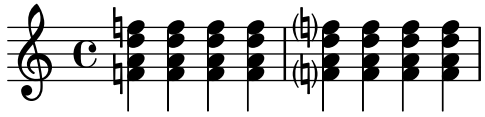
Minor chords may be printed as lowercase letters, in which case the ‘m’ suffix is omitted in the output.

`chord-names-lower-case-minor.ly`



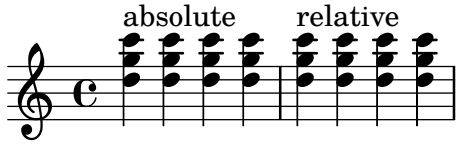
Chord repeats should omit forced and reminder accidentals.

chord-repetition-accidentals.ly



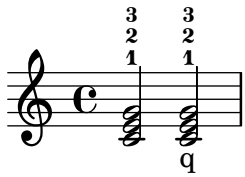
Chord repetition handles \relative mode: the repeated chords have the same octaves as the original one.

chord-repetition-relative.ly



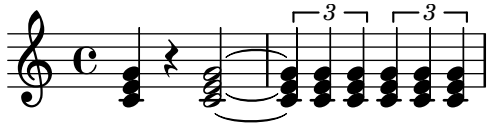
Post events such as fingerings and scripts added to a chord repetition follow the same basic stacking order as chords.

chord-repetition-script-stack.ly



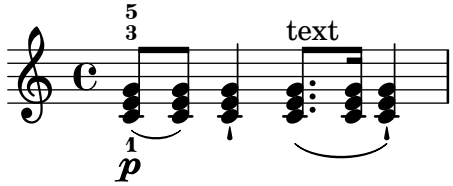
Chord repetitions are expanded late in the processing order and get their note events only then. Check that `\times` still works correctly on them.

chord-repetition-times.ly
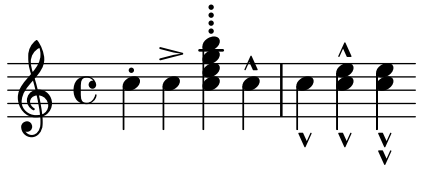


A repetition symbol can be used to repeat the previous chord and save typing. Only note events are copied: articulations, text scripts, fingerings, etc are not repeated.

chord-repetition.ly



Scripts can also be attached to chord elements. They obey manual direction indicators.

chord-scripts.ly
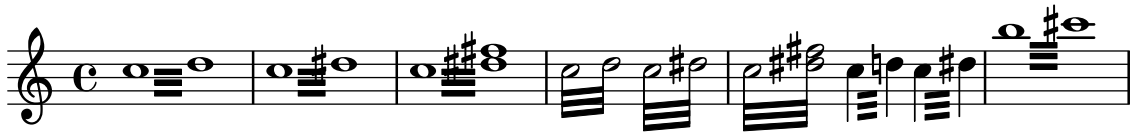


The layout of chord inversions can be tuned with `slashChordSeparator`.

`chord-slash-separator.ly`

D♭/C D♭ over C
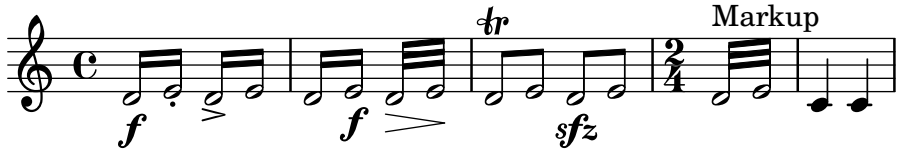
Chord tremolos adapt to the presence of accidentals.

`chord-tremolo-accidental.ly`



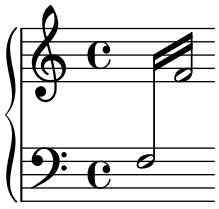
Articulations on chord tremolos should not confuse the time-scaling of the notes. In particular, only the number of real notes should be considered.

`chord-tremolo-articulations.ly`
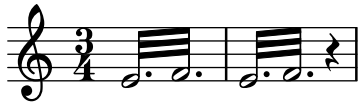


To calculate the total duration of chord tremolos, only real notes shall be counted, no other commands.

`chord-tremolo-other-commands.ly`
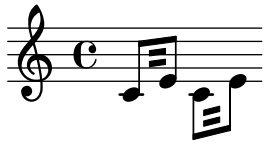


Don't allow scaled durations to confuse the tremolo beaming. The tremolos should each have 3 beams.

`chord-tremolo-scaled-durations.ly`
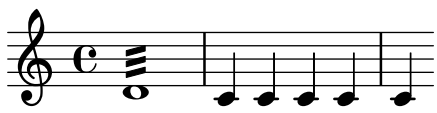


Tremolo repeats can be constructed for short tremolos (total duration smaller than 1/4) too. Only some of the beams are connected to the stems.

`chord-tremolo-short.ly`



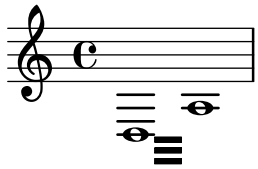
Chord tremolos on a single note.

chord-tremolo-single.ly
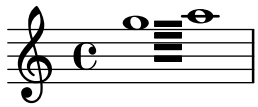


Stem directions influence positioning of whole note tremolo beams.

chord-tremolo-stem-direction.ly



chord tremolos don't collide with whole notes.

chord-tremolo-whole.ly
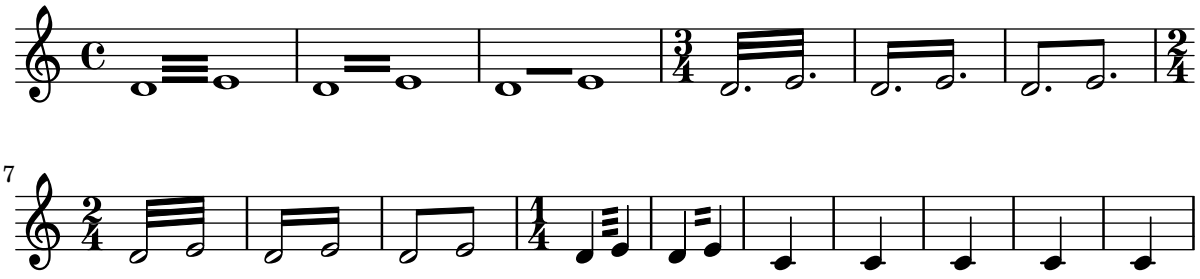


Chord tremolos look like beams, but are a kind of repeat symbol. To avoid confusion, chord tremolo beams do not reach the stems, but leave a gap. Chord tremolo beams on half notes are not ambiguous, as half notes cannot appear in a regular beam, and should reach the stems.

In this example, each tremolo lasts exactly one measure.

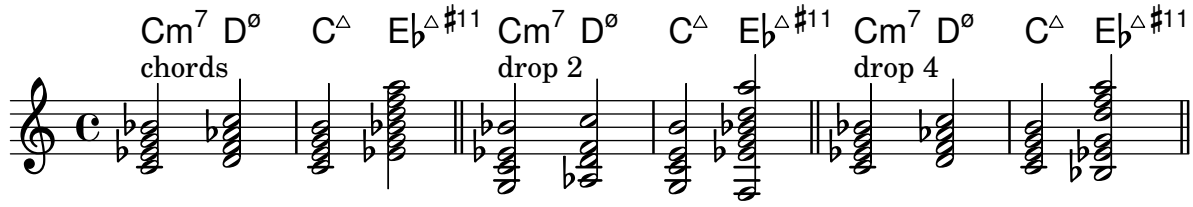
(To ensure that the spacing engine is not confused we add some regular notes as well.)

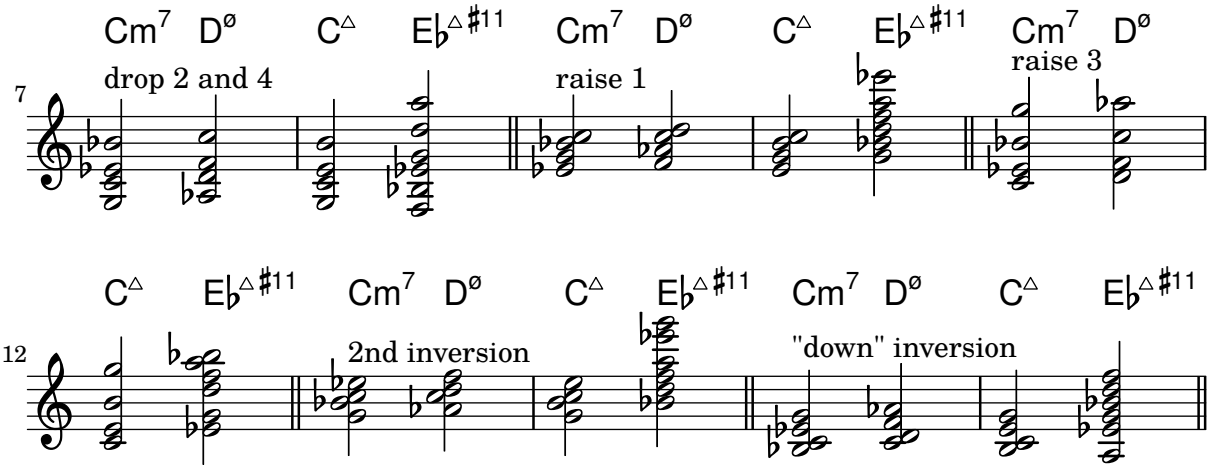
chord-tremolo.ly



Chord voicings may be transformed or inverted automatically through Scheme functions. These work even when chord notes are not entered in order (e.g. from the lowest to the uppermost note), and may also be used in chordmode. Even when using voicings, chord names remain unchanged.

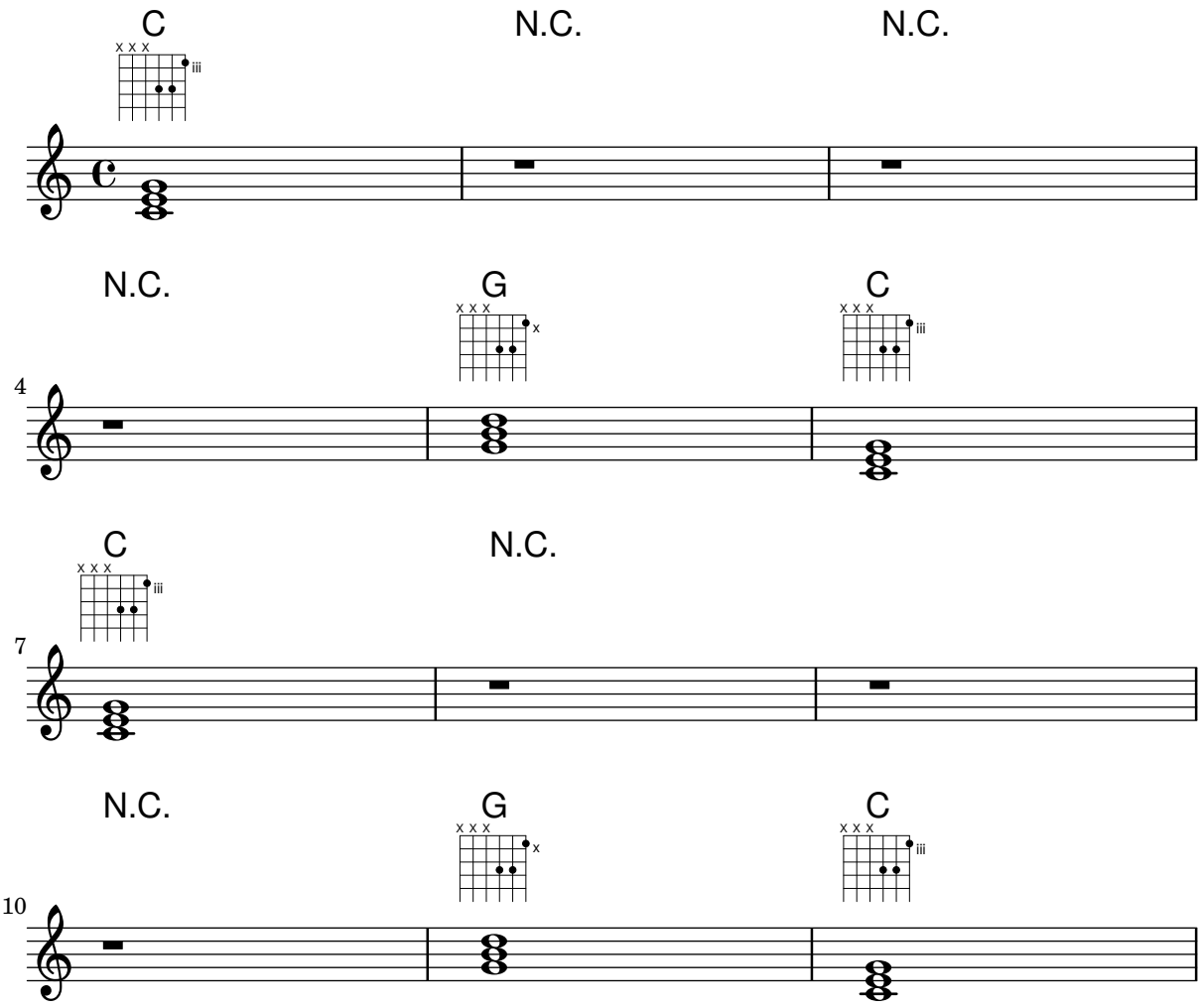
chord-voicings.ly





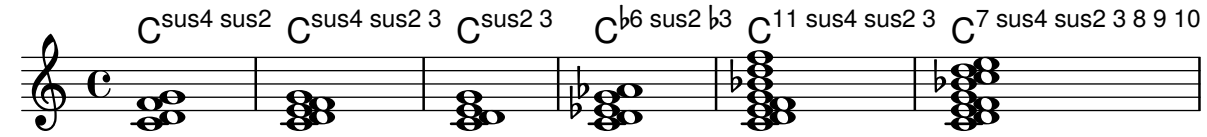
Rests in music passed to ChordNames context display noChordSymbol. noChordSymbol is treated like a ChordName with respect to chordChanges.

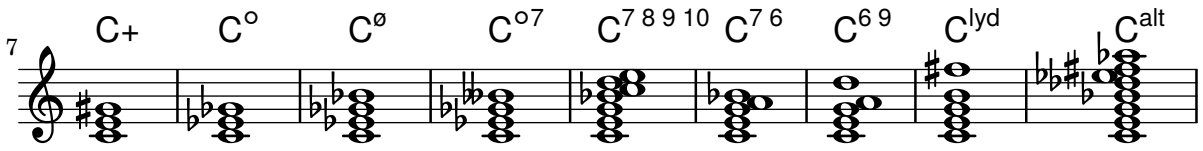
`chordnames-nochord.ly`



Jazz chords may have unusual combinations.
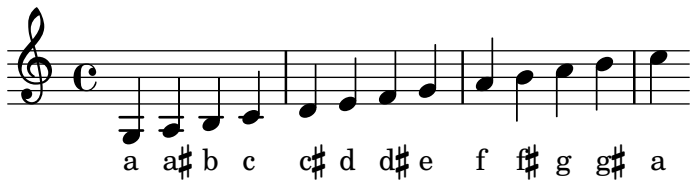
`chords-funky-ignatzek.ly`
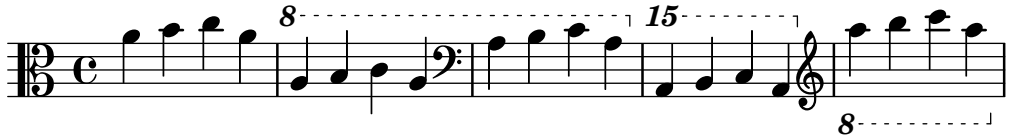




`staffLineLayoutFunction` is used to change the position of the notes. This sets `staffLineLayoutFunction` to `ly:pitch-semitones` to produce a chromatic scale with the distance between a consecutive space and line equal to one semitone.

`chromatic-scales.ly`



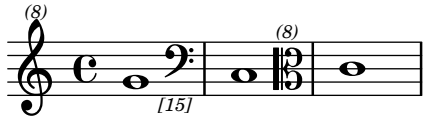
Ottava brackets and clefs both modify Staff.middleCPosition, but they don't confuse one another.

`clef-ottava.ly`



Clef transposition symbols may be parenthesized or bracketed by using parentheses or brackets in the command string.
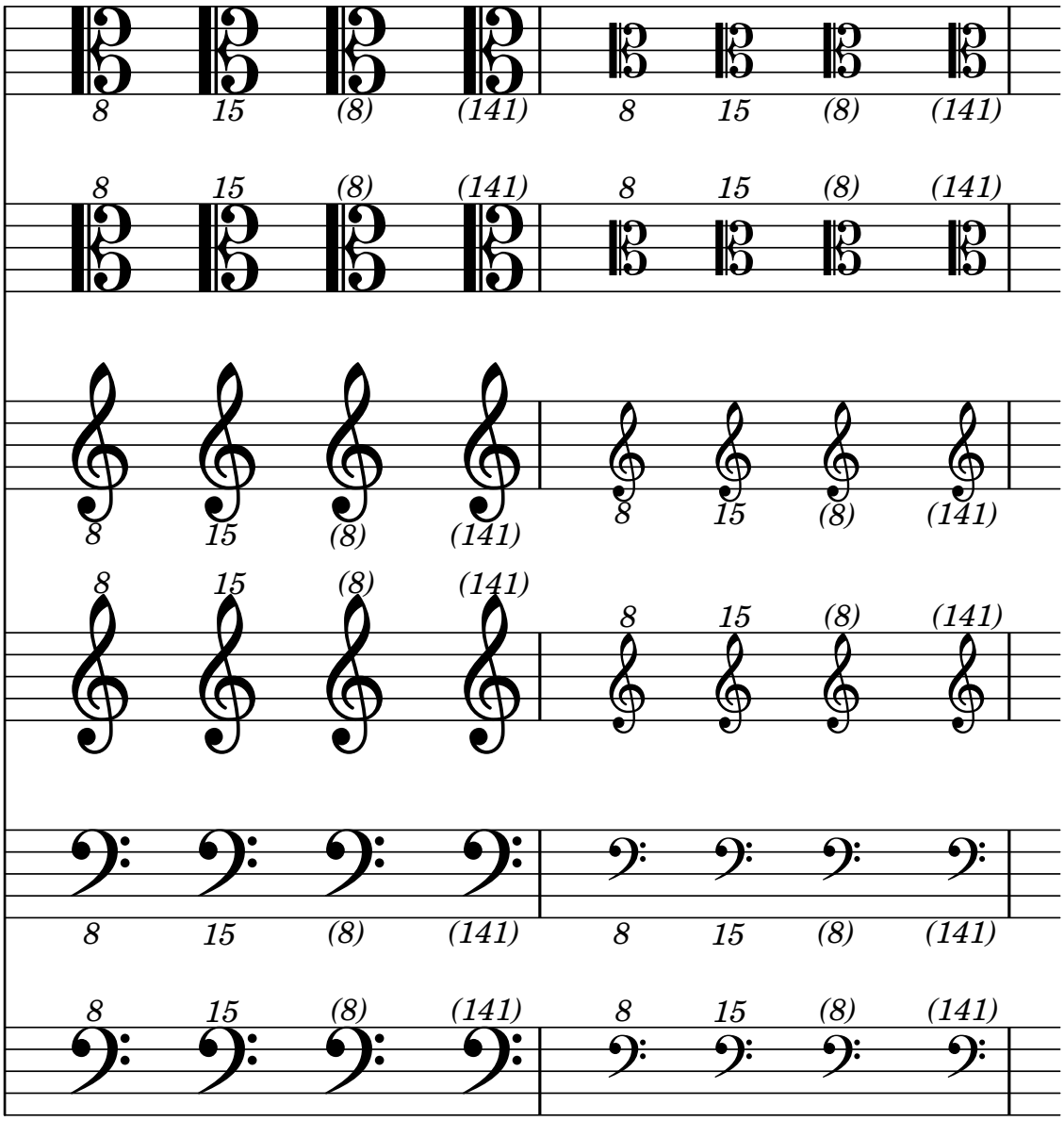
`clef-transposition-optional.ly`



Transposition symbols should be correctly positioned close to the parent clef. Horizontal alignment is fine-tuned for standard C, G and F clefs: for example, downwards transposition of a G clef should be centered exactly under the middle of clef hook. For clefs that don't have fine-tuned alignment the transposition number should be centered.

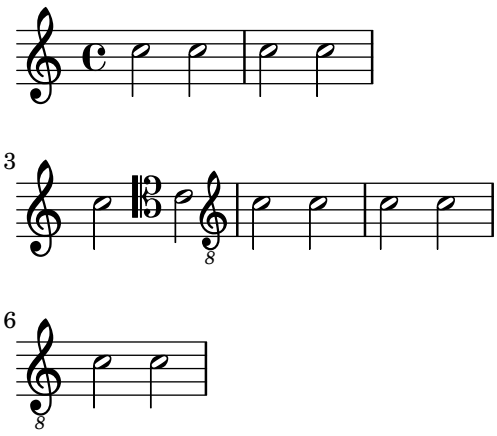
`clef-transposition-placement.ly`

Even the smallest positioning changes may indicate a prob



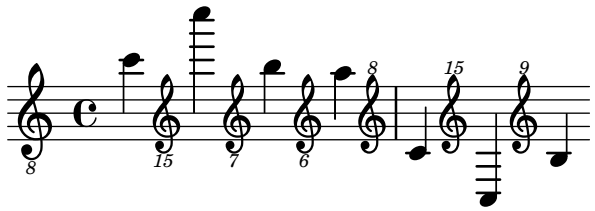
Clefs may be transposed. By default, break-visibility of ClefModifiers is derived from the associated clef, but it may be overridden explicitly. The initial treble_8 clef should not have an 8, while the treble_8 clef after the tenor clef should. These settings also need to apply to clefs on new lines.

`clef-transposition-visibility.ly`



Clefs may be transposed up or down by arbitrary amount, including 15 for two octaves.

`clef-transposition.ly`



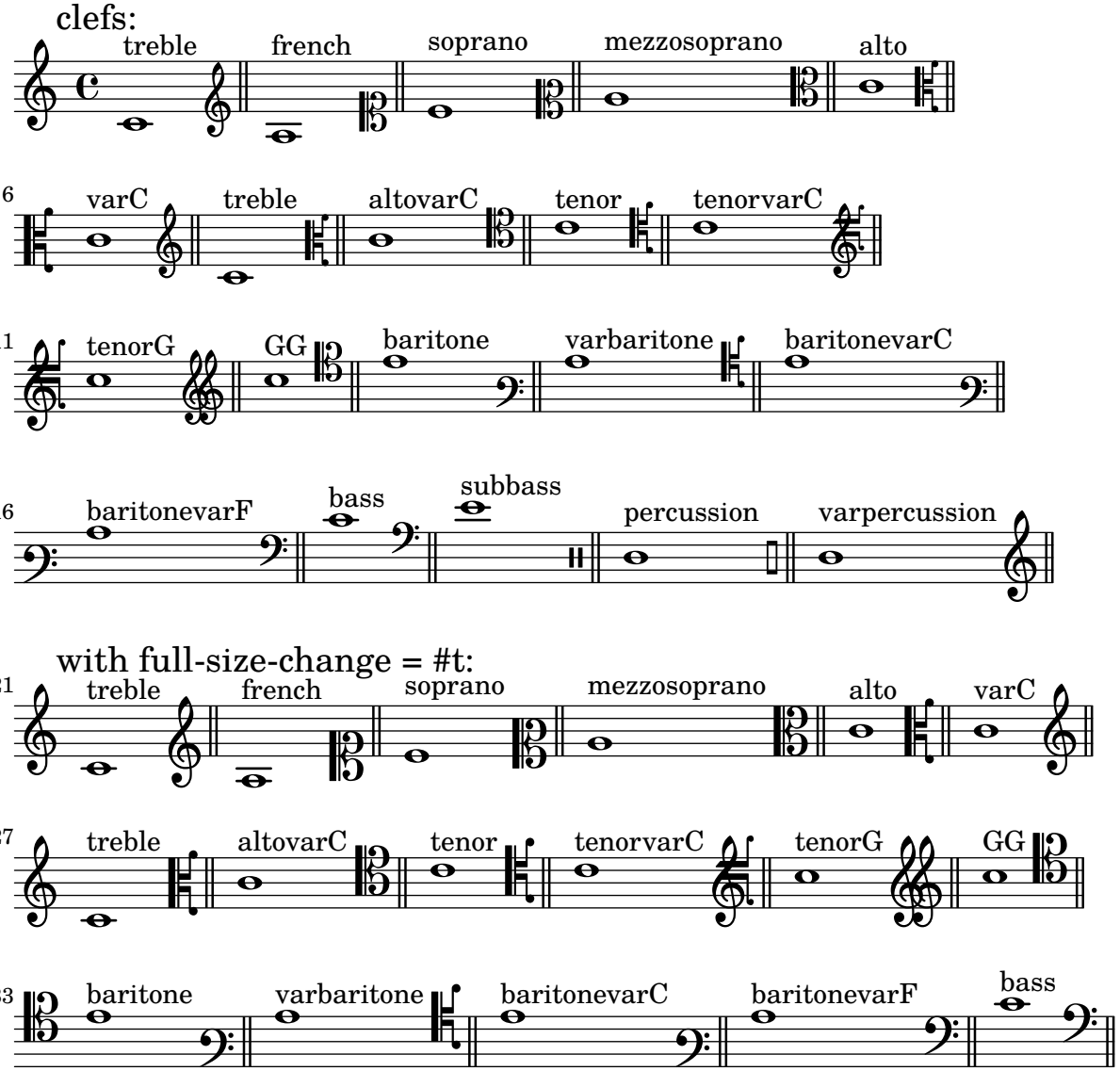
Unknown clef name warning displays available clefs

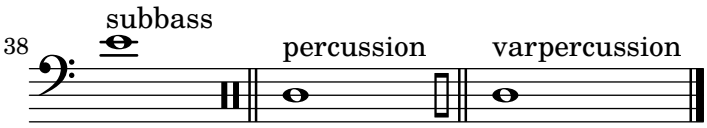
`clef-warn.ly`



Clefs with `full-size-change` should be typeset in full size.

`clefs.ly`





Clipping snippets from a finished score

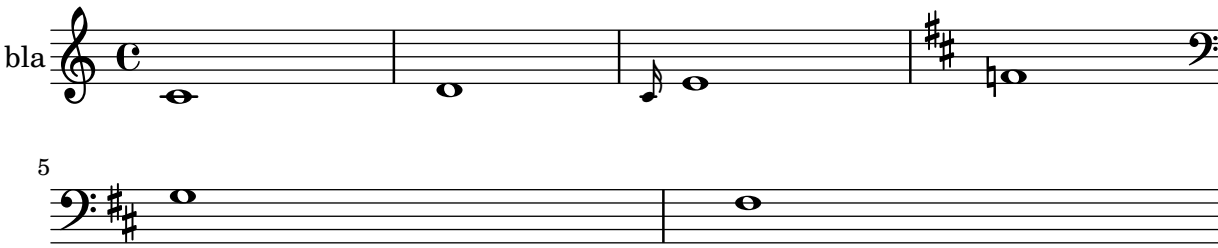
Notes:

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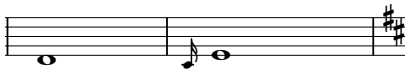
This file needs to be run separately with `-dclip-systems`; the collated-files.html of the regression test does not adequately show the results.

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

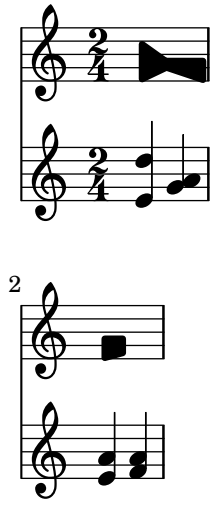
clip-systems.ly



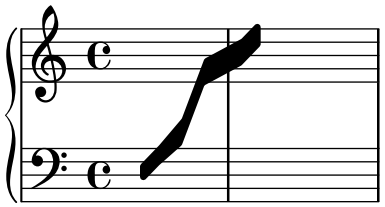
clips

from-2.0.1-to-4.0.1-clip.eps 
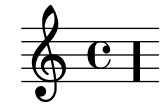
Clusters behave well across line breaks.
`cluster-break.ly`



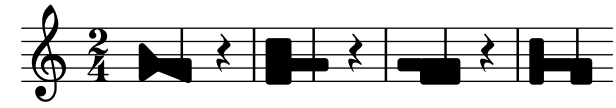
Clusters can be written across staves.
`cluster-cross-staff.ly`



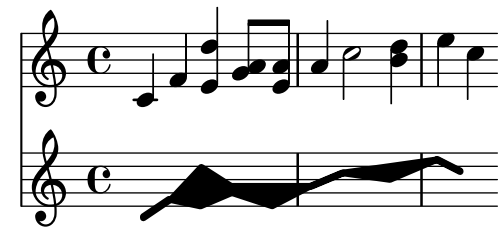
don’t crash on single chord clusters.
`cluster-single-note.ly`



Clusters behave well across line breaks.
`cluster-style.ly`
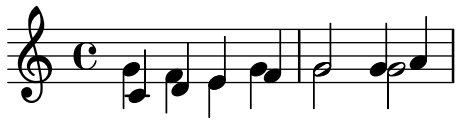


Clusters are a device to denote that a complete range of notes is to be played.
`cluster.ly`
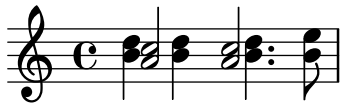


Single head notes may collide.

`collision-2.ly`
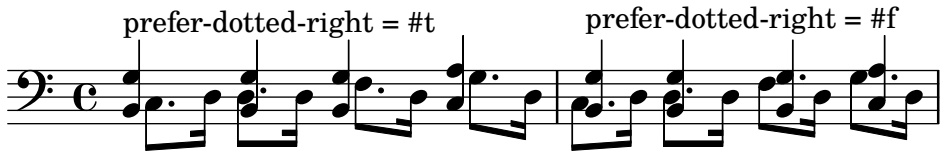


When notes are colliding, the resolution depends on the dots: notes with dots should go to the right, if there could be confusion to which notes the dots belong.

`collision-dots-invert.ly`
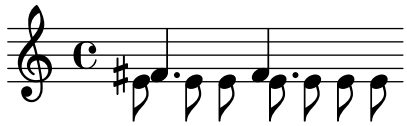


If dotted note heads must remain on the left side, collision resolution moves the dots to the right.

`collision-dots-move.ly`



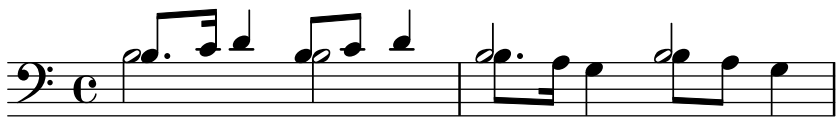
For collisions where the upper note is dotted and in a space, the upper is moved to right. This behavior can be tuned by prefer-dotted-right.

`collision-dots-up-space-dotted.ly`
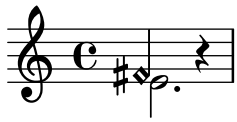


Collision resolution tries to put notes with dots on the right side.

`collision-dots.ly`
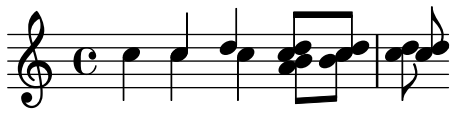


Collision resolution involving dotted harmonic heads succeeds when dots are hidden since `rhythmic-head-interface` will only retrieve `'dot-count` from live grobs.

`collision-harmonic-no-dots.ly`
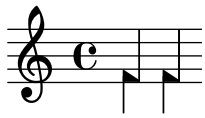


Note heads in collisions should be merged if they have the same positions in the extreme note heads.

`collision-head-chords.ly`
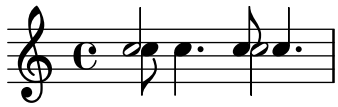


The FA note (a triangle) is merged to avoid creating a block-shaped note.

`collision-head-solfa-fa.ly`



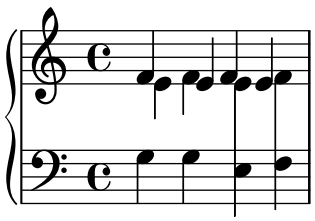
Open and black note heads are not merged by default.

`collision-heads.ly`
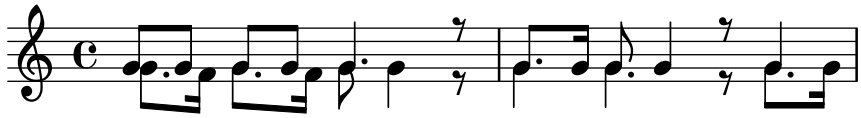


Colliding note-columns may be shifted manually with `force-hshift`. Arrangements of notes after collision-resolution have their main columns (not suspended notes) left-aligned, excluding columns with forced shifts.

`collision-manual.ly`
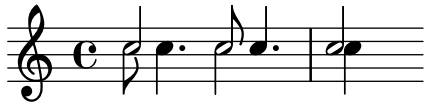


If `NoteCollision` has `merge-differently-dotted = ##t` note heads that have differing dot counts may be merged anyway. Dots should not disappear when merging similar note heads.

`collision-merge-differently-dotted.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-differently-headed.ly`



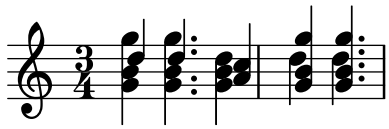
When merging heads, the dots are merged too.

`collision-merge-dots.ly`
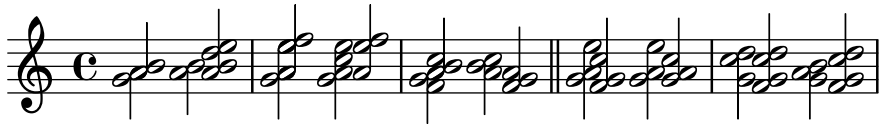


Oppositely stemmed chords, meshing into each other, are resolved.

`collision-mesh.ly`
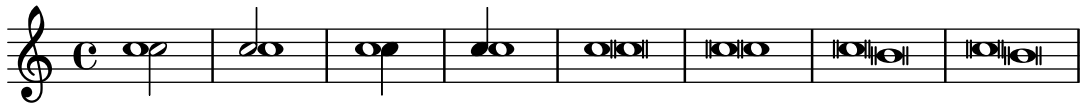


Seconds do not confuse the collision algorithm. The first pair of chords in each measure should merge, mesh, or come relatively close, but the second in each measure needs more space to make clear which notes belong to which voice.

`collision-seconds.ly`



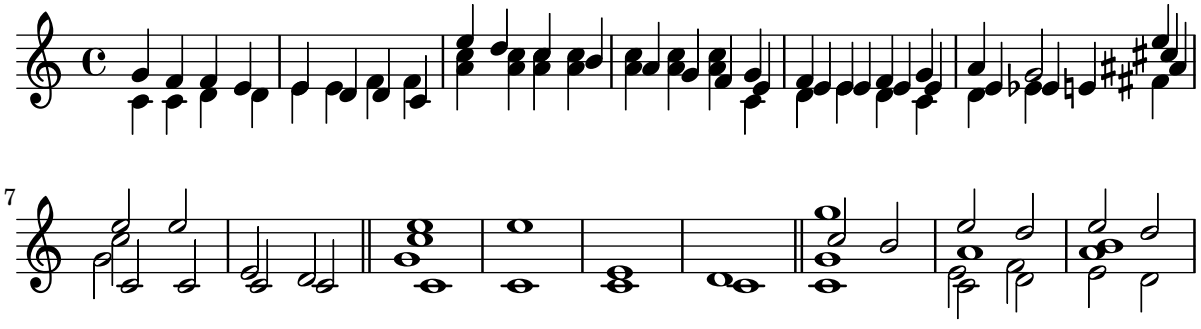
Mixed collisions with whole and longer notes require asymmetric shifts.

`collision-whole.ly`



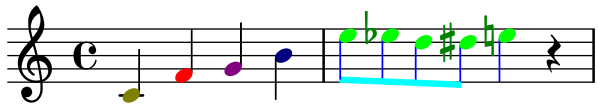
In addition to normal collision rules, there is support for polyphony, where the collisions are avoided by shifting middle voices horizontally.

`collisions.ly`
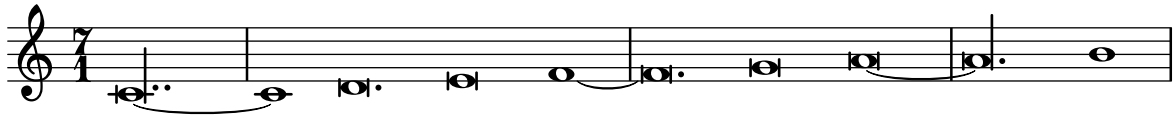


Each grob can have a color assigned to it. Use the `\override` and `\revert` expressions to set the `color` property.

`color.ly`



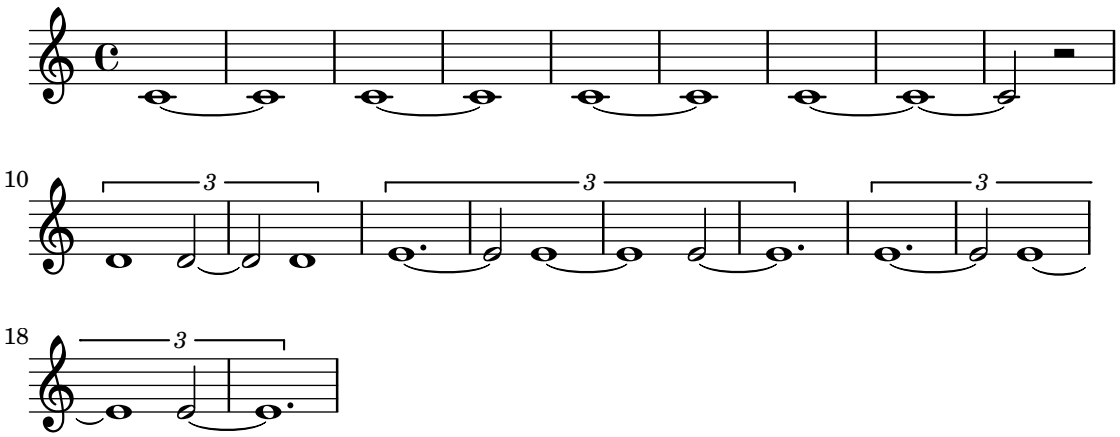
The `Completion_heads_engraver` uses dotted breve/longa durations if possible.

`completion-heads-dotted-durations.ly`
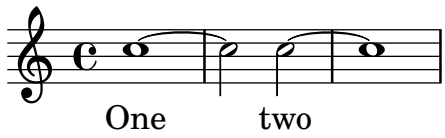


If the `Note_heads_engraver` is replaced by the `Completion_heads_engraver`, long notes, longer than `measureLength`, are split into un-scaled notes, even if the original note used a scale-factor. `completionFactor` controls this behavior.

`completion-heads-factor.ly`
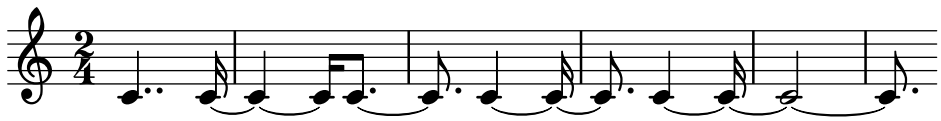


You can put lyrics under completion heads.

`completion-heads-lyrics.ly`
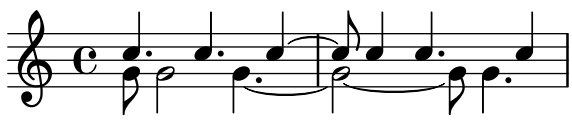


The `Completion_heads_engraver` correctly handles notes that need to be split into more than 2 parts.

`completion-heads-multiple-ties.ly`



Complex completion heads work properly in a polyphonic environment.

`completion-heads-polyphony-2.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-polyphony.ly`
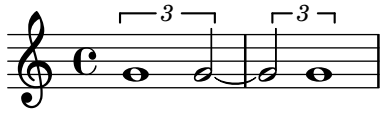


Completion heads will remember ties, so they are started on the last note of the split note.

`completion-heads-tie.ly`
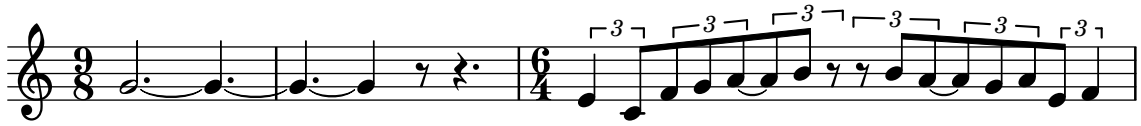


Completion heads may be used with tuplets (and compressed music) too.

`completion-heads-tuplets.ly`
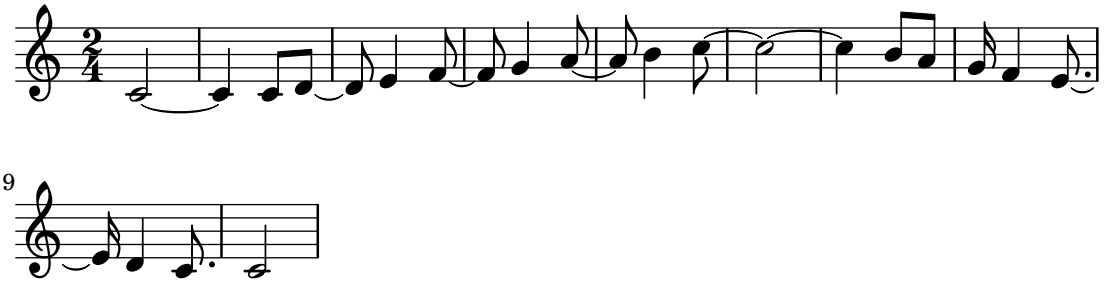


Note head completion may be broken into sub-bar units by setting the `completionUnit` property.

`completion-heads-unit.ly`
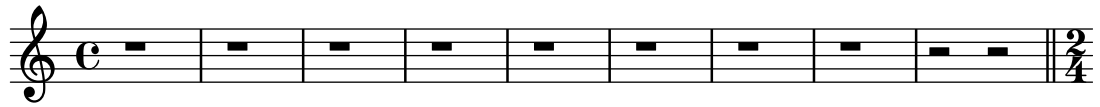


If the `Note_heads_engraver` is replaced by the `Completion_heads_engraver`, notes that cross bar lines are split into tied notes.
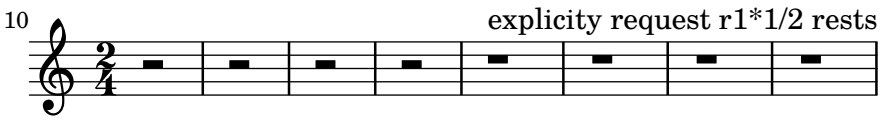
`completion-heads.ly`



If the `Rest_engraver` is replaced by the `Completion_rest_engraver`, long rests, longer than `measureLength`, are split into un-scaled rests, even if the original duration used a scale-factor. `completionFactor` controls this behavior.

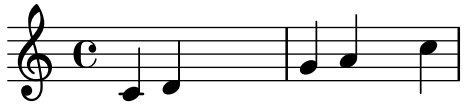
`completion-rest.ly`
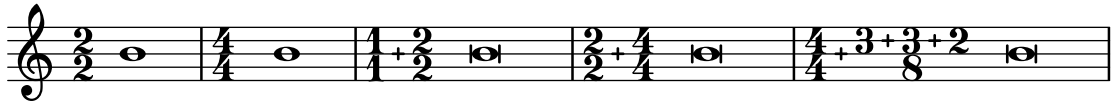




This tests `\once` applied to multiple property operations.

`complex-once.ly`
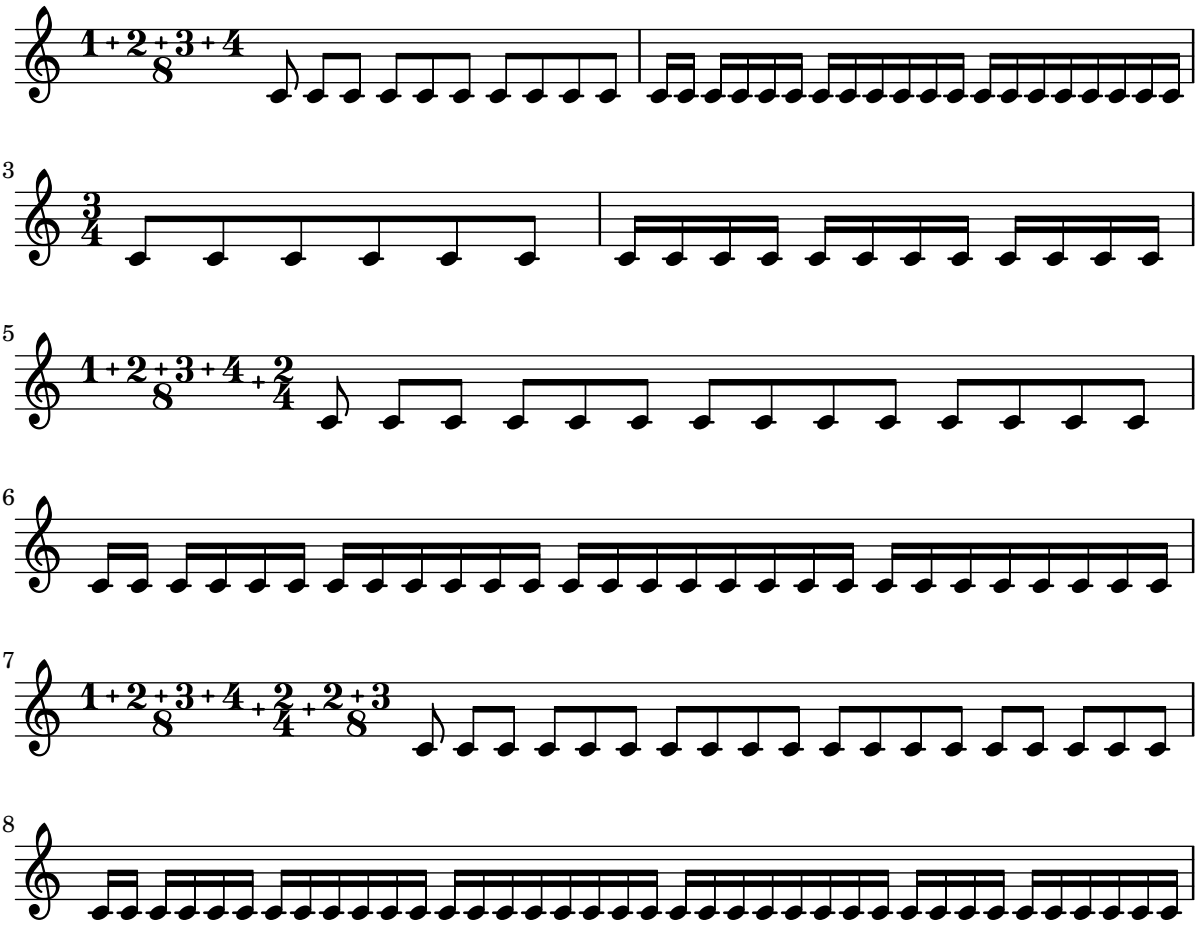


Simple-fraction components of a compound time signature are numeric regardless of the time signature style.
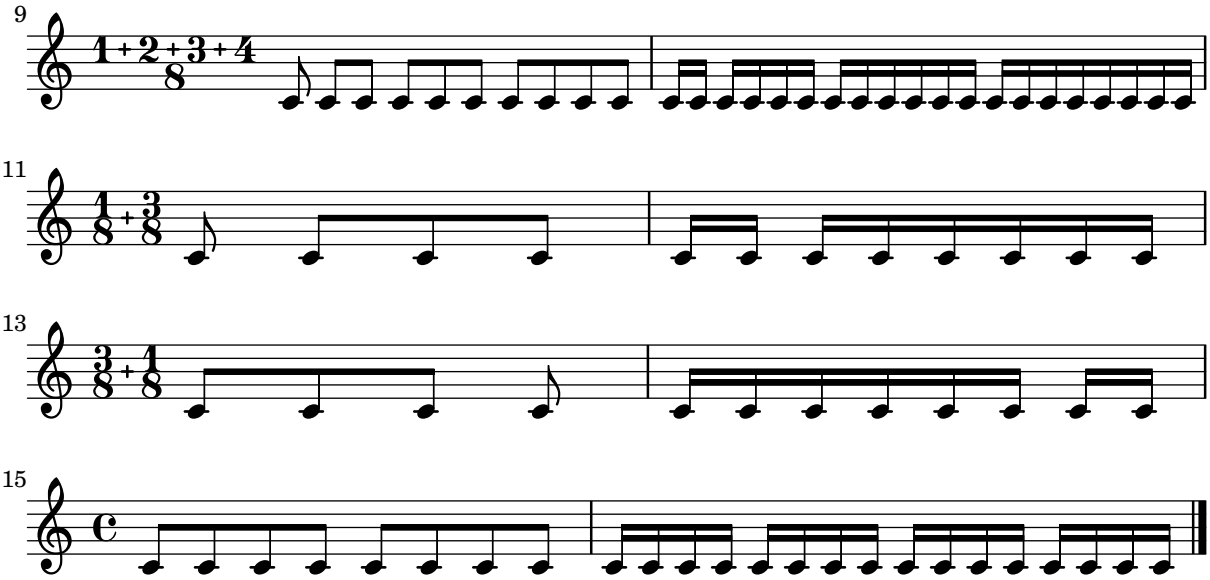
`compound-time-signature-style.ly`



Create compound time signatures. The argument is a Scheme list of lists. Each list describes one fraction, with the last entry being the denominator, while the first entries describe the summands in the enumerator. If the time signature consists of just one fraction, the list can be given directly, i.e. not as a list containing a single list. For example, a time signature of (3+1)/8 + 2/4 would be created as `\compoundMeter #'((3 1 8) (2 4))`, and a time signature of (3+2)/8 as `\compoundMeter #'((3 2 8))` or shorter `\compoundMeter #'(3 2 8)`.

`compound-time-signatures.ly`





A `\defaultchild` cycle does not induce an endless loop. The output of this test is not important.

`context-defaultchild-cycle.ly`

`\defaultchild` can be overridden in a context definition. CREATED should appear in the left margin.

`context-defaultchild-def.ly`



`\defaultchild` can be overridden in `\with` blocks. CREATED should appear in the left margin.

`context-defaultchild-mod.ly`



`\denies` *context* in a context definition cancels a prior `\defaultchild` *context*. CREATED should appear in the left margin.

`context-denies-defaultchild-def.ly`



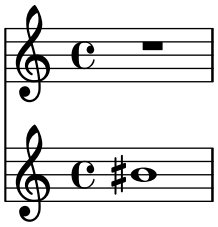
`\denies` *context* in a `\with` block cancels a prior `\defaultchild` *context*. CREATED should appear in the left margin.

`context-denies-defaultchild-mod.ly`



A `\denies` statement in a `\with` block applies to the local context only; it does not change the global context definition. The lower staff should hold a B-sharp.
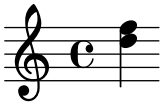
`context-denies-nondestructive-mod.ly`



If the descend-to-context function cannot find or create its context below the current context, then it does not create its context anywhere, and it leaves the current context unchanged.

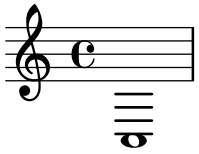
The expected output of this test is one staff with two notes.

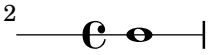
`context-descend-only-not-found.ly`



a staff should die if there is reference to it.

`context-die-staff.ly`
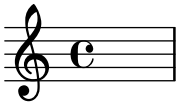




`\context` finds a child by type and ID even when the parent also matches.

```
StaffGroup A
   \
  StaffGroup B (from here, find StaffGroup A)
     \
    StaffGroup A (this is found)
```

RESULT should appear in the left margin.

`context-find-child.ly`

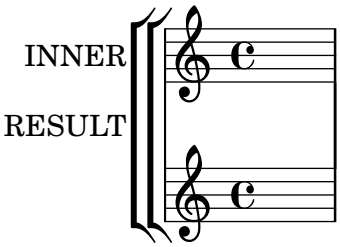
RESULT 

`\context` finds the current context by type and ID even when there are matching contexts both above and below.

```
StaffGroup A
   \
  StaffGroup A (from here, find StaffGroup A)
     \
    StaffGroup A
```

INNER and RESULT should appear in the left margin.

`context-find-current.ly`

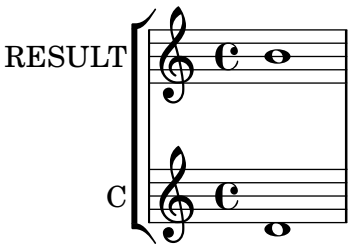


`\context` finds a grandchild by type and ID when there are multiple matching contexts.

```
        StaffGroup A (from here, find Staff D)
       /            \
 StaffGroup B  StaffGroup C
    /                \
 Staff D            Staff D
```

RESULT and either B or C should appear in the left margin.

`context-find-grandchild-ambiguous.ly`



`\context` finds a grandchild by type and ID even when the parent also matches.

```
 StaffGroup A
    \
   StaffGroup B (from here, find StaffGroup A)
      \
     StaffGroup C
        \
       StaffGroup A (this is found)
```

RESULT should appear in the left margin.

`context-find-grandchild.ly`

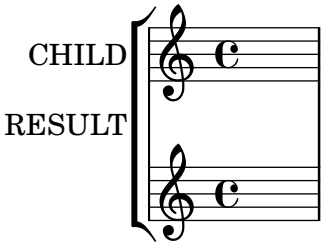


`\context` can find the parent context by type and ID.

```
 StaffGroup A
    \
   StaffGroup B (from here, find StaffGroup A)
```

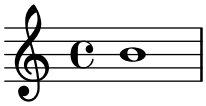
CHILD and RESULT should appear in the left margin.

`context-find-parent.ly`



Attempting to find a Score context by alias before it exists triggers creation of a Score context. The output should have a note on the middle line of the staff.

`context-find-score-alias.ly`

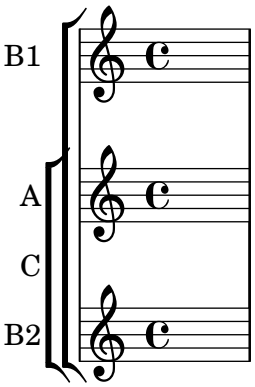


`\context` creates a new context rather than finding a matching context in another branch.

```
          StaffGroup A
         /            \
  StaffGroup B  StaffGroup C (from here, find StaffGroup B)
                        \
                    [StaffGroup B] (this is created)
```
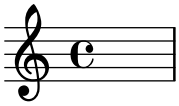
B1, A, C, and B2 should appear in the left margin.

`context-find-sibling.ly`
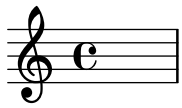


User code is not allowed to access the Global context. The visual output of this test is not important.

`context-global-find.ly`



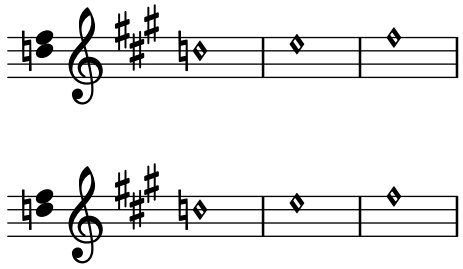
User code is not allowed to create a Global context. The visual output of this test is not important.

`context-global-new.ly`



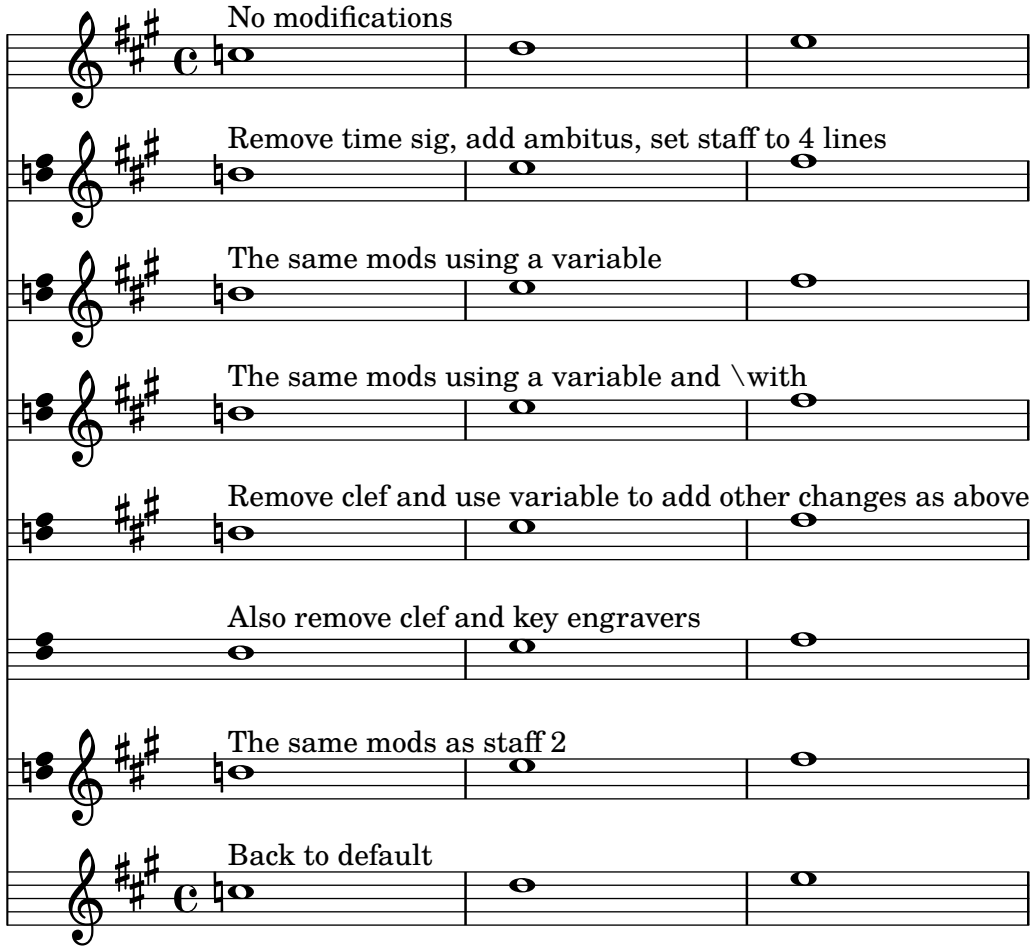
Context modifications can be stored into a variable as a \with object. They can be later inserted directly into a context definition.

`context-mod-context.ly`



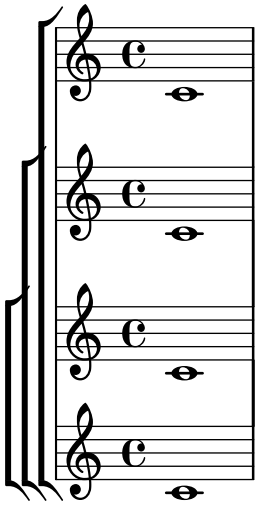
Context modifications can be stored into a variable as a \with object. They can be later inserted into another \with block.

`context-mod-with.ly`
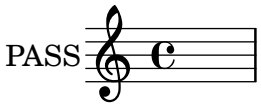


Contexts of the same type can be nested.

`context-nested-staffgroup.ly`
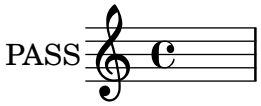


`\new` can create a child of the same type and name as its parent. PASS should appear in the left margin.

`context-new-child-same-name.ly`



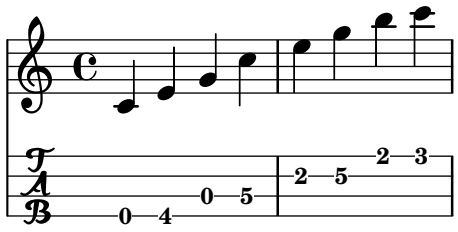
`\new` can create a sibling of an existing context with the same type and name. The instrument name should be PASS.

`context-new-sibling-same-name.ly`



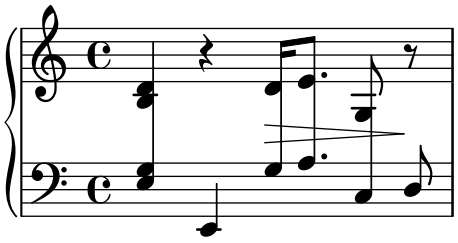
Using \contextStringTuning does not break compiling.

`context-string-tuning.ly`
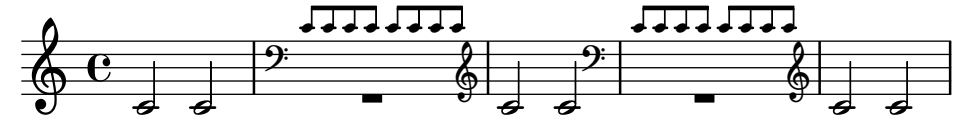


Test for cross-staff stems. The test produces a piano staff with cross-staff connected crochet, semi-quaver, dotted quaver (beamed with the semi-quaver) and finally a quaver. All stems should connect, showing correct spacing and stem length. The lower connected notes should have no flags.

`cross-staff-stems.ly`
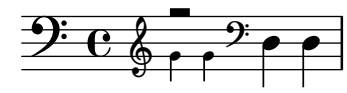


Cue clefs can be printed after a barline.

`cue-clef-after-barline.ly`



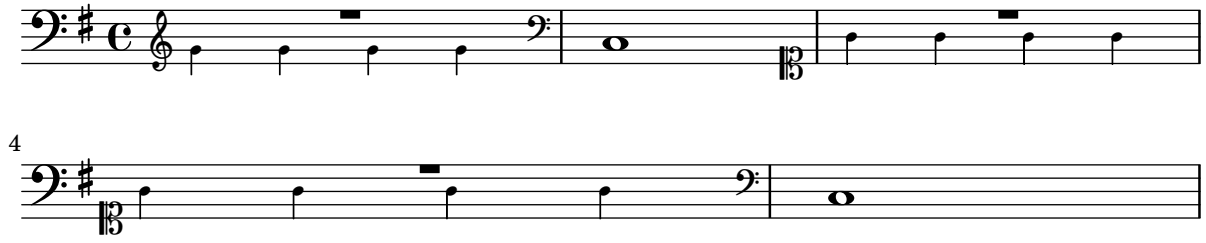
Clefs for cue notes at the start of a score should print the standard clef plus a small cue clef after the time/key signature.

`cue-clef-begin-of-score.ly`



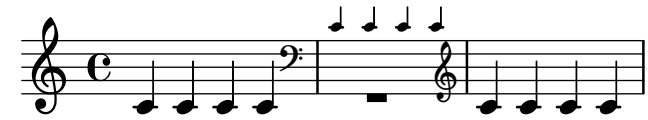
Clefs for cue notes should not influence the printed key signature.

`cue-clef-keysignature.ly`



Cue clefs can be printed manually.

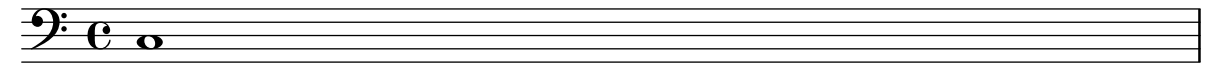
`cue-clef-manually.ly`

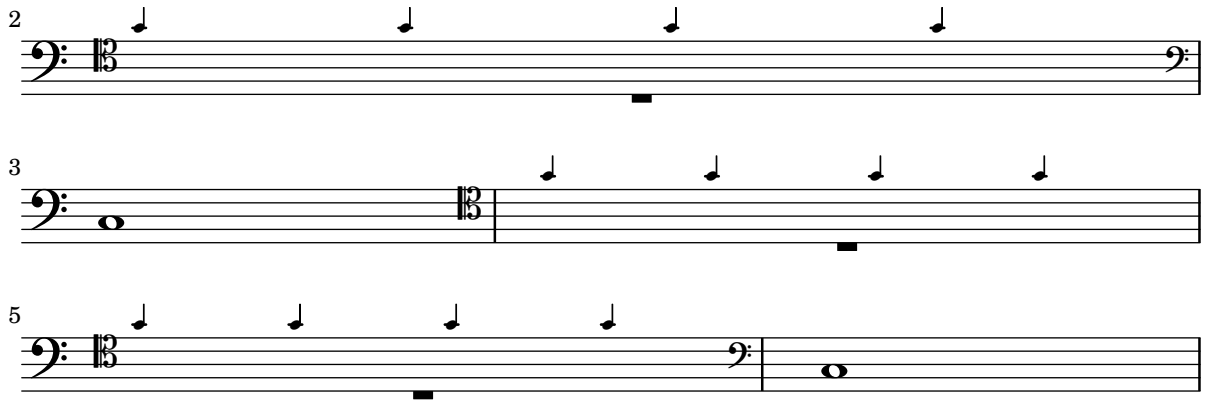


Clefs for cue notes and line breaks. If the cue notes start in a new line, the cue clef should not be printed at the end of the previous line. Similarly, an end clef for cue notes ending at a line break should only be printed at the end of the line.

Cue notes going over a line break should print the standard clef on the new line plus an additional cue clef after the time/key signature.

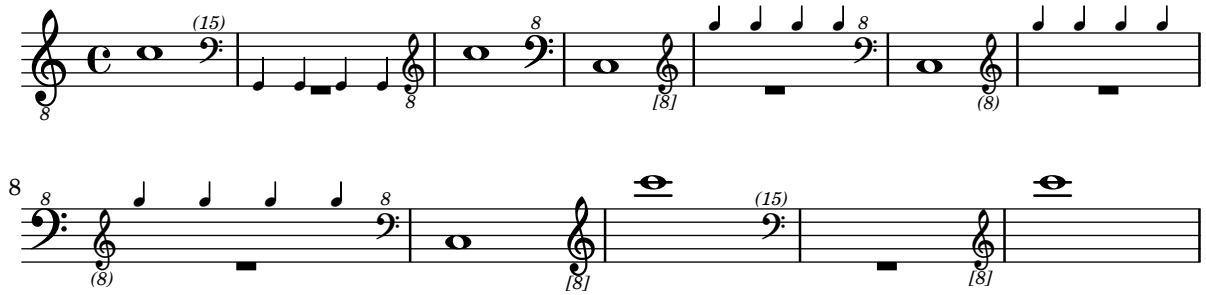
`cue-clef-new-line.ly`
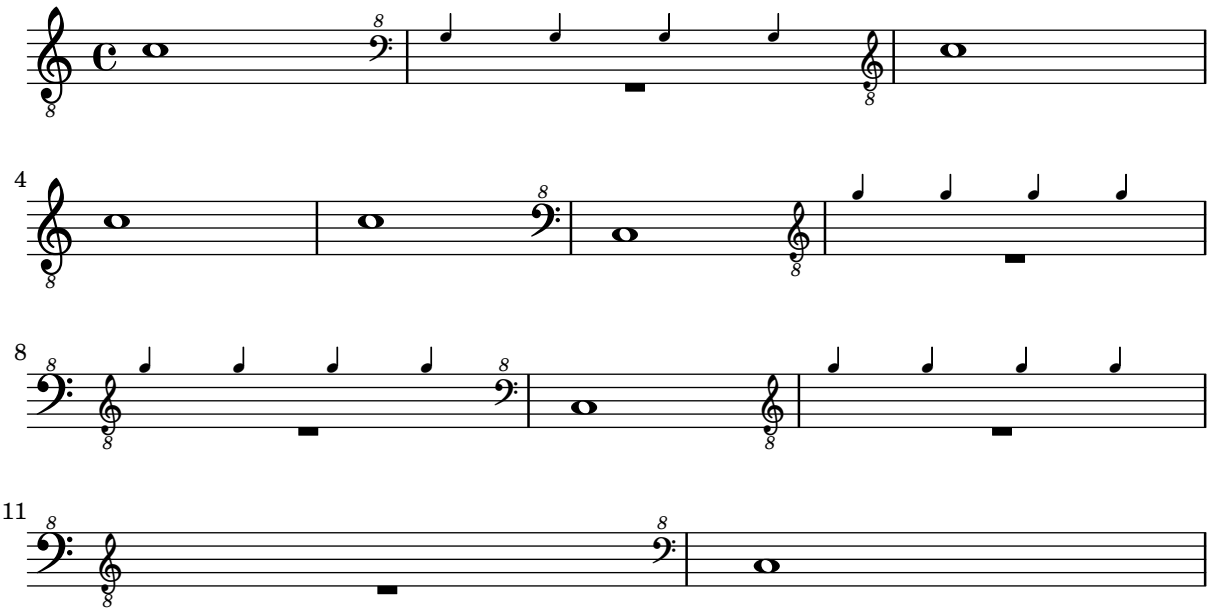




Optional transposition for clefs for cue notes is supported by using parentheses or brackets around the transposition number.

`cue-clef-transposition-optional.ly`
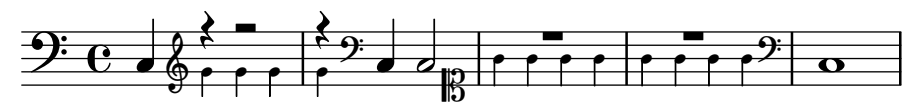


Transposition for clefs for cue notes.

`cue-clef-transposition.ly`



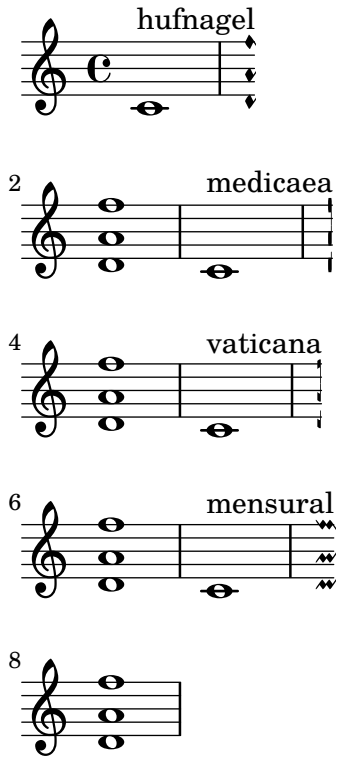
Clefs for cue notes: Print a cue clef at the begin of the cue notes and a canceling clef after the cue notes.

`cue-clef.ly`



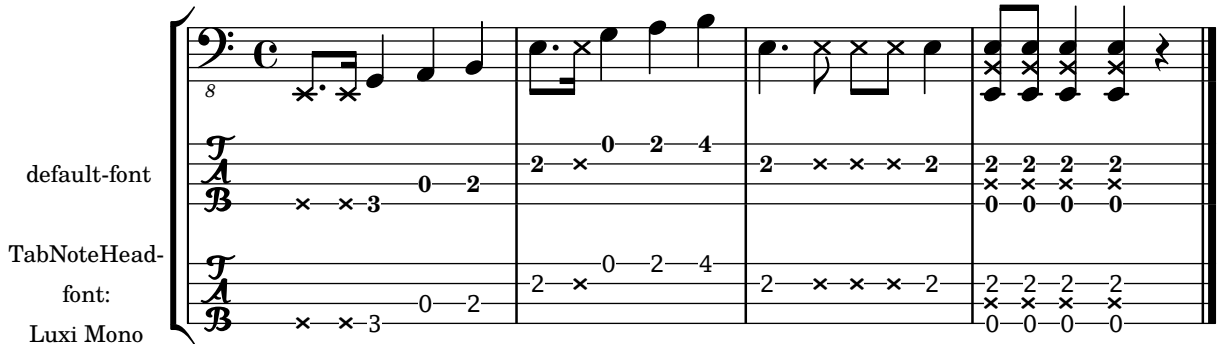
Custodes may be engraved in various styles.

`custos.ly`



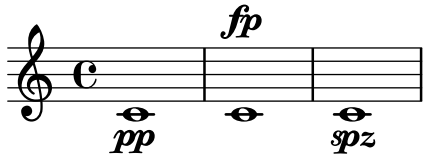
Muted notes (also called dead notes) are supported within normal staves and tablature. They are printed correctly, even if another font for TabNoteHead is used.

`dead-notes.ly`



Tests `define-event-function` by creating a trivial function converting a markup into a dynamic script post-event. As opposed to music functions, a direction indicator is not required.

`define-event-function.ly`

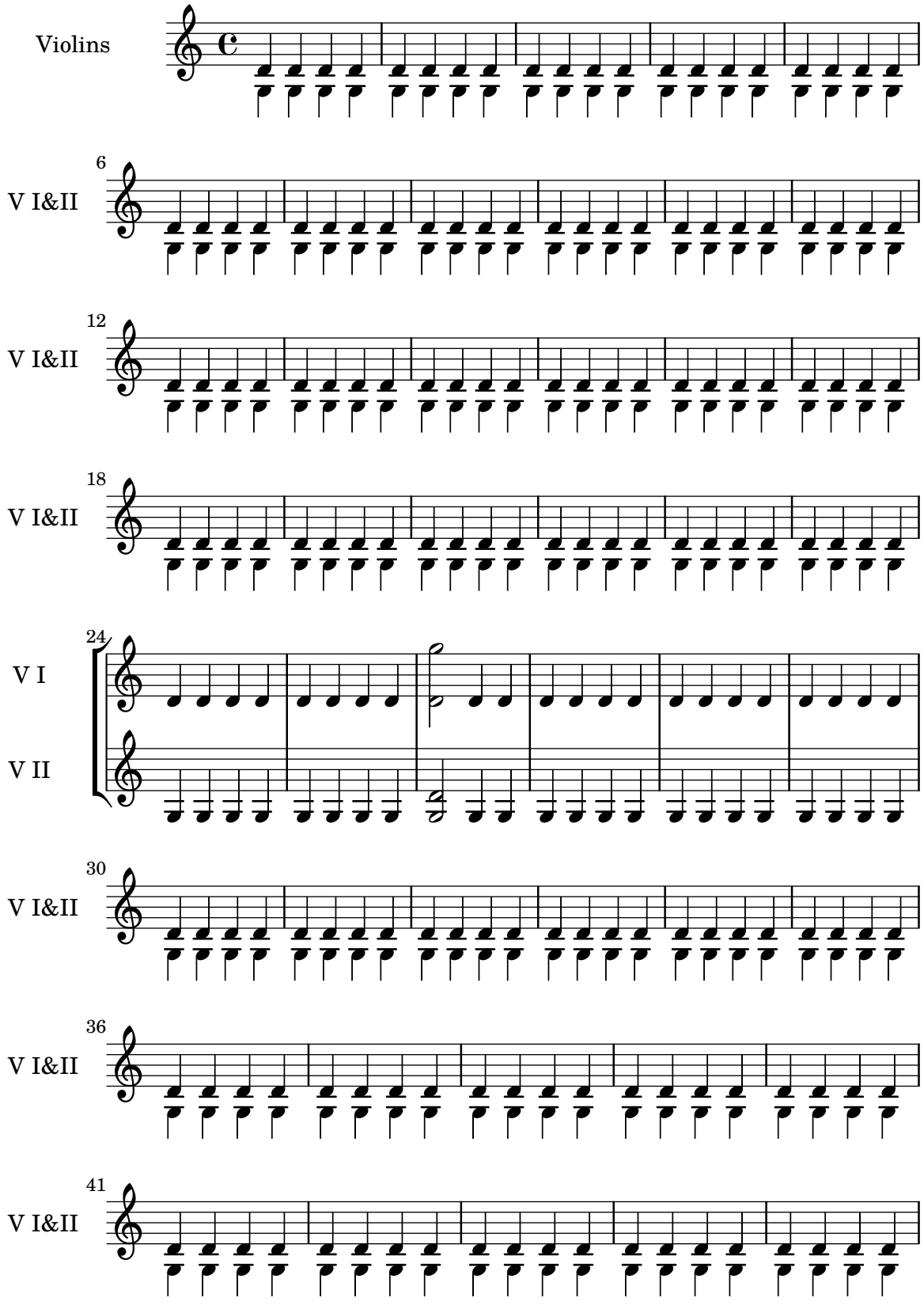


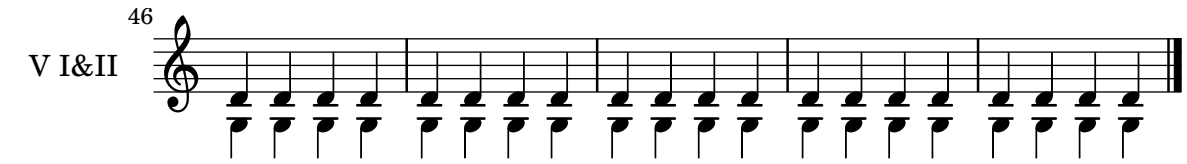
This is a test of the display-lily-music unit. Problems are reported on the stderr of this run.

`display-lily-tests.ly`

The `VerticalAxisGroup.remove-layer` property can be used for typesetting temporary divisi staves where the switch to split staves is done only at line breaks such that all complex passages are rendered in separate staves.
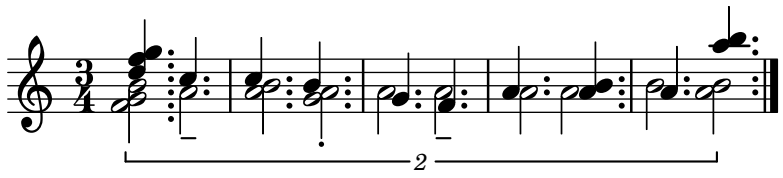
`divisi-staves.ly`
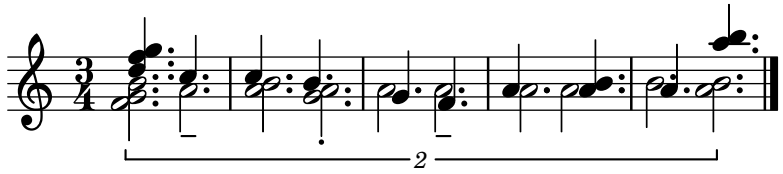




Dot Columns are engraved in the Staff by default, enabling dots to move vertically to make room for dots from another voice. If Dot_column_engraver is moved to Voice, separate dot columns are engraved, and these dots avoid notes in other voices.

`dot-column-engraver.ly`
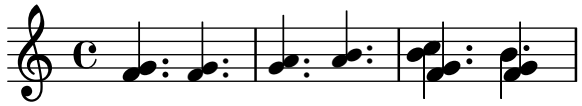


move Dot_column_engraver to Voice :



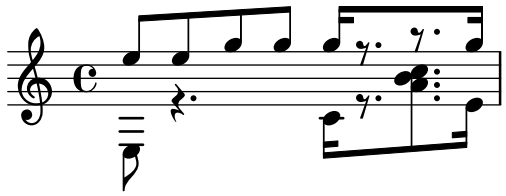
Dots and note-heads should not collide.

`dot-column-note-collision.ly`
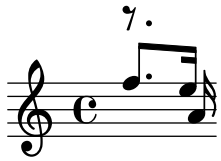


Dot columns do not trigger beam slanting too early. This input should compile with no programming error message, and the dots should be correctly placed on their rests.

`dot-column-rest-collision.ly`
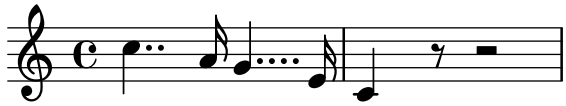


Dot columns should not trigger vertical spacing before line breaking. If the regtest issues a programming_error saying that vertical spacing has been called before line breaking, it has failed.

`dot-column-vertical-positioning.ly`
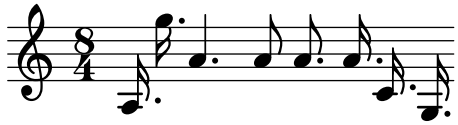


The `dot-count` property for `Dots` can be modified by the user.

dot-dot-count-override.ly
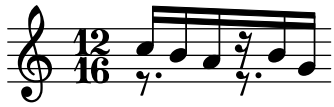


Dots move to the right when a collision with the (up)flag happens.

dot-flag-collision.ly
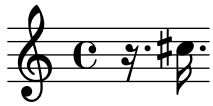


Dotted rests connected with beams do not trigger premature beam calculations. In this case, the beam should be sloped, and there should be no programming_error() warnings.

dot-rest-beam-trigger.ly



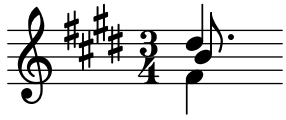
The dots on a dotted rest are correctly accounted for in horizontal spacing.

dot-rest-horizontal-spacing.ly



in collisions, the dots of outer voices avoid stems and flags of the inner voices.

dot-up-voice-collision.ly

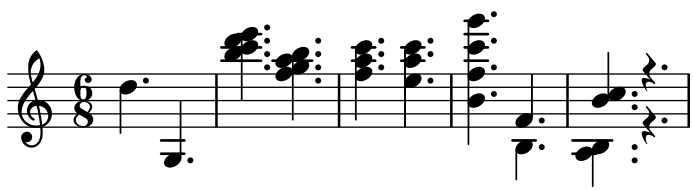


Both noteheads and rests can have dots. Augmentation dots should never be printed on a staff line, but rather be shifted vertically. They should go up, but in case of multiple parts, the down stems have down shifted dots. In case of chords, all dots should be in a column. The dots follow the shift of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

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

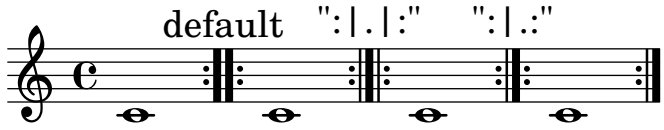
- keeping dots off staff lines,
- keeping dots close to their note heads,
- moving dots in the direction specified by the voice,
- moving dots up.

dots.ly



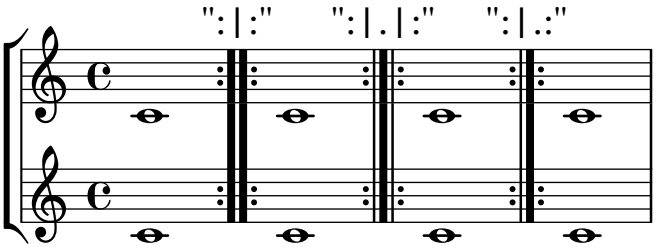
For volte, the style of double repeats can be set using `doubleRepeatType`.

`double-repeat-default-volta.ly`



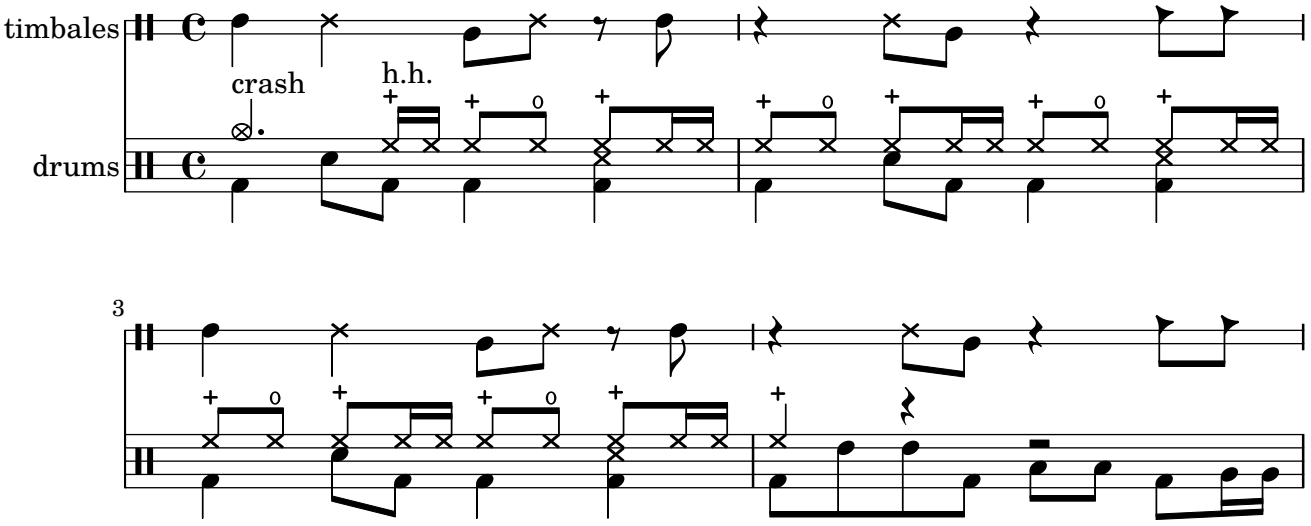
Three types of double repeat bar line are supported.

`double-repeat.ly`



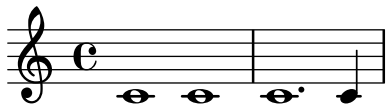
In drum notation, there is a special clef symbol, drums are placed to their own staff positions and have note heads according to the drum, an extra symbol may be attached to the drum, and the number of lines may be restricted.

`drums.ly`



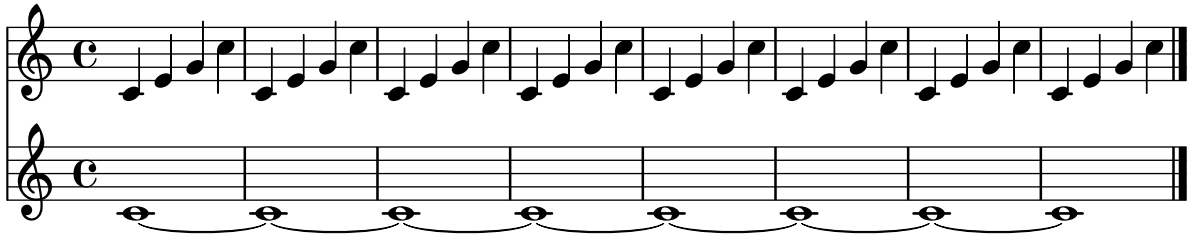
The compression factor of a duration identifier is correctly accounted for by the parser.

`duration-identifier-compressed.ly`



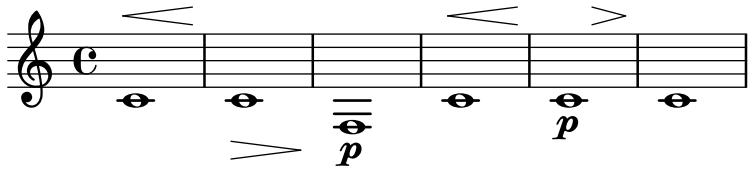
Duration multipliers can be specified as scheme expressions, either as rationals or as a moment.

`duration-multiplier-scheme.ly`



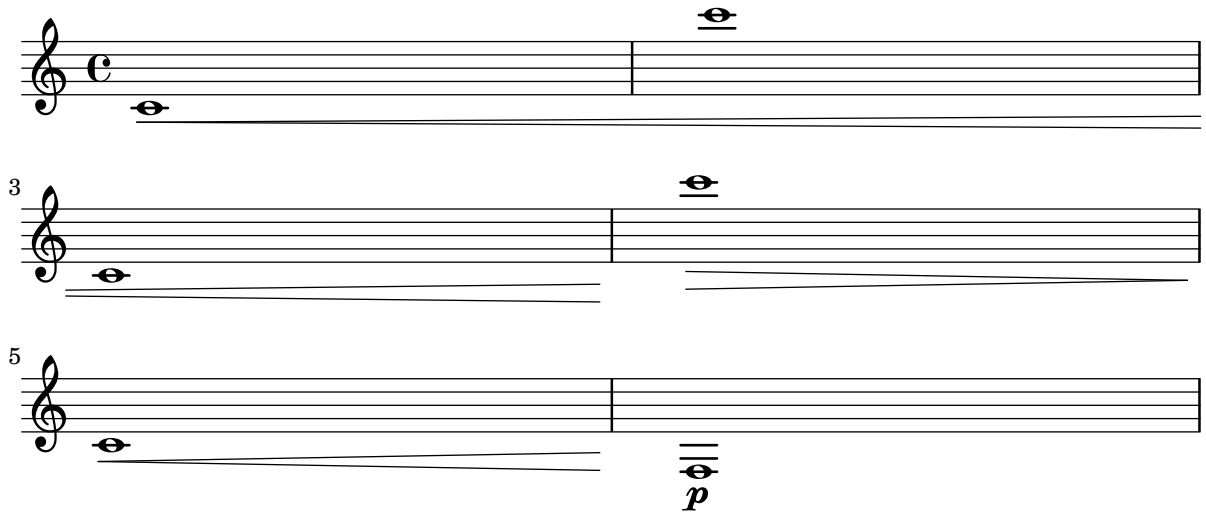
If a dynamic has an explicit direction that differs from the dynamic line spanner's direction, automatically break the dynamic line spanner.

`dynamics-alignment-autobreak.ly`



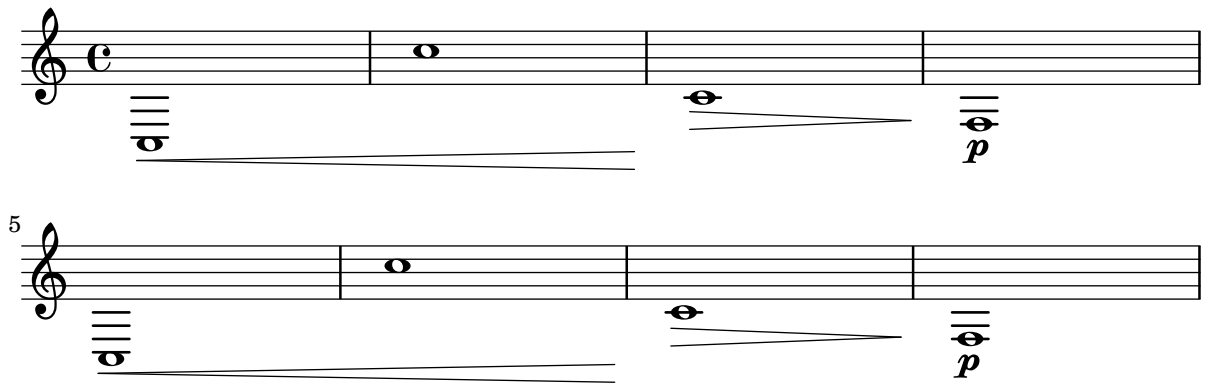
`\breakDynamicSpan` shall also work if a dynamic spanner crosses a line break.

`dynamics-alignment-breaker-linebreak.ly`



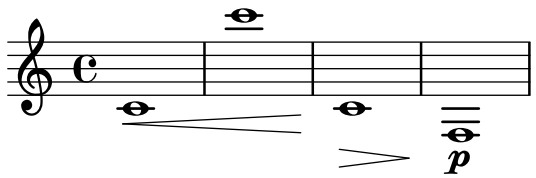
`\breakDynamicSpan` work whether it is placed together with the start or the end of a spanner. Both lines should be identical.

`dynamics-alignment-breaker-order.ly`



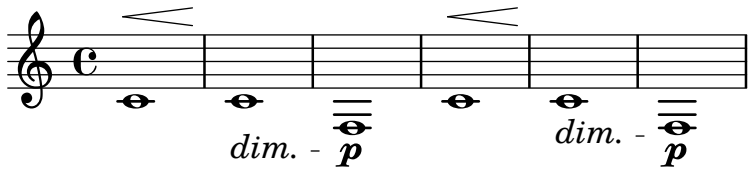
`\breakDynamicSpan` shall only have an effect on the current spanner, not on subsequent spanners.

`dynamics-alignment-breaker-subsequent-spanner.ly`



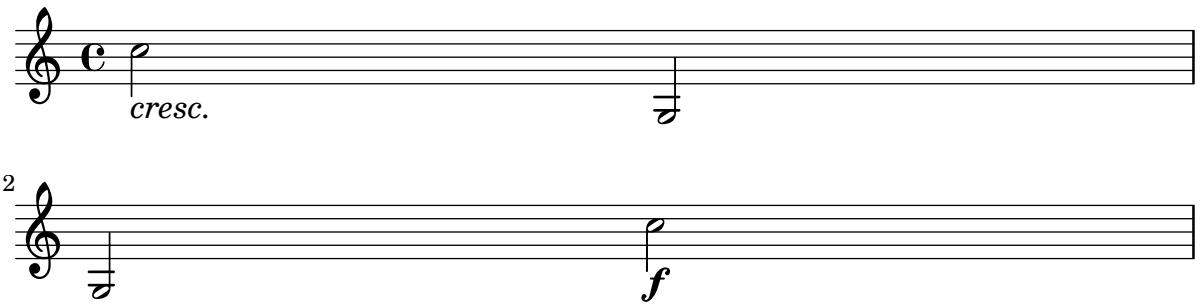
Hairpins, DynamicTextSpanners and dynamics can be positioned independently using `\breakDynamicSpan`, which causes the alignment spanner to end prematurely.

`dynamics-alignment-breaker.ly`



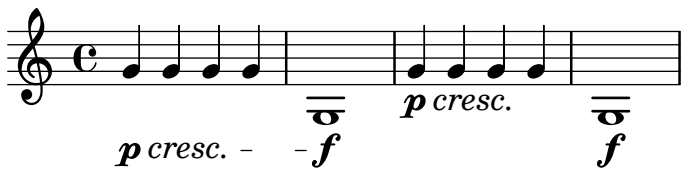
Setting the style of a `DynamicTextSpanner` to `'none` to hide the line altogether should also work over line breaks.

`dynamics-alignment-no-line-linebreak.ly`



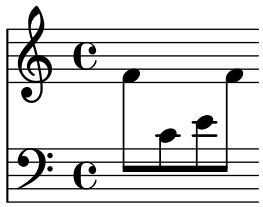
If the line for a `DynamicTextSpanner` is hidden, the alignment spanner for dynamics is ended early. This allows consecutive dynamics to be unlinked.

`dynamics-alignment-no-line.ly`



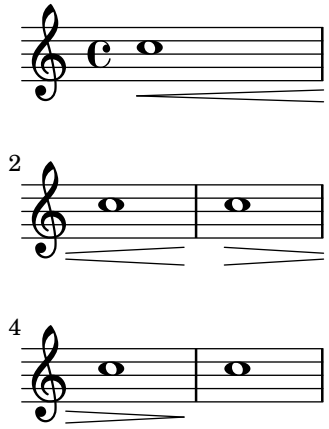
Cross-staff `Dynamic` does not trigger a cyclic dependency for direction look-up.

`dynamics-avoid-cross-staff-stem-3.ly`



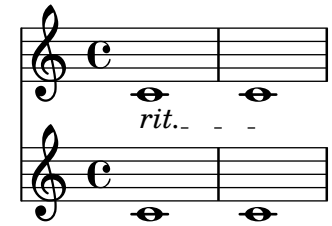
When a hairpin is broken, the broken parts should be open at the ‘breaking point’.

`dynamics-broken-hairpin.ly`



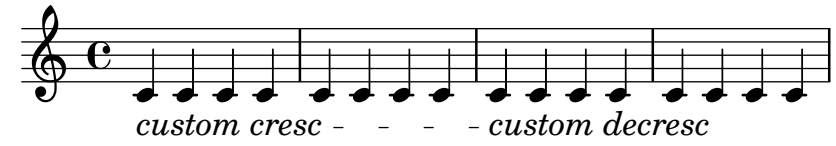
Text spanners work in the `Dynamics` context.

`dynamics-context-textspan.ly`



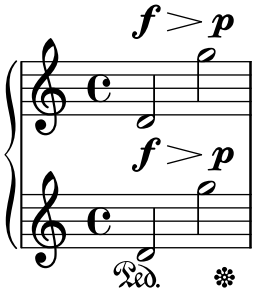
Postfix functions for custom crescendo text spanners. The spanners should start on the first note of the measure. One has to use -\mycresc, otherwise the spanner start will rather be assigned to the next note.

`dynamics-custom-text-spanner-postfix.ly`



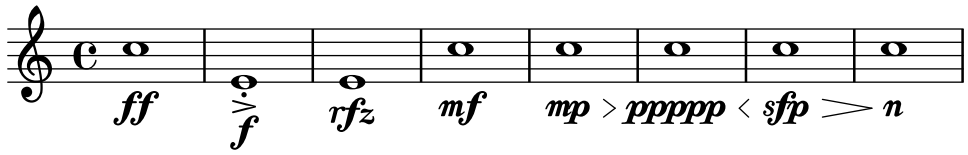
An empty Dynamics context does not confuse the spacing.

`dynamics-empty.ly`



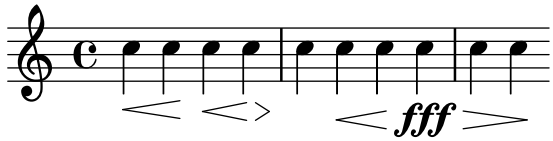
Dynamic letters are kerned, and their weight matches that of the hairpin signs. The dynamic scripts should be horizontally centered on the note head. Scripts that should appear closer to the note head (staccato, accent) are reckoned with.

`dynamics-glyphs.ly`



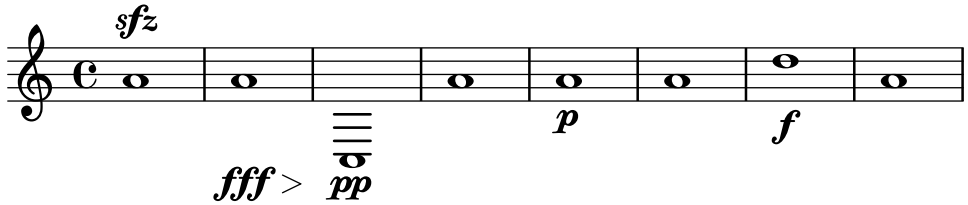
Hairpins extend to the extremes of the bound if there is no adjacent hairpin or dynamic-text. If there is, the hairpin extends to the center of the column or the bound of the text respectively.

`dynamics-hairpin-length.ly`
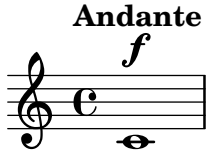


Dynamics appear below or above the staff. If multiple dynamics are linked with (de)crescendi, they should be on the same line. Isolated dynamics may be forced up or down.

`dynamics-line.ly`
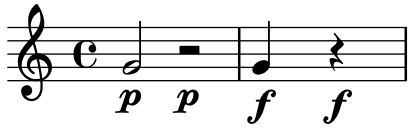


`DynamicText`, `DynamicLineSpanner`, and `Hairpin` do not have `outside-staff-priority` in `Dynamics` contexts. This allows grobs with `outside-staff-priority` set to be positioned above and below them.

`dynamics-outside-staff-priority.ly`



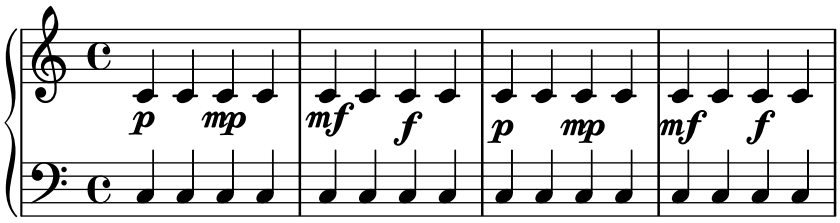
Text dynamics are positioned correctly on rests, i.e., centered on the parent object.

`dynamics-rest-positioning.ly`
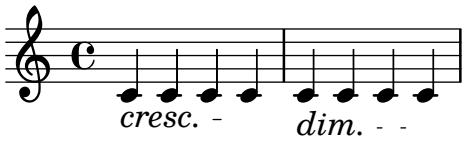


The `X-offset` of `DynamicText` grobs in a `Dynamics` context should be averaged over the center of `NoteColumn` grobs in the `DynamicText`'s `PaperColumn`.

dynamics-text-dynamics-context.ly
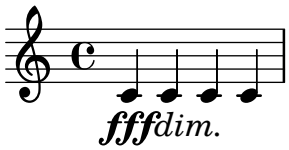


The left text of a DynamicTextSpanner is left-aligned to its anchor note.

dynamics-text-left-text-alignment.ly



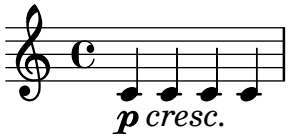
The space between an absolute dynamic and a dynamic text span can be changed using 'right-padding.

dynamics-text-right-padding.ly



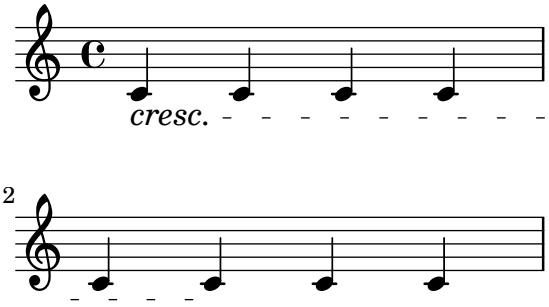
left attach dir for text crescendi starting on an absolute dynamic is changed, so cresc. and the absolute dynamic don't overstrike.

dynamics-text-spanner-abs-dynamic.ly
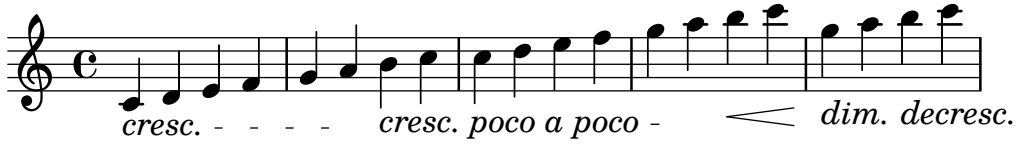


The 2nd half of the cresc. stays at a reasonable distance from the notes.

dynamics-text-spanner-padding.ly
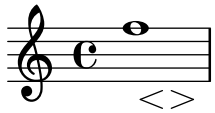


The \cresc, \dim and \decresc spanners are now postfix operators and produce one text spanner. Defining custom spanners is also easy. Hairpin and text crescendi can be easily mixed. \< and \> produce hairpins by default, \cresc etc. produce text spanners by default.

`dynamics-text-spanner-postfix.ly`



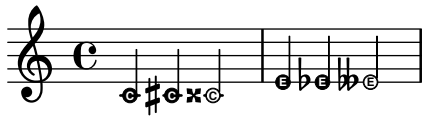
Crescendi may start off-notes, however, they should not collapse into flat lines.

`dynamics-unbound-hairpin.ly`
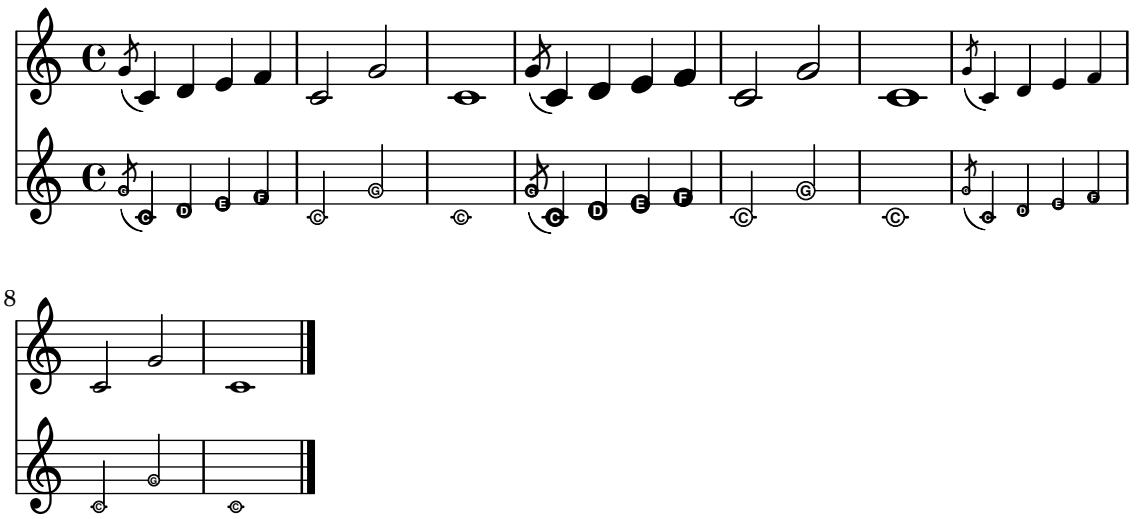


Accidentals are positioned correctly when using Easy notation.

`easy-notation-accidentals.ly`



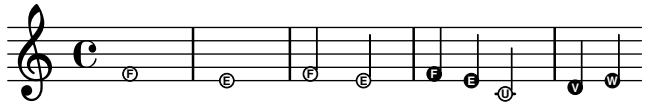
Easy noteheads should be scalable in size, like in grace notes.

`easy-notation-size.ly`



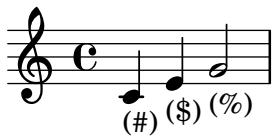
Easy-notation (or Ez-notation) prints names in note heads. You also get ledger lines, of course.

`easy-notation.ly`



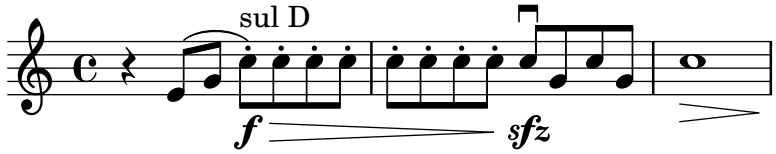
Strings and comments inside of `#{...#}` should not be confiusing to the embedded LilyPond parser. If this test succeeds, three notes with `(#)`, `($)`, and `(%)` underneath will get displayed here.

`embedded-strings-comments.ly`



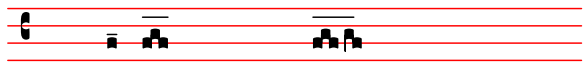
Empty chords accept articulations, occupy no time, and leave the current duration unchanged.

`empty-chord.ly`



An episema can be typeset over a single neume or a melisma. Its position is quantized between staff lines.

`episema.ly`

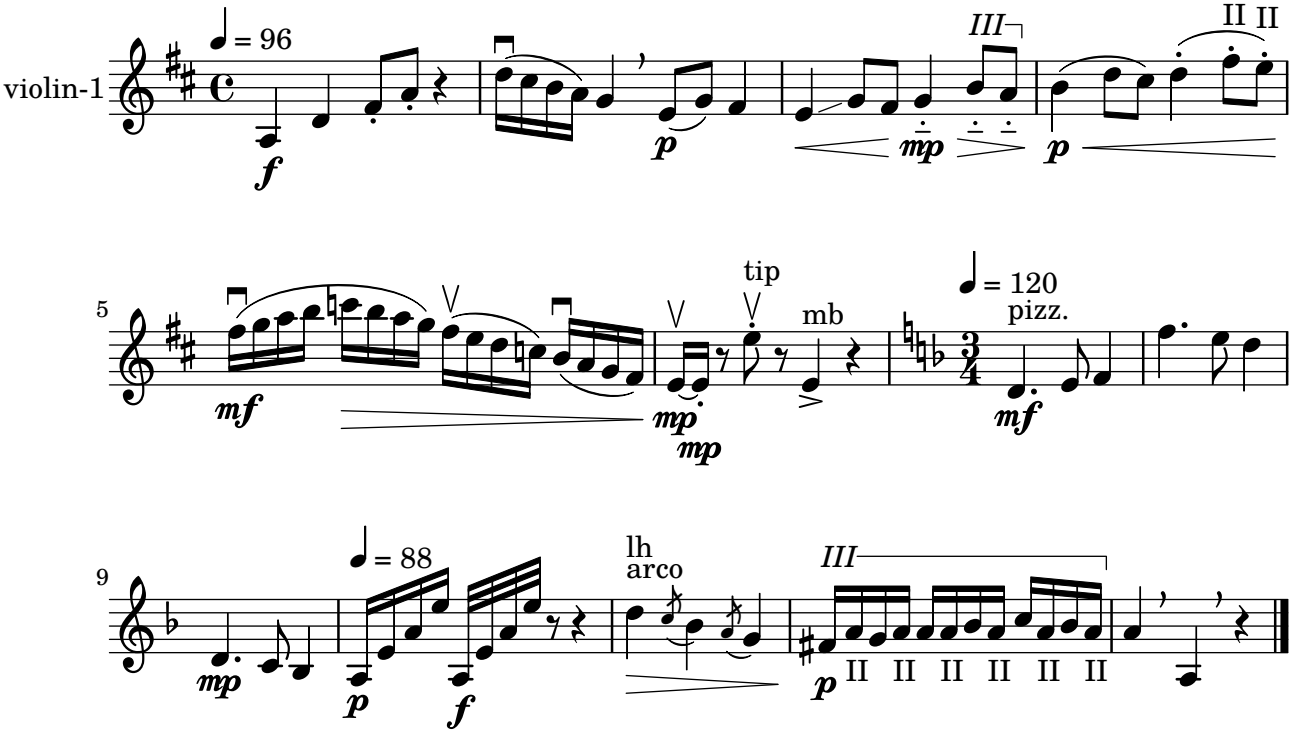


Music events can be extracted from a score with event listeners.

`event-listener-output.ly`
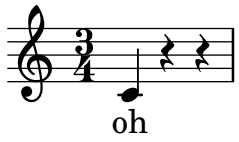
Black-box Testing

Graham Percival



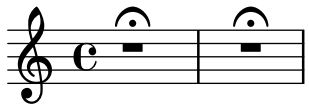
A mode switching command like `\lyricsto` will 'pop state' when seeing the lookahead token `\time`, a music function, after its non-delimited argument. This must not cause the extra token parsing state for the music function to disappear.

`extratoken.ly`



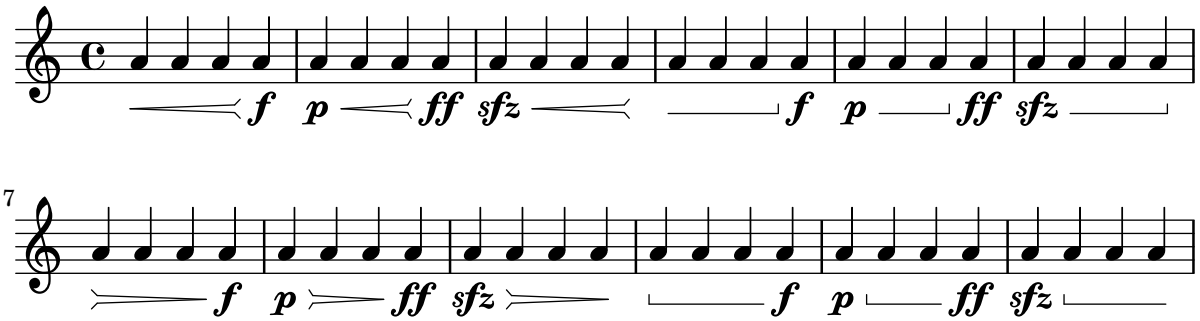
Fermatas over multimeasure rests are positioned as over normal rests.

`fermata-rest-position.ly`
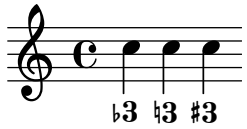


LilyPond creates hairpins found in Ferneyhough scores.

`ferneyhough-hairpins.ly`



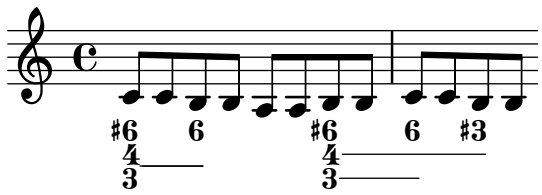
Bass figures can carry alterations.

`figured-bass-alteration.ly`



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

`figured-bass-continuation-center.ly`



Figured bass extender for figures of different width (e.g. with alteration or two-digit figures) should still stop at the same position.

`figured-bass-continuation-end-position.ly`



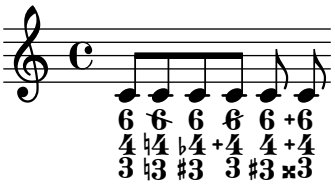
By adorning a bass figure with `\!`, an extender may be forbidden.

`figured-bass-continuation-forbid.ly`



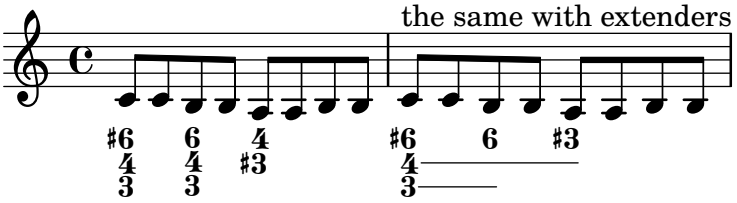
Figured bass extender lines shall be broken when a figure has a different alteration, augmentation or diminishment.

`figured-bass-continuation-modifiers.ly`
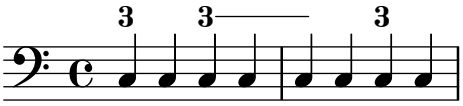


Figured bass extender lines run between repeated bass figures. They are switched on with `useBassFigureExtenders`

`figured-bass-continuation.ly`



Bass figures and extenders shall also work correctly if the figure has a different duration than the bass note. In particular, if a timestep does not have a new figure (because the old figure still goes on), extenders should be drawn and not be reset.

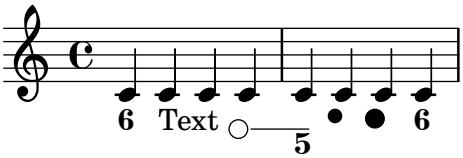
`figured-bass-durations.ly`



When using extender lines in FiguredBass, markup objects should be treated like ordinary figures and work correctly with extender lines.
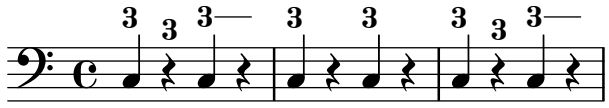
Extenders should only be used if the markup is really identical.

`figured-bass-extenders-markup.ly`



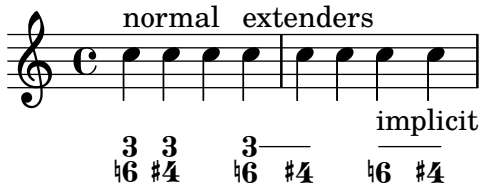
When figures appear inside a voice, `ignoreFiguredBassRest` causes all figures on rests to be discarded and all spanners ended. If set to `#f`, figures on rests are printed.

`figured-bass-ignore-rest.ly`



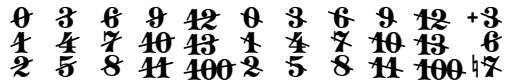
Implicit bass figures are not printed, but they do get extenders.

`figured-bass-implicit.ly`



Figured bass supports numbers with slashes through them.
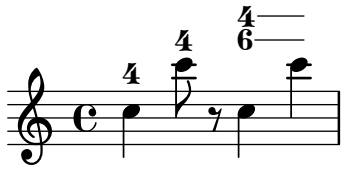
`figured-bass-slashed-numbers.ly`



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.

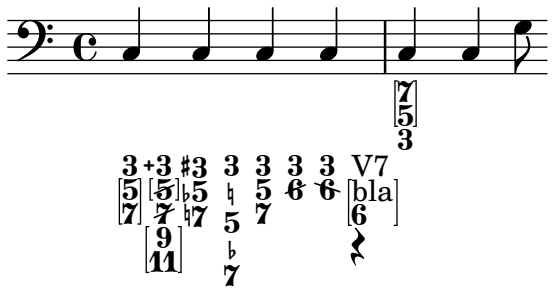
`figured-bass-staff.ly`



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, backslashes and pluses.

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

`figured-bass.ly`



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

fill-line-test.ly

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

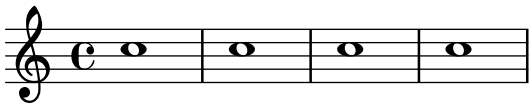
1 2 3 4 5 6 7 8 9 10

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
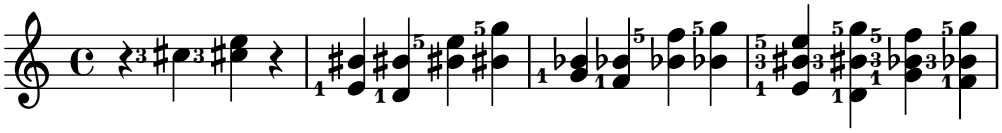


Context modification via `\with` filters translators of the wrong type: performers for an `Engraver_group` and engravers for a `Performer_group`. In this test, the `Instrument_name_engraver` is added to a `StaffGroup`, but does not affect midi output, since it is filtered out.

`filter-translators.ly`
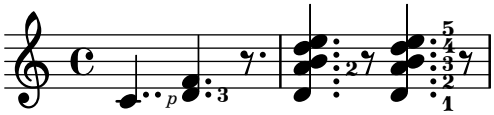


Scripts left of a chord avoid accidentals.

`finger-chords-accidental.ly`



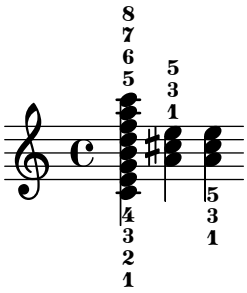
Scripts right of a chord avoid dots.

`finger-chords-dot.ly`



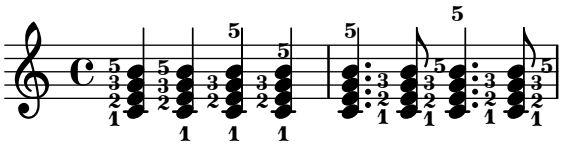
Ordering of the fingerings depends on vertical ordering of the notes, and is independent of up/down direction.

`finger-chords-order.ly`



It is possible to associate fingerings uniquely with notes. This makes it possible to add horizontal fingerings to notes. Fingering defaults to not clearing flags and stems unless there is a collision or a beam.

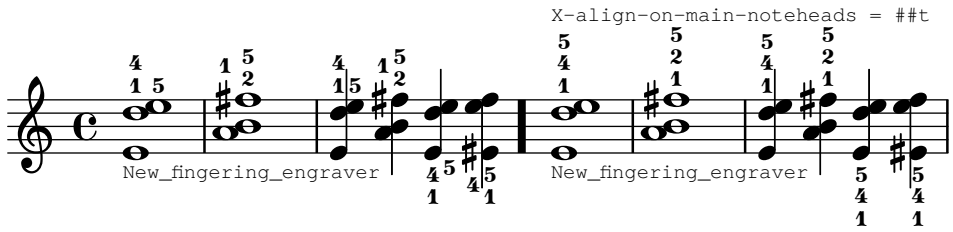
`finger-chords.ly`



`Fingering` grobs created by the `New_fingering_engraver` (i.e. fingerings entered outside `<>`) with `fingeringOrientations` set to `up` or `down` avoid accidentals of displaced notes that might get into the way in chords containing adjacent notes (seconds) or unison notes.

With `\override Fingering.X-align-on-main-noteheads = ##t`, the fingerings oriented `up` and `down` will be arranged in a straight column aligned on the noteheads on the "correct" side of the stem.

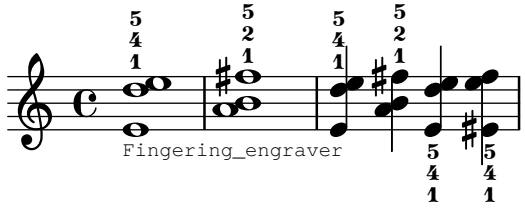
`fingering-adjacent-note-chord-new.ly`



`Fingering` grobs created by the `Fingering_engraver` (i.e. fingerings entered inside `<>`) above/below chords containing adjacent notes (seconds) or unison notes should be aligned on the main noteheads, i.e. on the noteheads that are on the "correct" side of the stem.
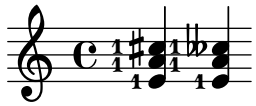
Incidentally, this also avoids collisions with accidentals.

`fingering-adjacent-note-chord.ly`
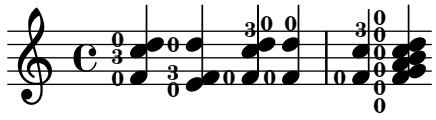


Horizontally-offset `Fingerings` align along the Y axis when they are within `FingeringColumn.snap-radius` of each other.

`fingering-column-snap-radius.ly`



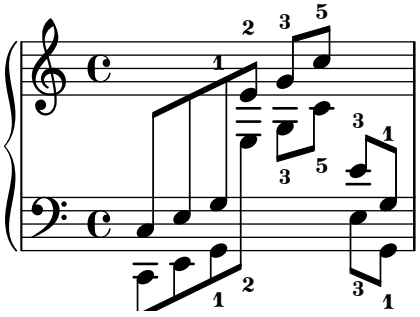
Horizontal `Fingering` grobs that collide do not intersect. Non-intersecting `Fingering` grobs are left alone. This is managed by the `FingeringColumn` grob.

`fingering-column.ly`



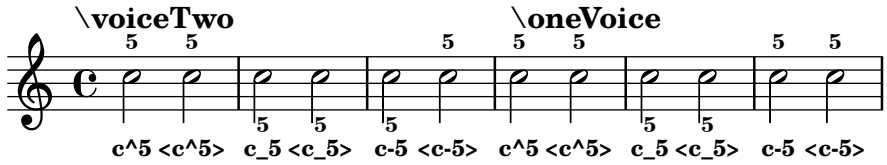
Fingerings work correctly with cross-staff beams.

`fingering-cross-staff.ly`



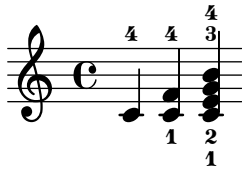
Fingering directions in directed and undirected contexts.

`fingering-directions.ly`



Automatic fingering tries to put fingering instructions next to noteheads.

`fingering.ly`
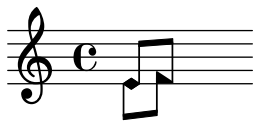


Stems reach correct begin points of merged noteheads.

`flag-stem-begin-position.ly`

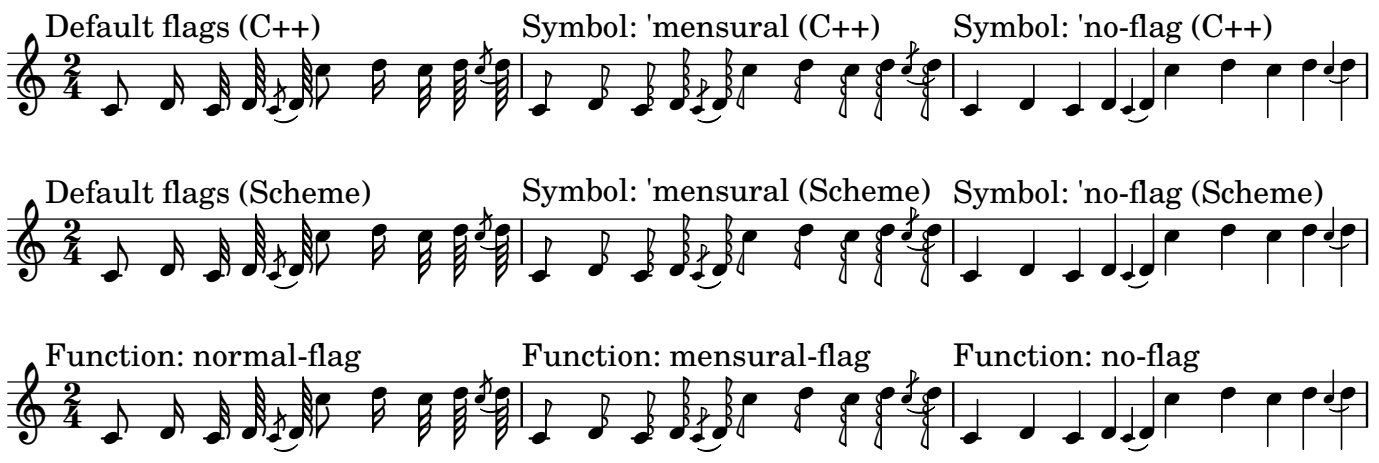






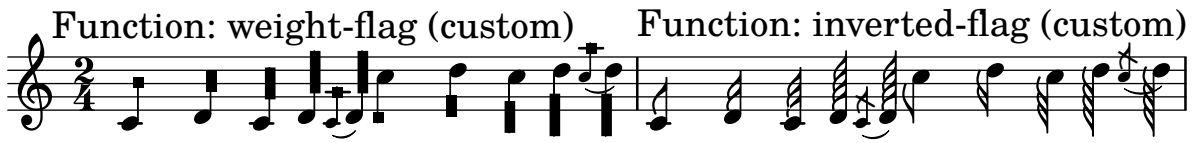
Default flag styles: '(), 'mensural and 'no-flag. Compare all three methods to print them: (1) C++ default implementation, (2) Scheme implementation using the 'style grob property and (3) setting the 'flag property explicitly to the desired Scheme function. All three systems should be absolutely identical.

`flags-default.ly`
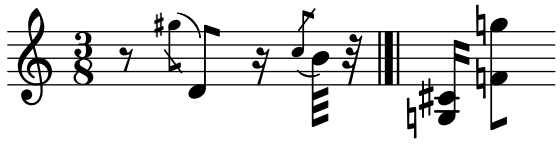


The 'stencil property of the Flag grob can be set to a custom scheme function to generate the glyph for the flag.

`flags-in-scheme.ly`
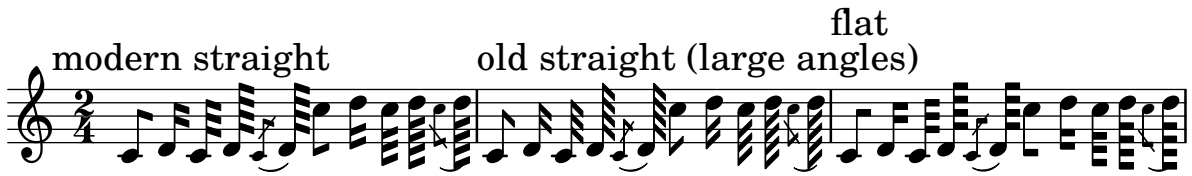


Flags can be drawn straight in the style used by Stockhausen and Boulez.

`flags-straight-stockhausen-boulez.ly`



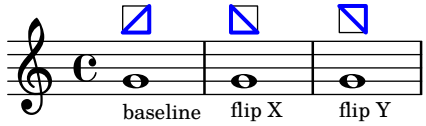
Straight flag styles.

`flags-straight.ly`



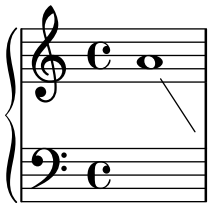
Stencils can be flipped horizontally or vertically within their bounding box using `flip-stencil`.

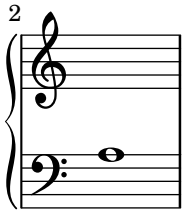
`flip-stencil.ly`



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.

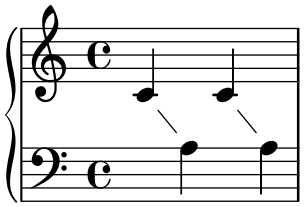
`follow-voice-break.ly`





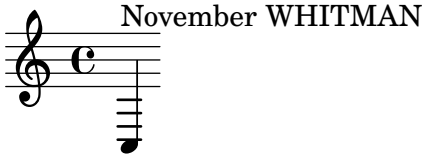
The voice follower is not confused when set for consecutive sets of staff switches.

`follow-voice-consecutive.ly`



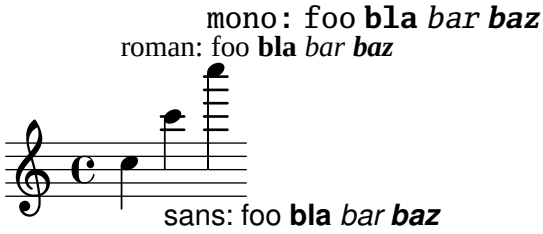
TM and No should not be changed into trademark/number symbols. This may happen with incorrect font versions.

`font-bogus-ligature.ly`



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

`font-family-override.ly`



Exercise font features. Requires a font that supports the features. This ensures no errors using the interface.

`font-features.ly`

Hello

HELLO

Hello

Hello

Hello

0123456789

0123456789

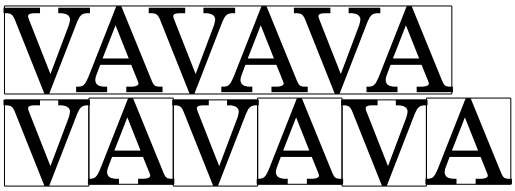
Hello 0123456789

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

`font-kern.ly`

With kerning:

Without kerning:



External fonts may be used without being installed on the operating system, by loading either a specific font file or a directory that contains font files. In this example two logos should be printed, rather than a letter glyph.

`font-name-add-files.ly`

These files and directories should have been removed:
- /tmp/dummyfont-Ivln7O-font.otf
- /tmp/dummyfont-Ivln7O-dir/font.otf
- /tmp/dummyfont-Ivln7O-dir

Setting the `font-name` property does not change the font size. The two strings below should be concatenated and have the same font size.

Note that 'the same font size' is related to what lilypond reports on the console if in verbose mode (3.865234375 units for this regression test). If you actually look at the two fonts the optical size differs enormously.

`font-name-font-size.ly`

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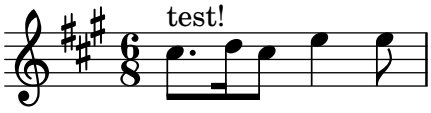
Other fonts can be used by setting `font-name` for the appropriate object. The string should be a Pango font description without size specification.

`font-name.ly`



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

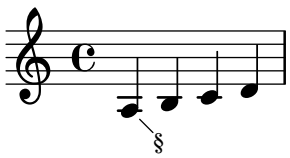
`font-postscript.ly`



This is an example of automatic footnote numbering where the number is reset on each page. It uses the symbol-footnotes numbering function, which assigns the symbols *, †, ‡, § and ¶ to successive footnotes, doubling up on the symbol after five footnotes have been reached.

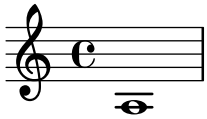
footnote-auto-numbering-page-reset.ly
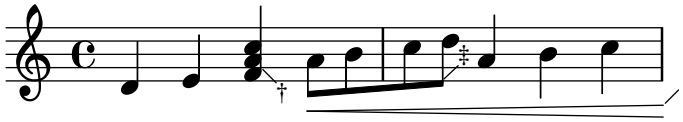
a b[*] d[†] f[‡]
h i
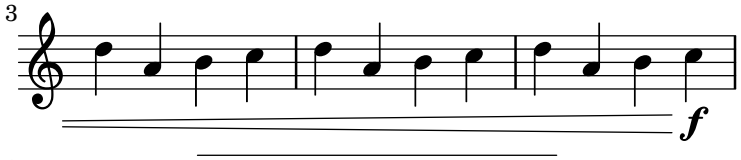


[*]**c**
[†]**e**
[‡]g
[§]j

2
k l*





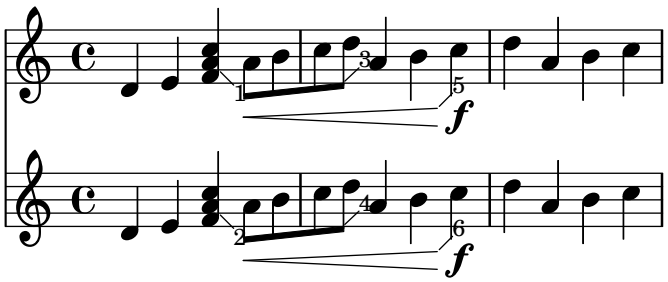
3

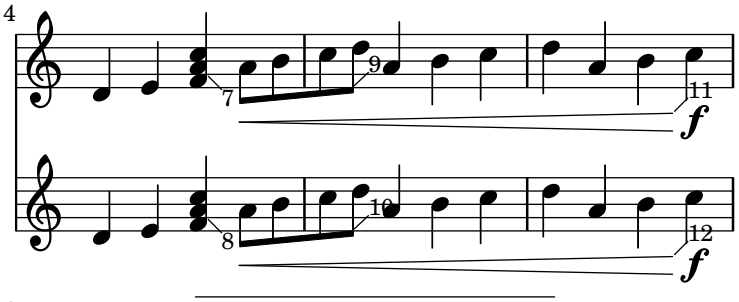


*m
†n
‡o
§p
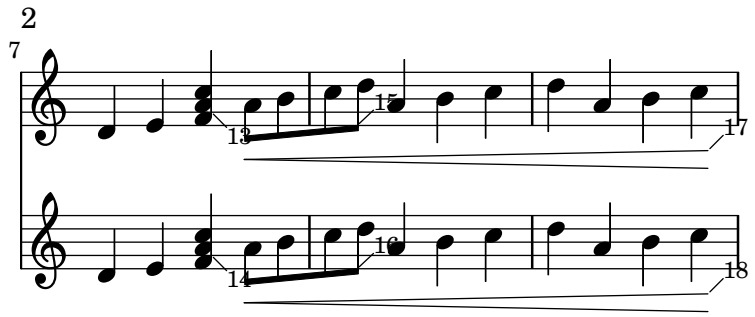
Music engraving by LilyPond 2.21.1—www.lilypond.org

footnote-auto-numbering-vertical-order.ly





[1]n
[2]n
[3]o
[4]o
[5]p
[6]p
[7]n
[8]n
[9]o
[10]o
[11]p
[12]p



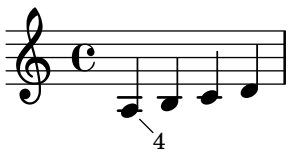
[13] n
[14] n
[15] o
[16] o
[17] p
[18] p

Music engraving by LilyPond 2.21.1—www.lilypond.org

This is an example of automatic footnote numbering where the number is not reset on each page. It uses the default numbering function, which assigns numbers starting at 1 to successive footnotes.

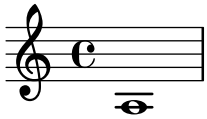
footnote-auto-numbering.ly
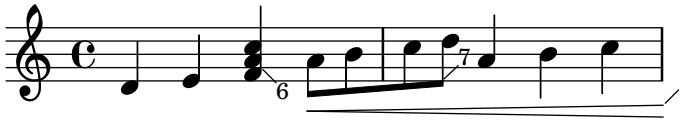
a b[1] d[2] f[3]
h i
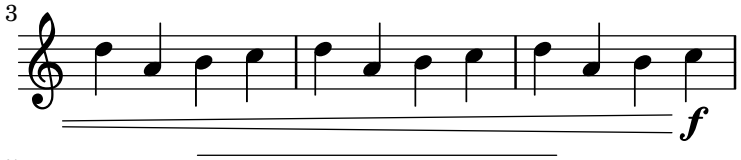


[1]c
[2]e
[3]g
[4]j

2
k l[5]





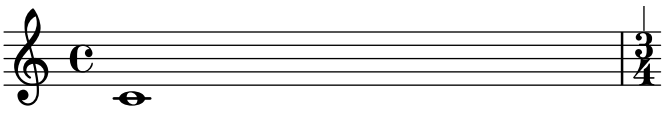


[5]m
[6]n
[7]o
[8]p

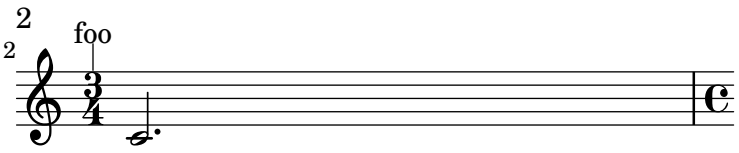
Music engraving by LilyPond 2.21.1—www.lilypond.org

With grobs that have break visibility, footnotes will automatically take the break visibility of the grob being footnoted. This behavior can be overridden.

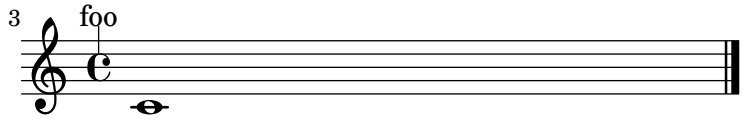
footnote-break-visibility.ly



bar



bar

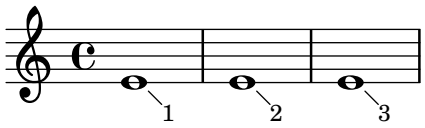


bar

Music engraving by LilyPond 2.21.1—www.lilypond.org

The padding between a footnote and the footer can be tweaked.

footnote-footer-padding.ly
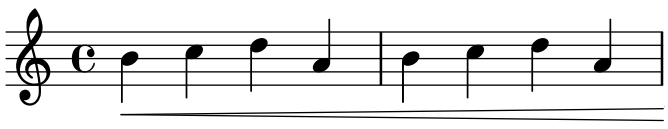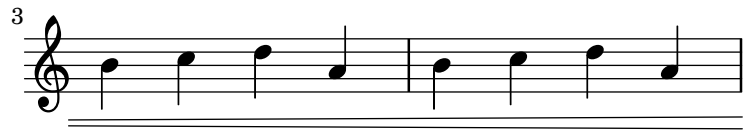


1. Tiny space below.
2. Tiny space below.
3. Big space below.
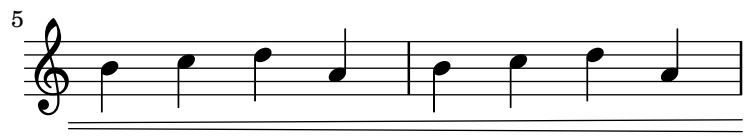
Music engraving by LilyPond 2.21.1—www.lilypond.org

Footnotes are annotated at the correct place, and the annotation goes to the correct page.
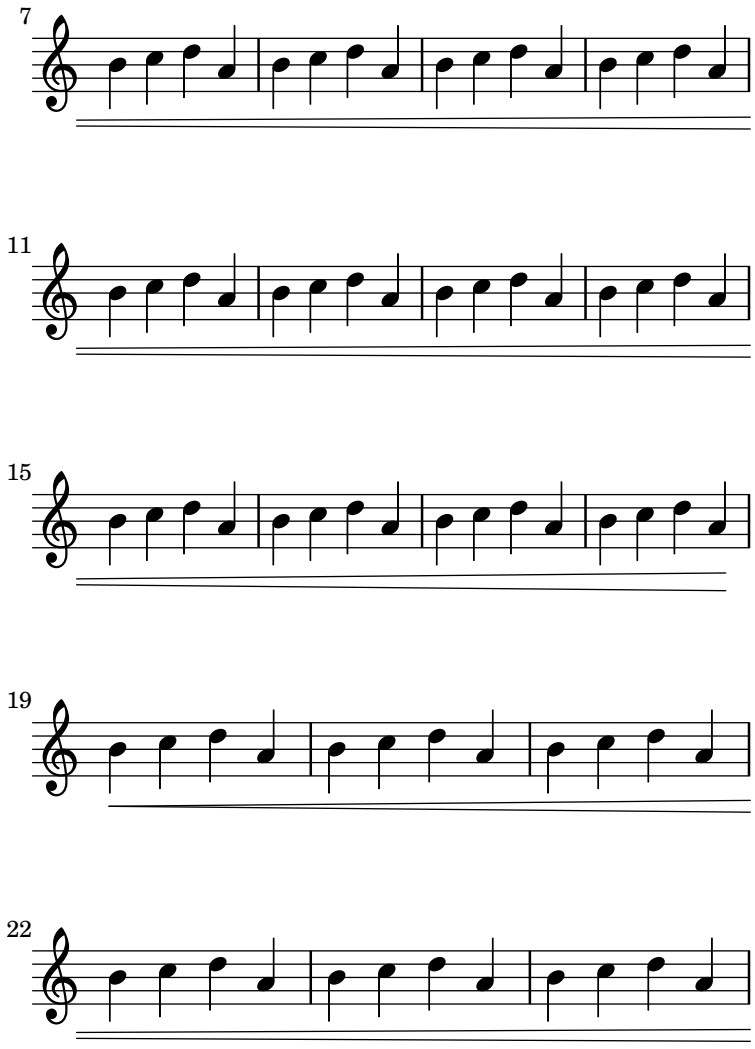
footnote-spanner.ly

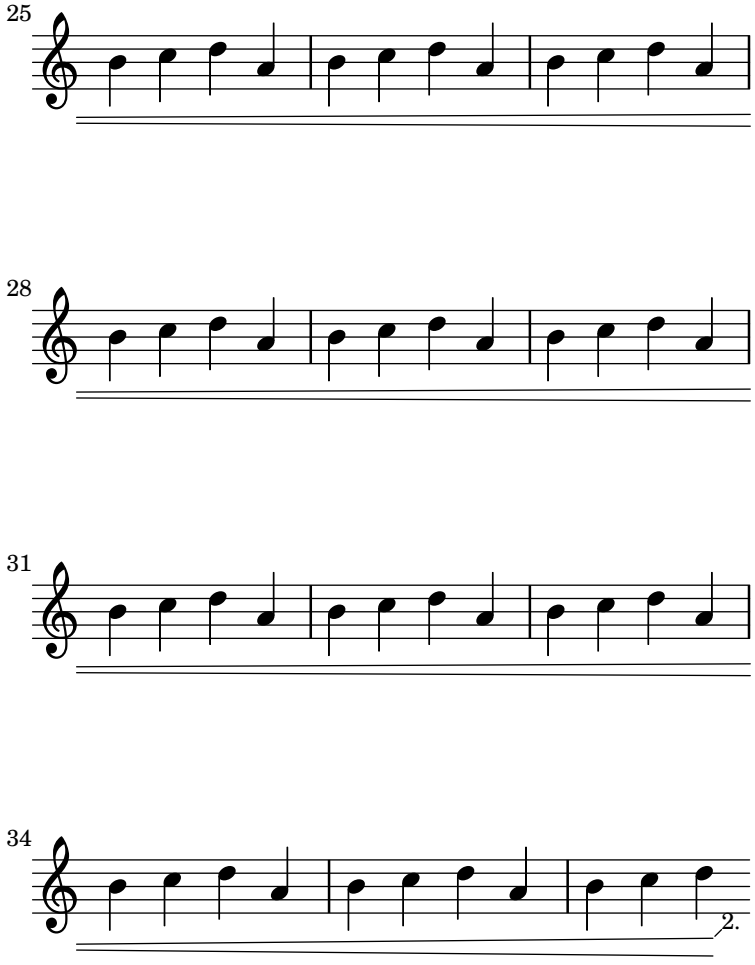






1. Goes to the first broken spanner.


7
11
15
19
22



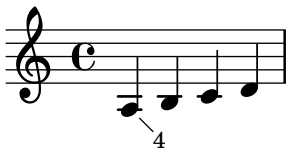
2. Goes to the last broken spanner.

Music engraving by LilyPond 2.21.1—www.lilypond.org

Lilypond does footnotes.

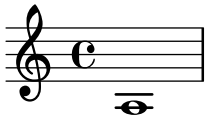
footnote.ly

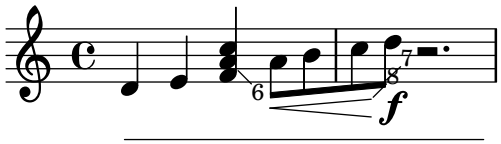
a b[1] d[2] f[3]
h i



1. c
2. e
3. g
4. j

2
k l[5]



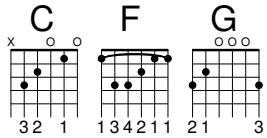


5. m
6. n
7. o
8. p

Music engraving by LilyPond 2.21.1—www.lilypond.org

FretBoards should be aligned in the Y direction at the fret-zero, string 1 intersection.

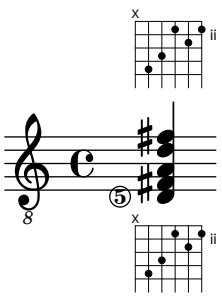
`fret-board-alignment.ly`



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

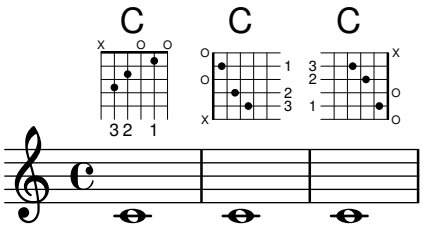
`fret-boards.ly`

autofrets



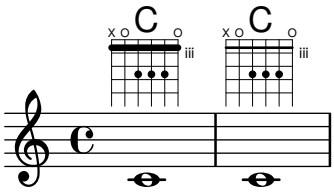
Fret diagrams of different orientation should share a common origin of the topmost fret or string.

`fret-diagram-origins.ly`



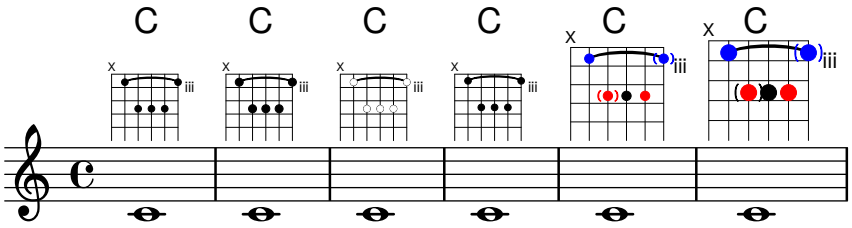
A capo indicator can be added with a fret-diagram-verbose string, and its thickness can be changed.

`fret-diagrams-capo.ly`



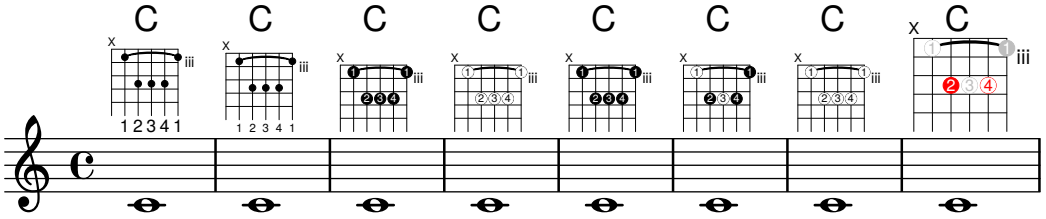
Dots indicating fingerings can be changed in location, size, and coloring. It is possible to parenthesize a single dot. The color of the paranthesis may be taken from dot or default. A possible collision between parathesis and fret-label- indication can be resolved by an override for `fret-label-horizontal-offset` in `fret-diagram-details`.

`fret-diagrams-dots.ly`



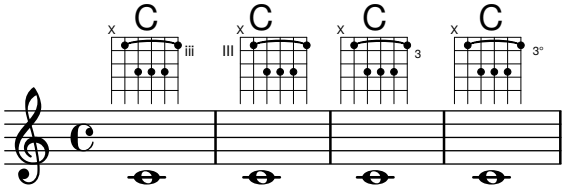
Finger labels can be added, either in dots or below strings. Dot color can be changed globally or on a per-dot basis, and fingering label font size can be adjusted.

`fret-diagrams-fingering.ly`



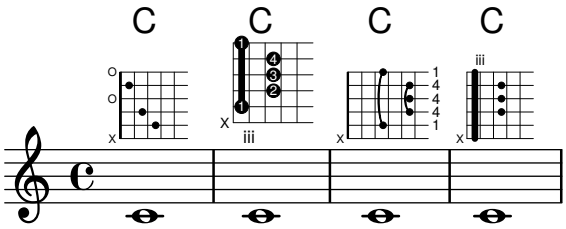
The label for the lowest fret can be changed in location, size, and number type.

`fret-diagrams-fret-label.ly`



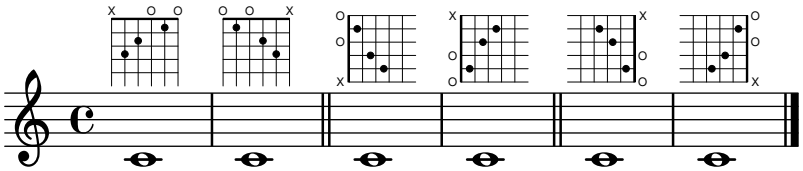
Fret diagrams can be presented in landscape mode.

`fret-diagrams-landscape.ly`



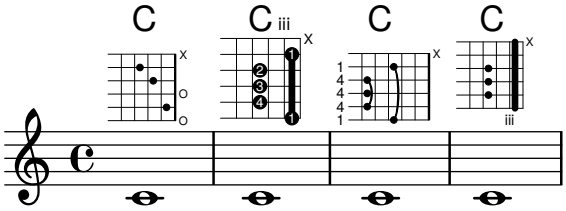
Fret-diagrams may be printed left-handed

`fret-diagrams-left-handed.ly`



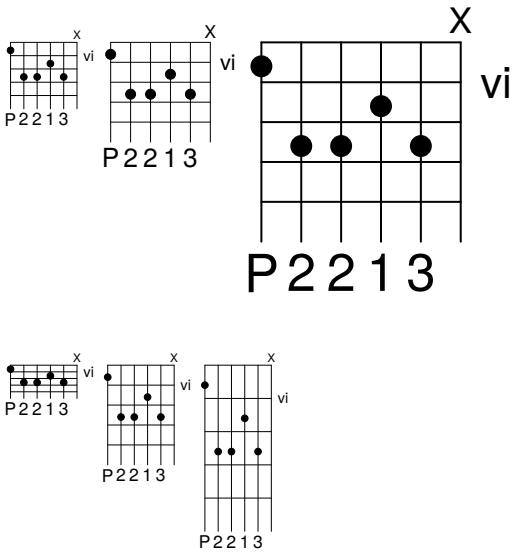
Fret diagrams can be presented in landscape mode.

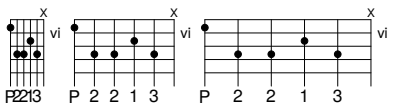
`fret-diagrams-opposing-landscape.ly`



Fret diagrams can be scaled using the `size` property. Also, scaling the distance between the frets and/or strings is possible with the properties `fret-distance` and/or `string-distance` of `fret-diagram-details`. The position and size of first fret label, mute/open signs, fingers, relative to the diagram grid, shall be the same in all cases.

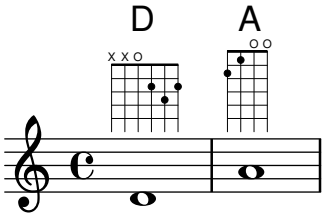
`fret-diagrams-size.ly`





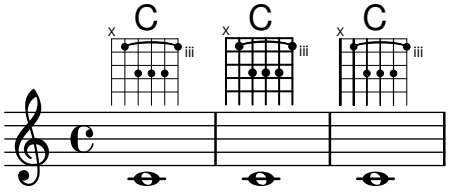
Number of frets and number of strings can be changed from the defaults.

`fret-diagrams-string-frets.ly`



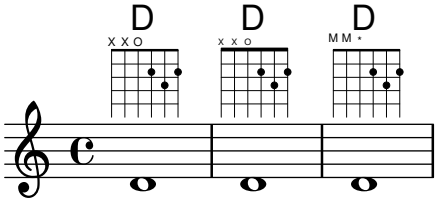
String thickness can be changed, and diagrams can have variable string thickness.

`fret-diagrams-string-thickness.ly`



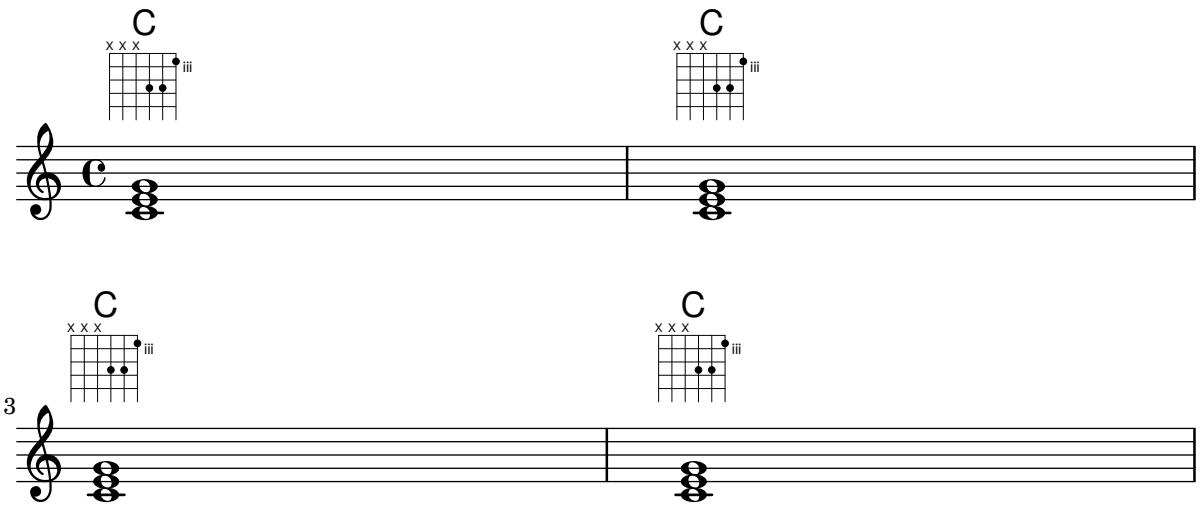
The size, spacing, and symbols used to indicate open and muted strings can be changed.

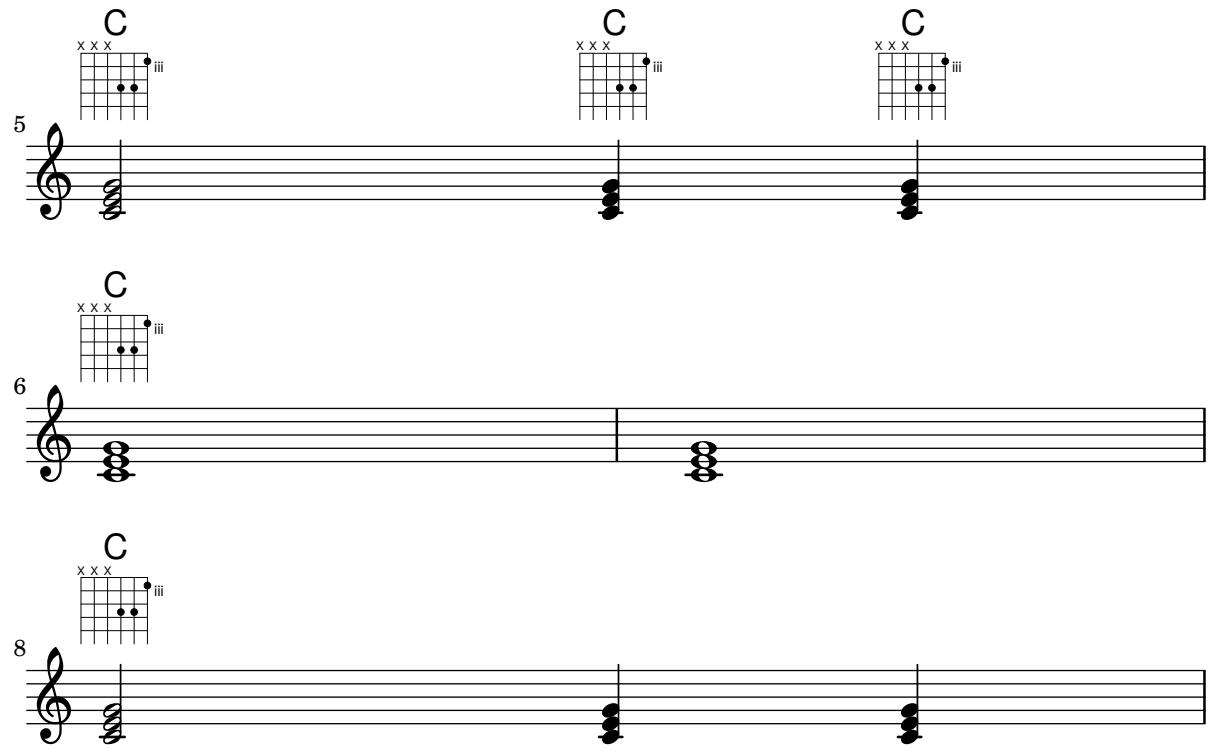
`fret-diagrams-xo-label.ly`



FretBoards can be set to display only when the chord changes or at the beginning of a new line.

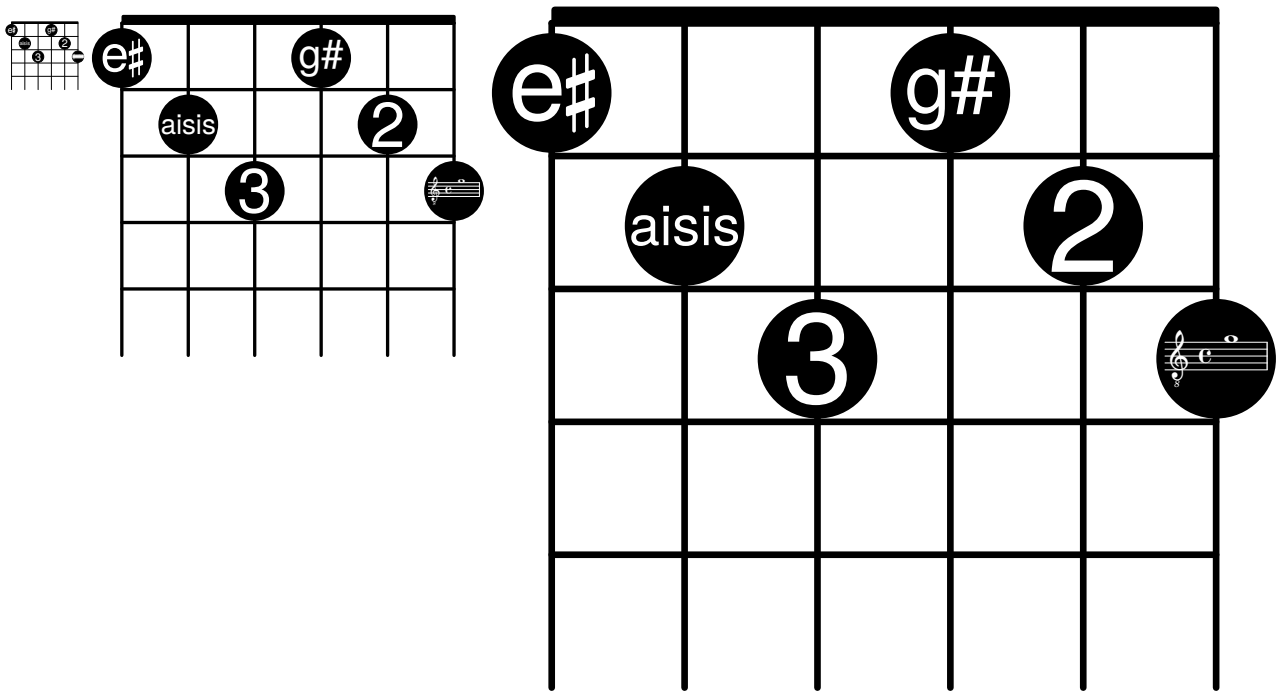
`fretboard-chordchanges.ly`





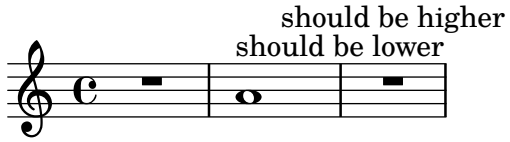
Markups can be put into the dots of a fret-diagram. Those markups are scaled automatically to fit into the dots.

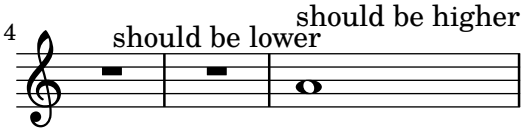
`fretdiagram-markup-in-dots.ly`

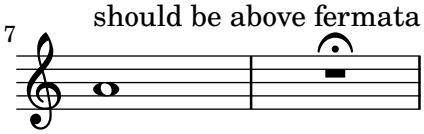


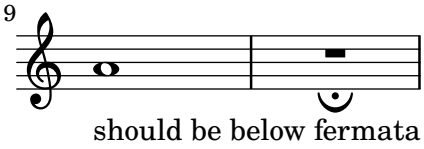
Fermata over full-measure rests should invert when below and be closer to the staff than other articulations.

`full-measure-rest-fermata.ly`
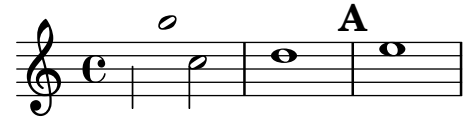








This file tests various Scheme utility functions.

`general-scheme-bindings.ly`

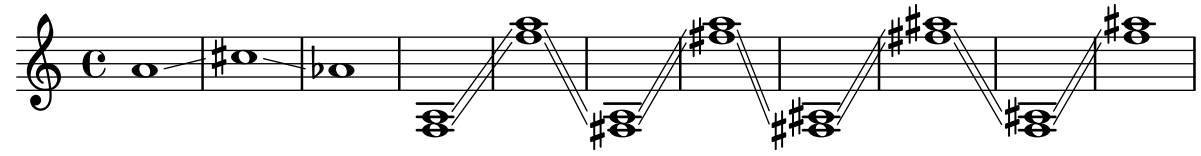
As a last resort, the placement of grobs can be adjusted manually, by setting the `extra-offset` of a grob.

`generic-output-property.ly`
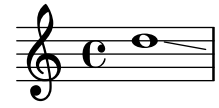


Glissandi stop before hitting accidentals. Chord glissandi stop at the same horizontal position and have the same slope, they do not cross.
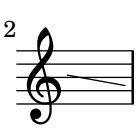
`glissando-accidental.ly`
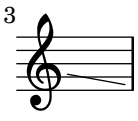


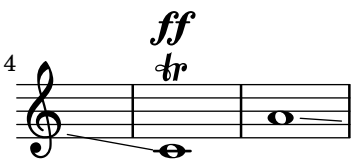
When broken, glissandi can span multiple lines.
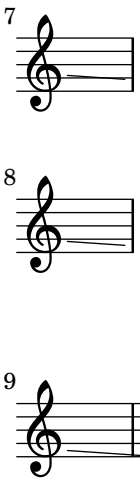
`glissando-broken-multiple.ly`





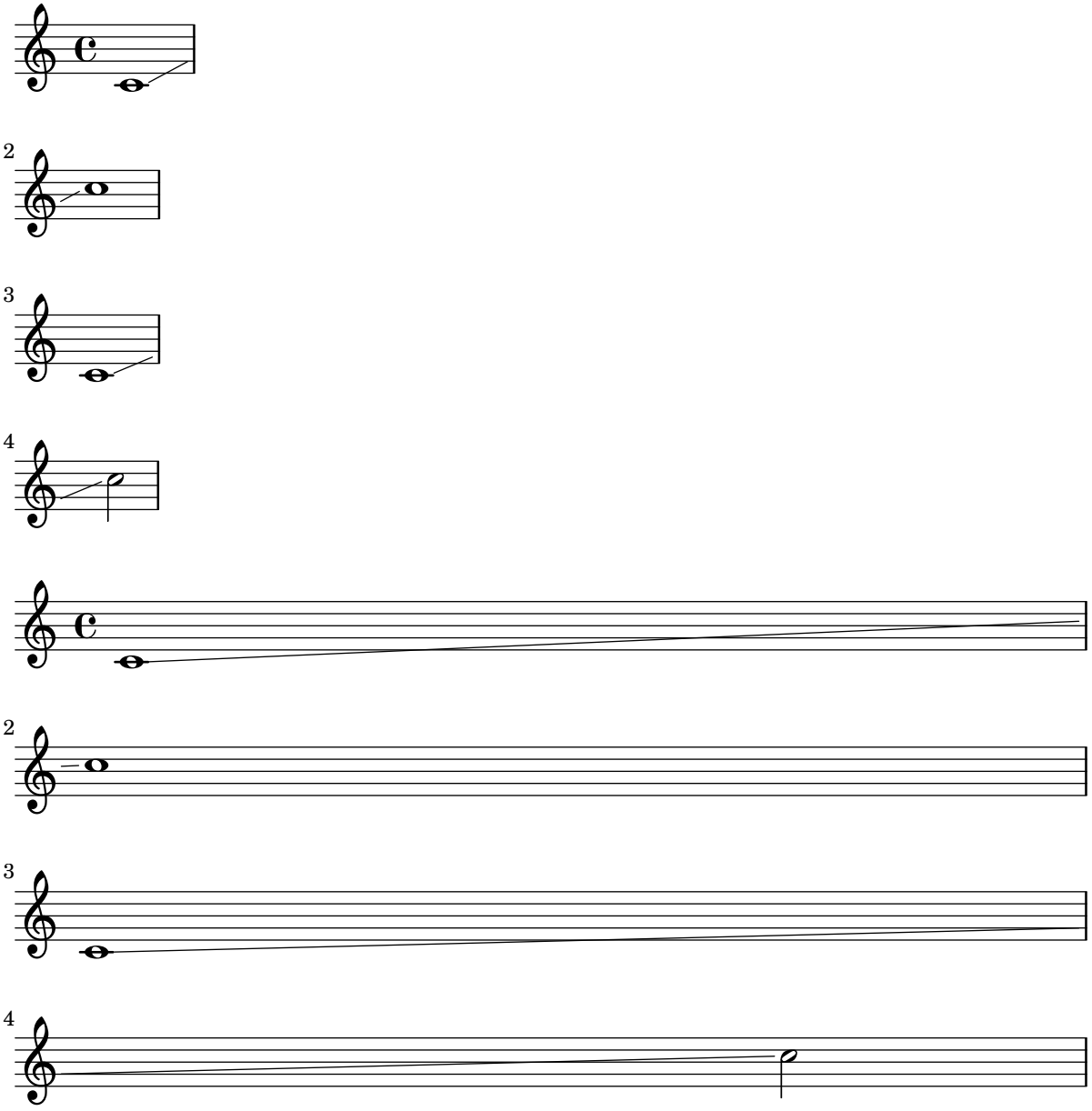






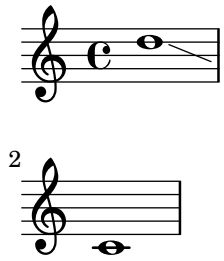
Broken glissandi anticipate the pitch on the next line.

`glissando-broken-unkilled.ly`



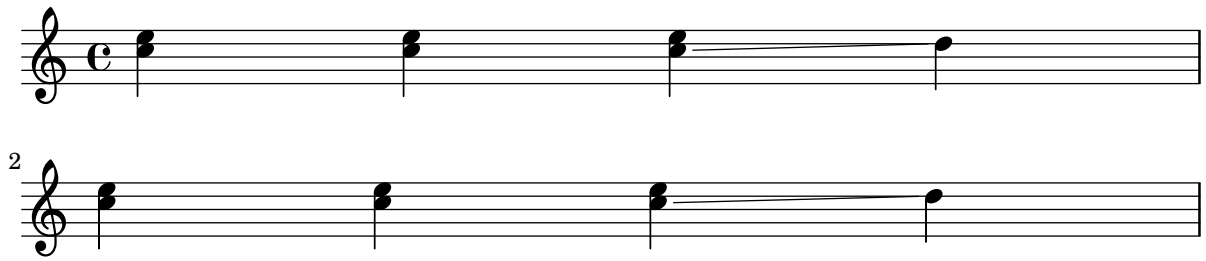
If broken, Glissandi anticipate on the pitch of the next line.

glissando-broken.ly



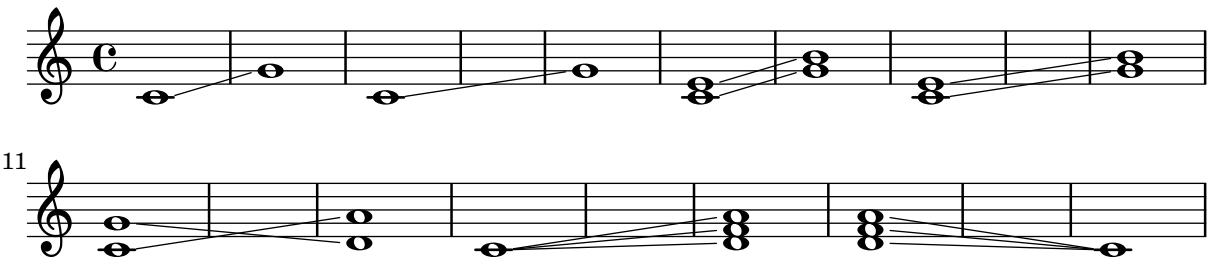
A glissando between chords should not interfere with line breaks. In this case, the music should be in two lines and there should be no warning messages issued. Also, the glissando should be printed.

glissando-chord-linebreak.ly



LilyPond typesets glissandi between chords.

glissando-chord.ly



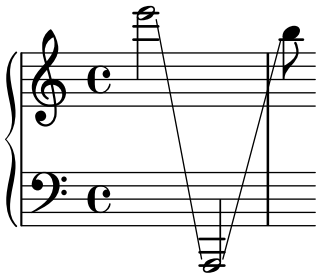
Lilypond prints consecutive glissandi.

glissando-consecutive.ly
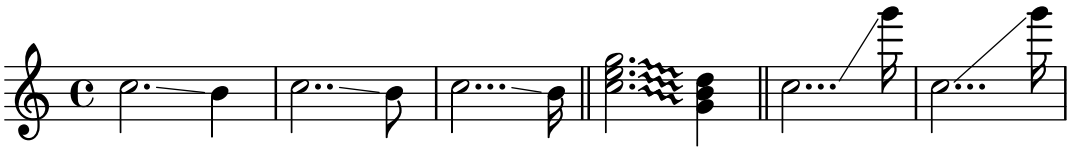


Cross staff glissandi reach their endpoints correctly.

glissando-cross-staff.ly



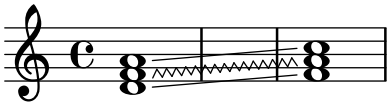
Glissandi begin after dots by default. This behavior may be changed by overriding the `start-at-dot` property.

`glissando-dots.ly`



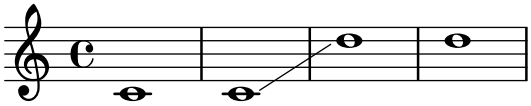
Individual glissandi within a chord can be tweaked.

`glissando-index.ly`
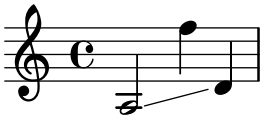


Glissandi are not broken. Here a `\break` is ineffective. Use `breakable` grob property to override.

`glissando-no-break.ly`



`NoteColumn` grobs can be skipped over by glissandi.
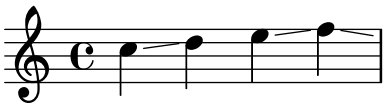
`glissando-skip.ly`
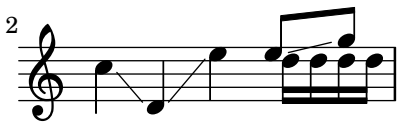


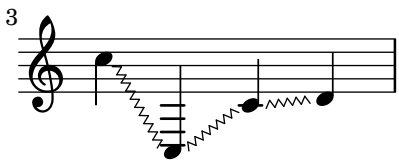
Between notes, there may be simple glissando lines. Here, the first two glissandi are not consecutive.

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

`glissando.ly`

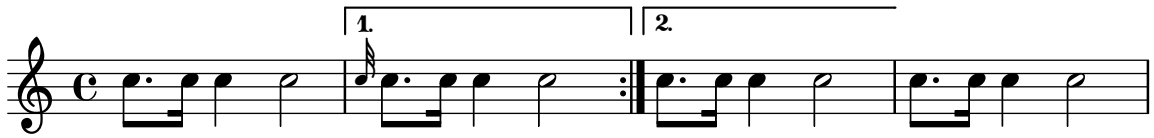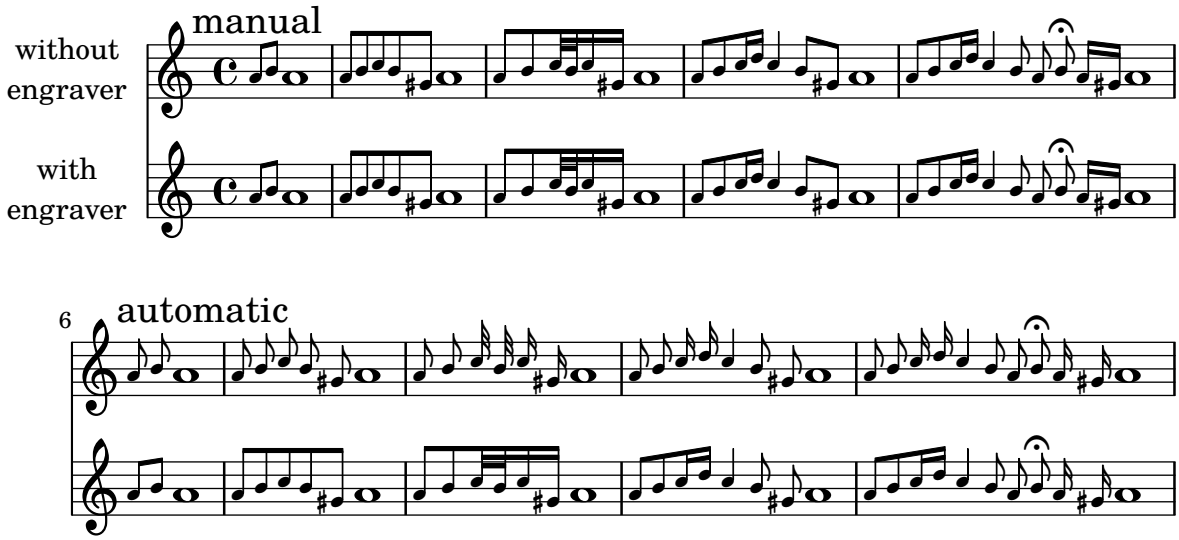






A grace in the first alternative does not cause the beaming to go awry in subsequent material

`grace-alternative.ly`
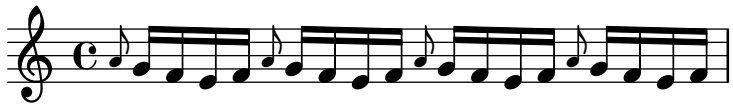


A separate '`Grace_auto_beam_engraver`' initiates autobeaming at the start of each `\grace` command.

`grace-auto-beam-engraver.ly`



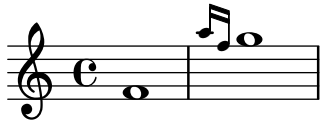
The autobeamer is not confused by grace notes.

`grace-auto-beam.ly`
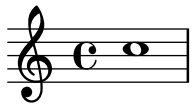


Bar line should come before the grace note.
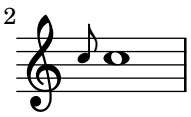
`grace-bar-line.ly`



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.

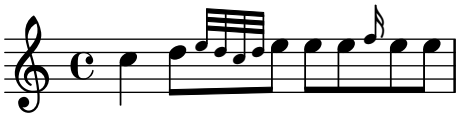
`grace-bar-number.ly`
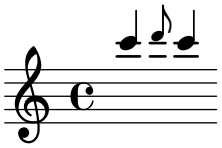




Grace beams and normal beams may occur simultaneously. Unbeamed grace notes are not put into normal beams.

`grace-beam.ly`
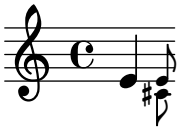


The `\voiceOne` setting is retained after finishing the grace section.

`grace-direction-polyphony.ly`
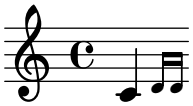


Grace notes at the end of an expression don't cause crashes.

`grace-end-2.ly`
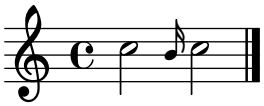


Grace notes after the last note do not confuse the timing code.

`grace-end.ly`
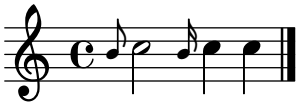


Grace code should not be confused by nested sequential music containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.

`grace-nest1.ly`
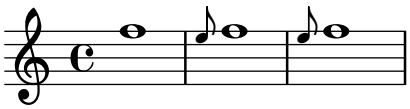


Grace code should not be confused by nested sequential music containing grace notes; practically speaking, this means that the end-bar and measure bar coincide in this example.

`grace-nest2.ly`
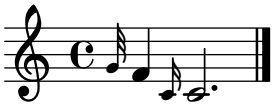


In nested syntax, graces are still properly handled.

grace-nest3.ly
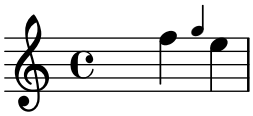


Also in the nested syntax here, grace notes appear rightly.

grace-nest4.ly
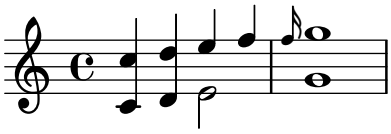


Graces notes may have the same duration as the main note.

grace-nest5.ly
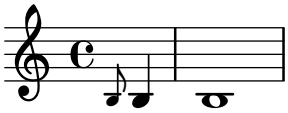


Grace notes may be put in a `partCombine`r.

grace-part-combine.ly
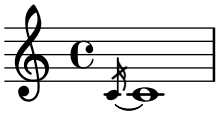


A `\partial` may be combined with a `\grace`.

grace-partial.ly



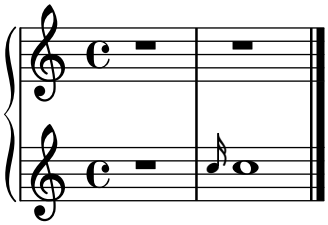
Create grace notes with slashed stem, but no slur. That can be used when the grace note is tied to the next note.

grace-slashed-no-slur.ly
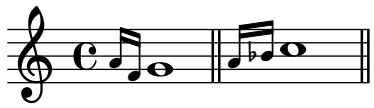


Stripped version of trip.ly. Staves should be of correct length.

grace-staff-length.ly
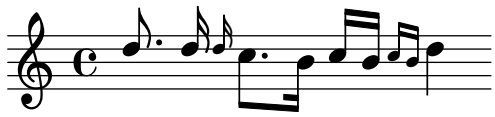


Pieces may begin with grace notes.

`grace-start.ly`
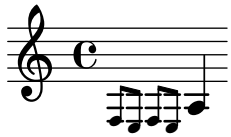


Stem lengths for grace notes should be shorter than normal notes, if possible. They should never be longer, even if that would lead to beam quanting problems.

`grace-stem-length.ly`
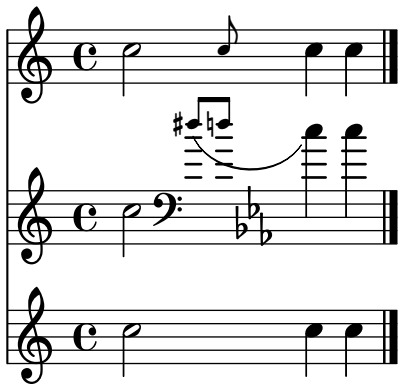


Here `startGraceMusic` should set `no-stem-extend` to true; the two grace beams should be the same here.

`grace-stems.ly`
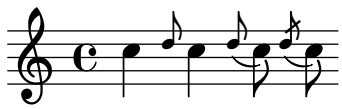


Grace notes in different voices/staves are synchronized.

`grace-sync.ly`



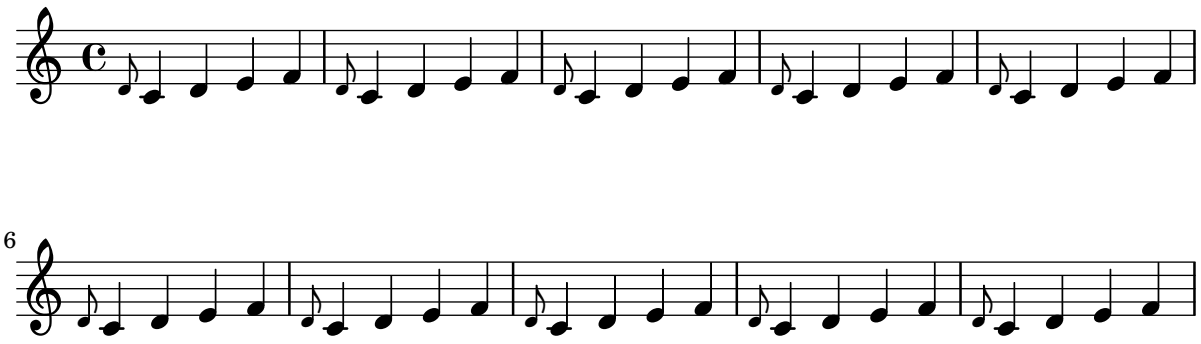
There are three different kinds of grace types: the base grace switches to smaller type, the appoggiatura inserts also a slur, and the acciaccatura inserts a slur and slashes the stem.

`grace-types.ly`



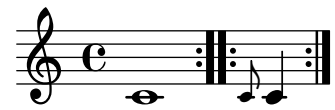
When grace notes are entered with unfolded repeats, line breaks take place before grace notes.

`grace-unfold-repeat.ly`
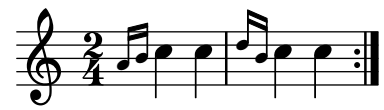


A volta repeat may begin with a grace. Consecutive ending and starting repeat bars are merged into one `:..:`.

`grace-volta-repeat-2.ly`



Repeated music can start with grace notes. Bar checks preceding the grace notes do not cause synchronization effects.
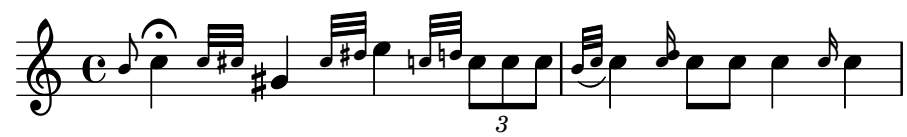
`grace-volta-repeat.ly`



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.

`grace.ly`



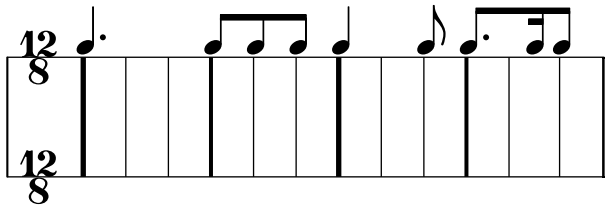
The graphviz feature draws dependency graphs for grob properties.

graphviz.ly
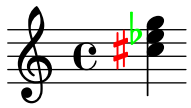


With grid lines, vertical lines can be drawn between staves synchronized with the notes.

`grid-lines.ly`



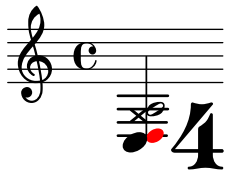
With the full form of the `\tweak` function, individual grobs that are indirectly caused by events may be tuned.

`grob-indirect-tweak.ly`



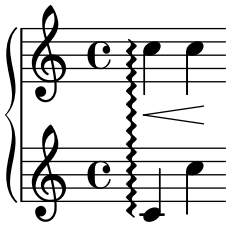
With the `\tweak` function, individual grobs that are directly caused by events may be tuned directly.

`grob-tweak.ly`



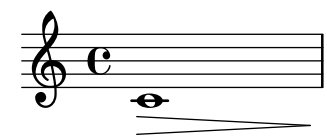
Hairpins in `Dynamics` contexts do not collide with arpeggios.

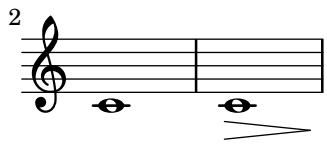
`hairpin-arpeggio.ly`
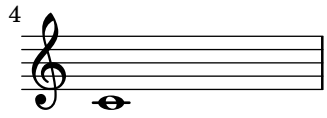


If a hairpin ends on the first note of a new staff, we do not print that ending. But on the previous line, this hairpin should not be left open, and should end at the bar line.

`hairpin-barline-break.ly`

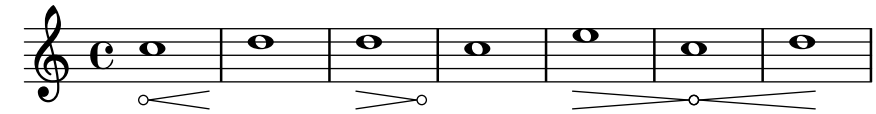




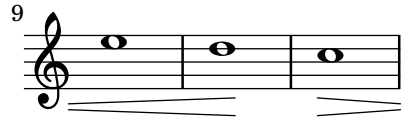


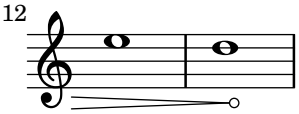
Hairpins can have circled tips. A decrescendo del niente followed by a crescendo al niente should only print one circle.

`hairpin-circled.ly`



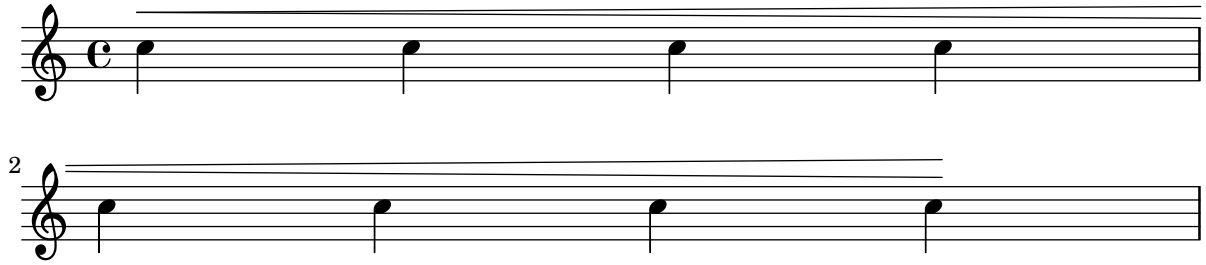






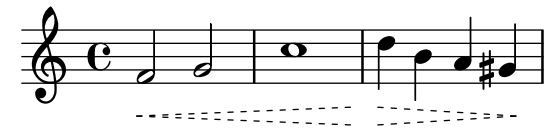
Broken hairpins are not printed too high after treble clefs.

`hairpin-clef.ly`



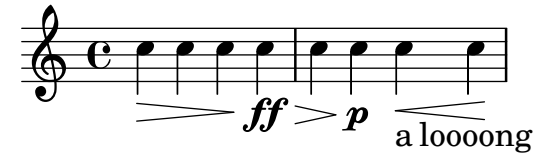
Hairpin crescendi may be dashed.

`hairpin-dashed.ly`



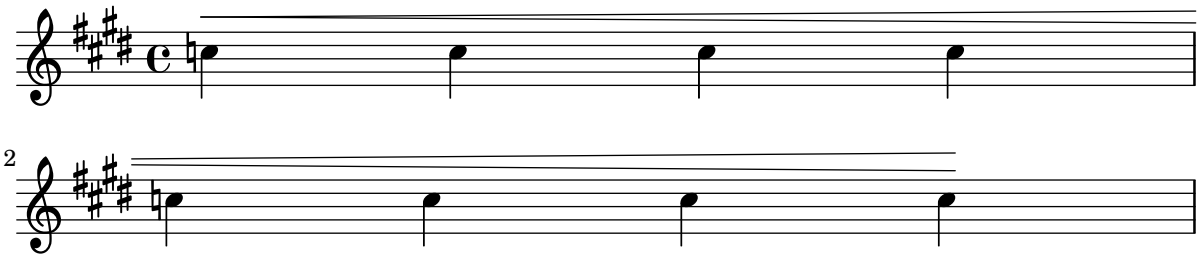
Hairpin dynamics start under notes if there are no text-dynamics. If there are text dynamics, the hairpin does not run into them.

`hairpin-ending.ly`



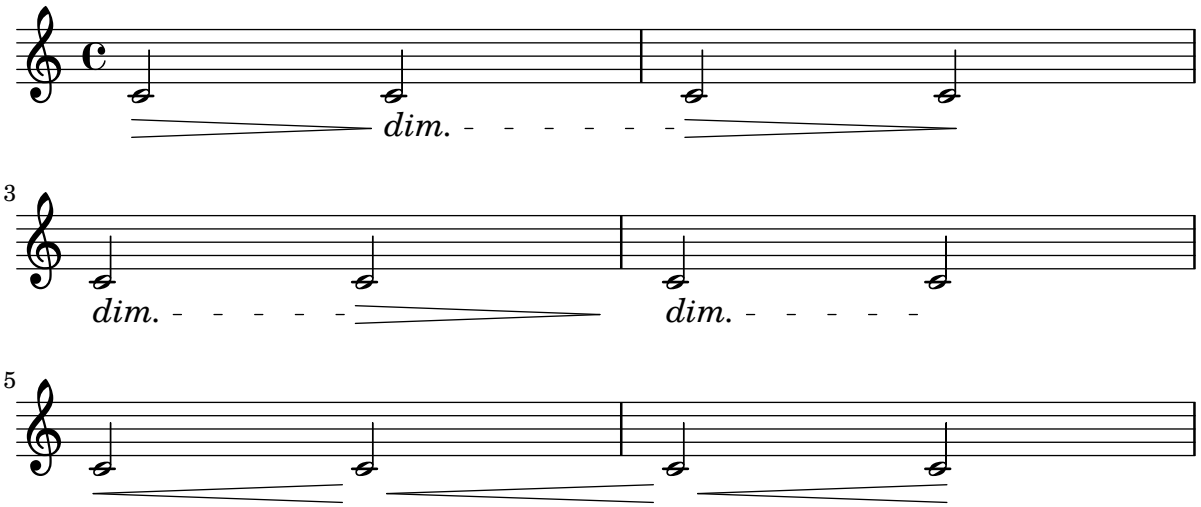
Broken hairpins are not printed too high after key signatures.

`hairpin-key-signature.ly`



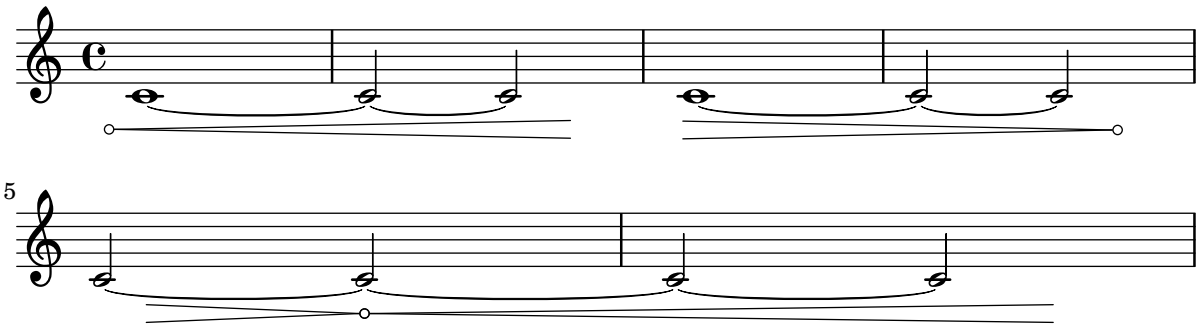
Bound padding for hairpins also applies before following `DynamicTextSpanner` grobs. In this case, `bound-padding` is not scaled down.

`hairpin-neighboring-span-dynamics.ly`
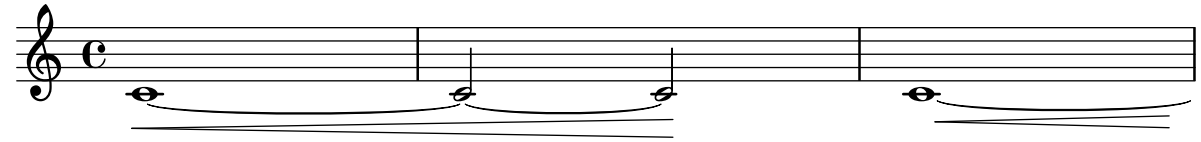


The `shorten-pair` property works with circled-tip hairpins. When two hairpins share a circle, the adjoining ends are not moved.

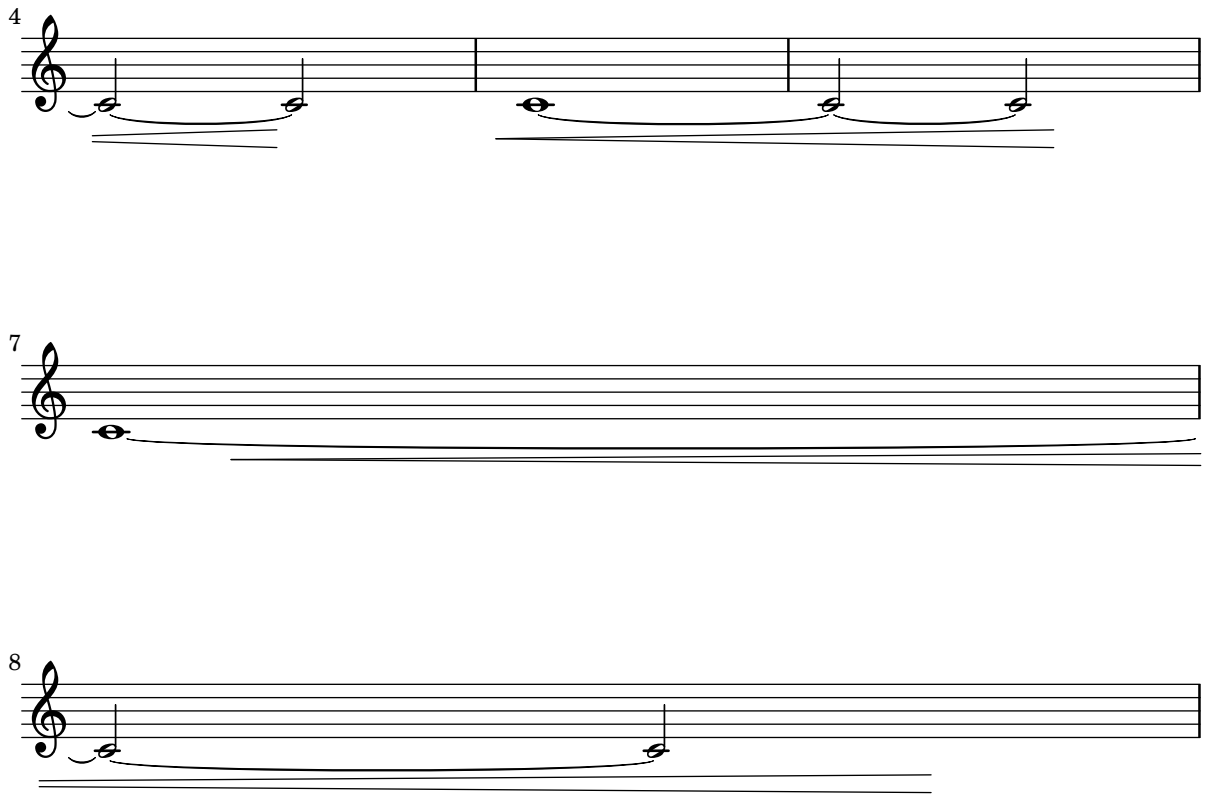
`hairpin-shorten-pair-circled-tip.ly`



The ends of hairpins may be offset with the `shorten-pair` property. Positive values offset ends to the right, negative values to the left.

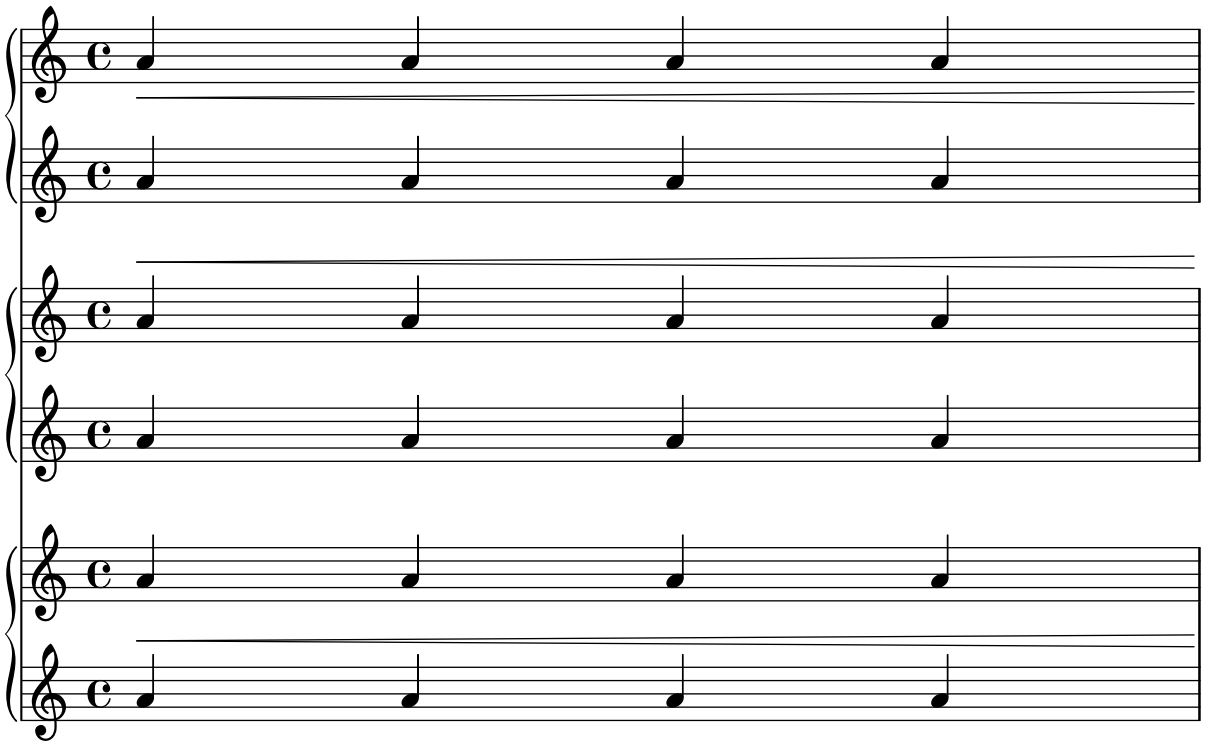
`hairpin-shorten-pair.ly`
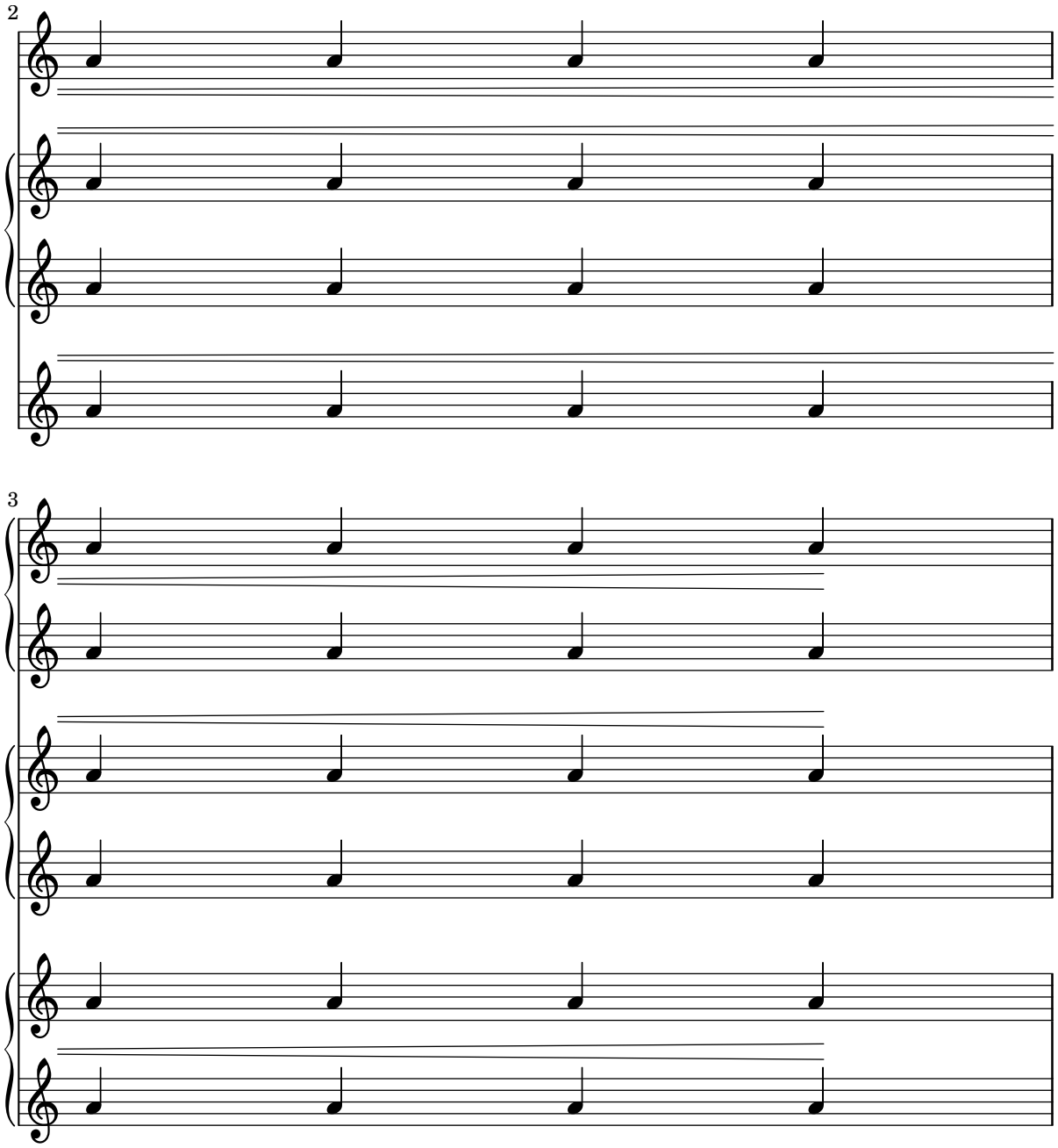




`Hairpin` grobs do not collide with `SpanBar` grobs. `Hairpin` grobs should, however, go to the end of a line when the `SpanBar` is not present.
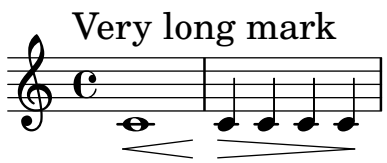
`hairpin-span-bar.ly`
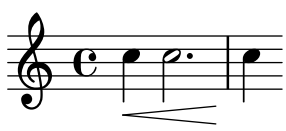




'to-barline is not confused by very long marks.

`hairpin-to-barline-mark.ly`



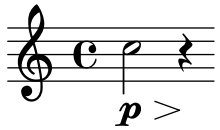
Hairpins whose end note is preceded by a bar line should end at that bar line.

`hairpin-to-barline.ly`



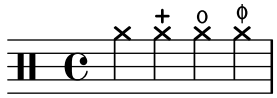
Hairpins end at the left edge of a rest.

`hairpin-to-rest.ly`



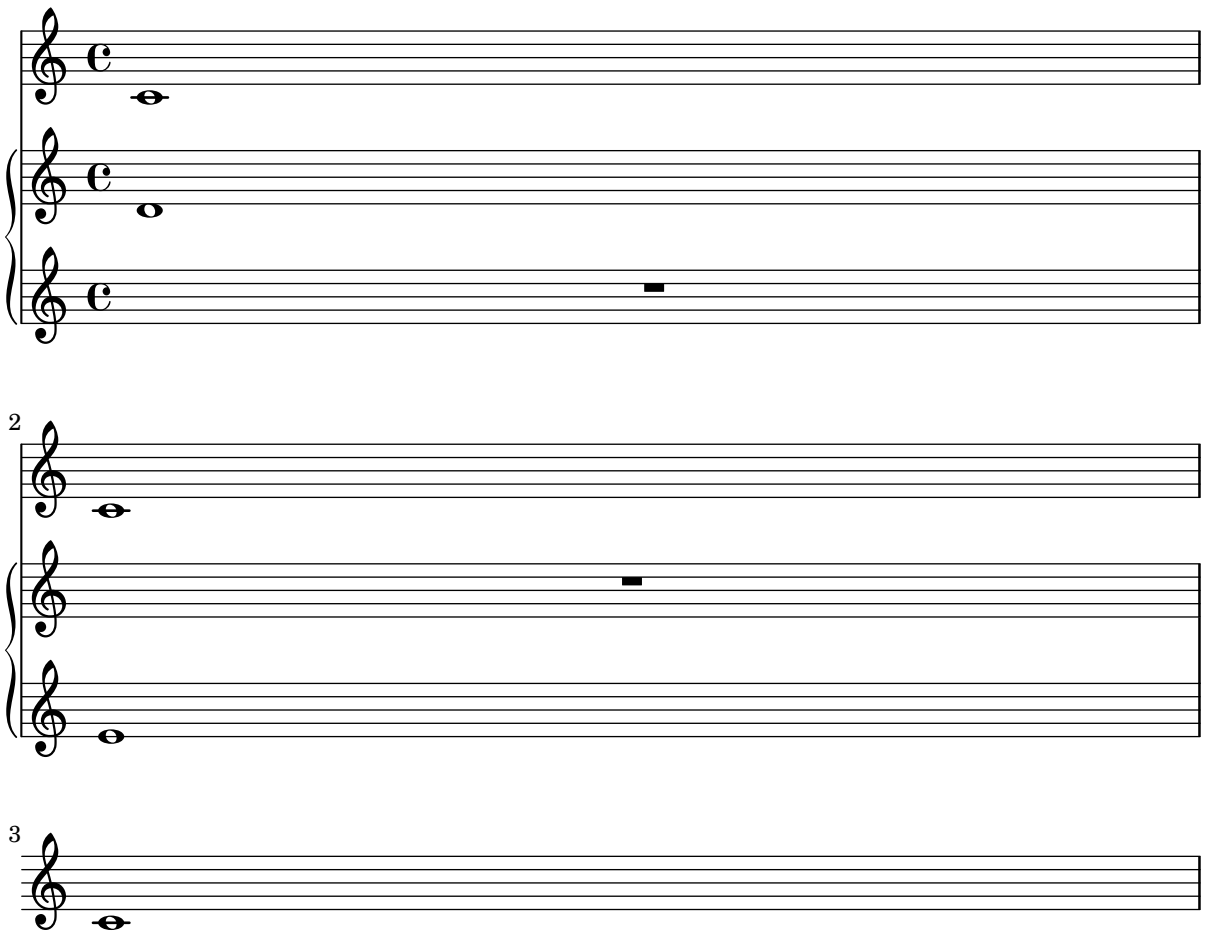
The halfopenvertical articulation is available.

`halfopenvertical.ly`



Staves in a PianoStaff remain alive as long as any of the staves has something interesting.

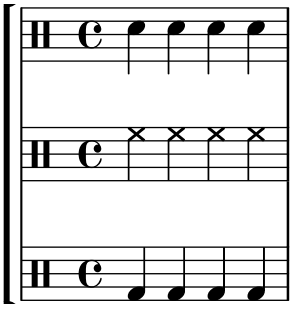
`hara-kiri-alive-with.ly`



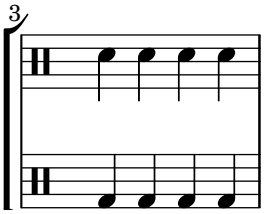
Hara-kiri staves are suppressed if they are empty. This example really contains three drum staves, but as it progresses, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered 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.
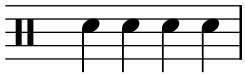
`hara-kiri-drumstaff.ly`



2
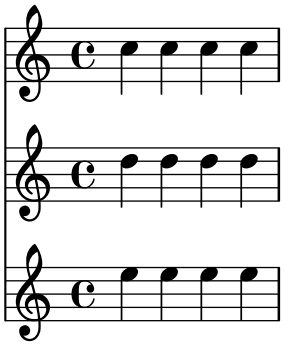


4

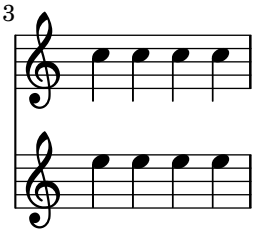


Inserting the harakiri settings globally into the Staff context should not erase previous settings to the Staff context.
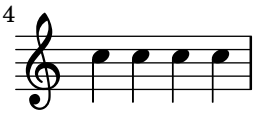
`hara-kiri-keep-previous-settings.ly`

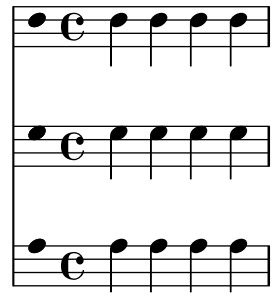


2



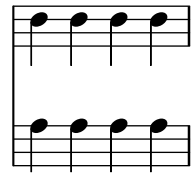
4
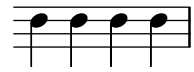




2

3

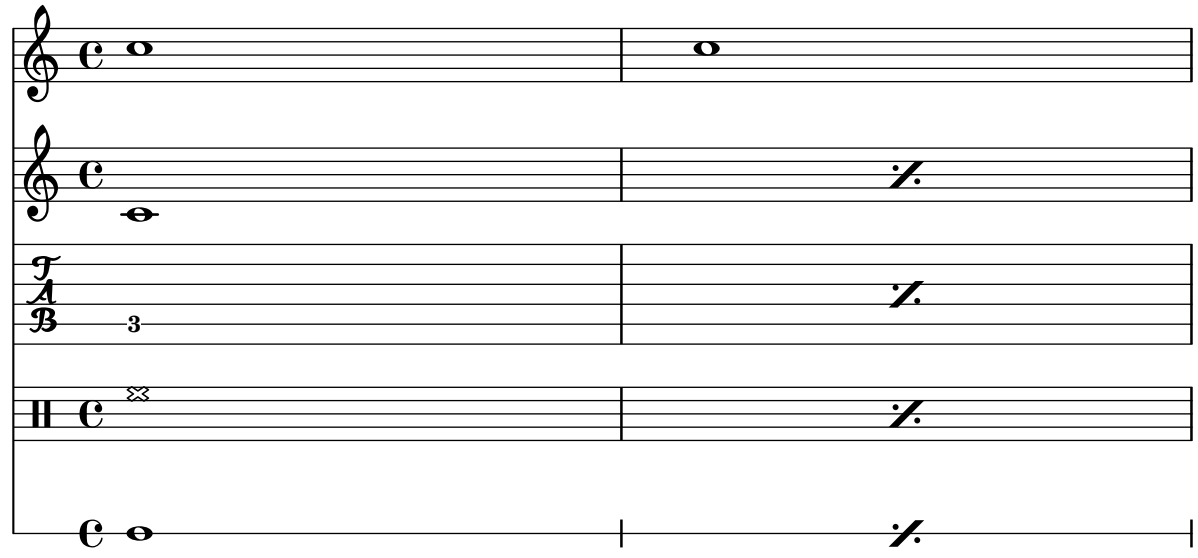


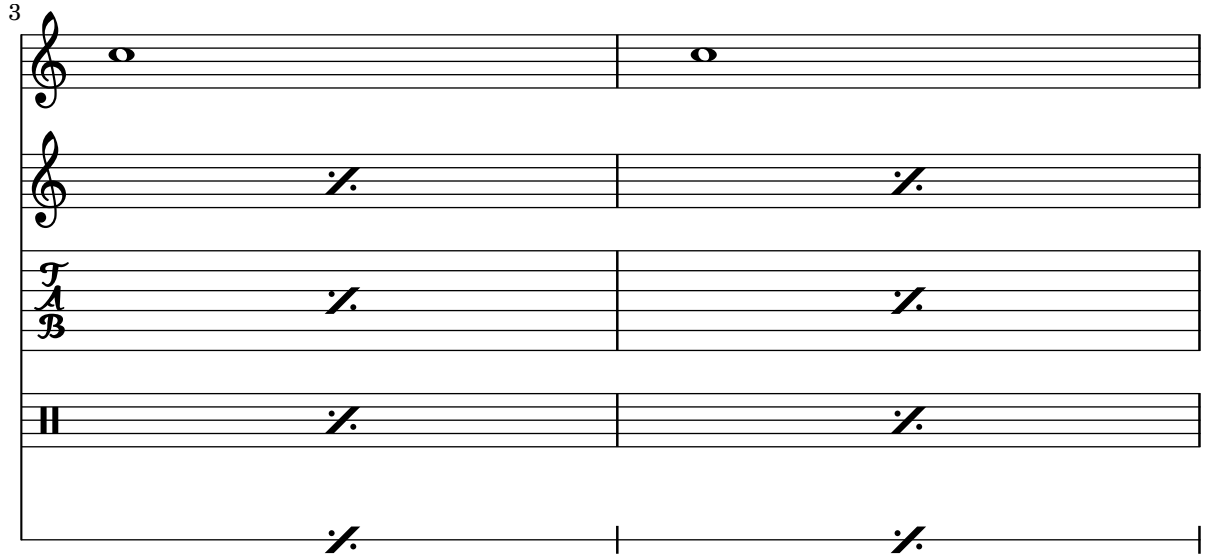
4



Staves, RhythmicStaves, TabStaves and DrumStaves with percent repeats are not suppressed.

`hara-kiri-percent-repeat.ly`

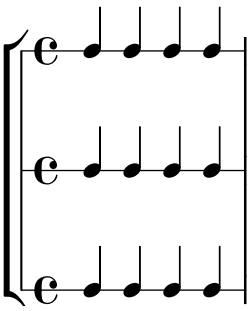


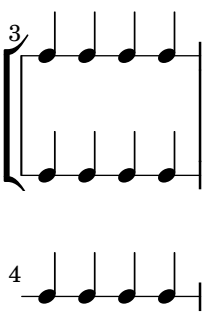


Hara-kiri staves are suppressed if they are empty. This example really contains three rhythmic staves, but as it progresses, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered 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.

`hara-kiri-rhythmicstaff.ly`

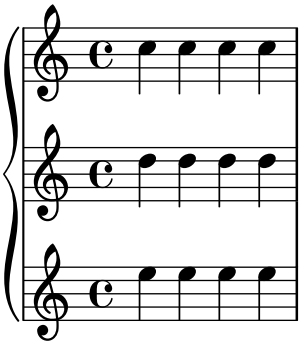




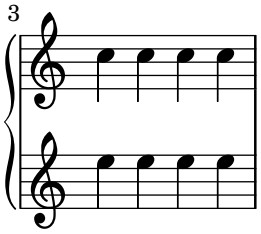
Hara-kiri staves kill themselves if they are empty. This example really contains three staves, but as they progress, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered a bug, however, the scenario of all staves disappearing does not happen in practice.)
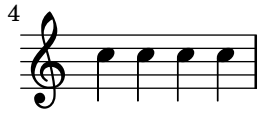
Any staff brackets and braces are removed, both in the single staff and no staff case.

`hara-kiri-staff.ly`
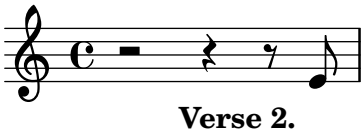


2
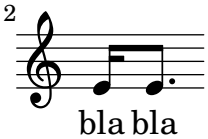




stanza numbers remain, even on otherwise empty lyrics lines.

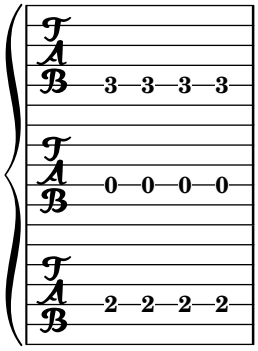
`hara-kiri-stanza-number.ly`





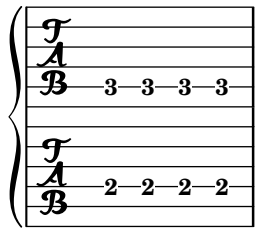
Hara-kiri staves are suppressed if they are empty. This example really contains three tab staves, but as it progresses, empty ones are removed: this example has three staves, but some of them disappear: note how the 2nd line only has the bar number 2. (That the bar number is printed might be considered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

`hara-kiri-tabstaff.ly`

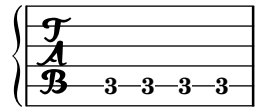


2

3

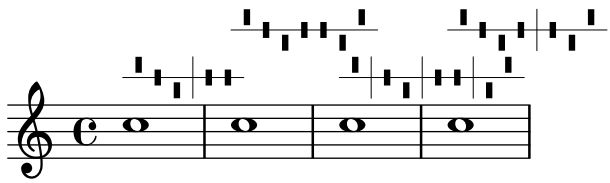


4



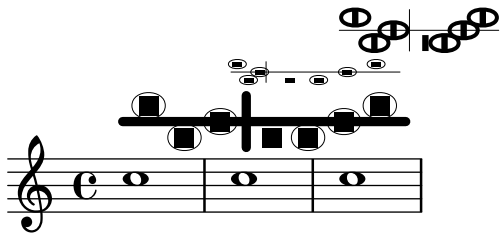
The harp-pedal markup function does some sanity checks. All the diagrams here violate the standard (7 pedals with divider after third), so a warning is printed out, but they should still look okay.

`harp-pedals-sanity-checks.ly`



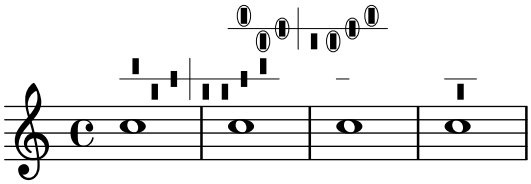
Harp pedals can be tweaked through the size, thickness and harp-pedal-details properties of TextScript.

`harp-pedals-tweaking.ly`



Basic harp diagram functionality, including circled pedal boxes. The third diagram uses an empty string, the third contains invalid characters. Both cases will create warnings, but should still not fail with an error.

`harp-pedals.ly`



A second book-level header block and headers nested in bookpart and score should not clear values from the first header block. This score should show composer, piece, subtitle and title.

header-book-multiple.ly

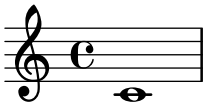
Title correct (superseded at book level)

Subtitle correct (superseded in bookpart)

Composer correct (set in book

Note: title, subtitle, piece, and composer expected.

Piece correct (superseded in score)

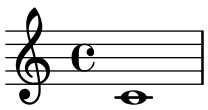


Changing the header fields in a book or a bookpart shall not have any effect on the global default values.

`header-book-multiplescores.ly`

Title correct (set at top level)

Note: expect only title.



A second bookpart-level header block shall retain previously set values from a first header block at the same or higher levels unless overriden.

header-bookpart-multiple.ly
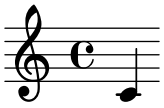
Title correct (set in book)

Subtitle correct (superseded in bookpart)

Composer correct (set at top level

Note: expect title, subtitle, piece and composer.

Piece correct (superseded at bookpart level)
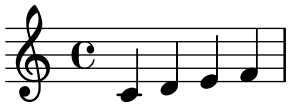


Cyclic references in header fields should cause a warning, but not crash LilyPond with an endless loop

`header-cyclic-reference.ly`

Cyclic reference to

Cyclic reference to Cyclic reference to



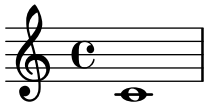
A second score-level header block shall not entirely replace a first header block, but only update changed variables.

`header-score-multiple.ly`

Note: expect piece and opus.

Piece correct (set in score)

Opus correct (superseded at score level)



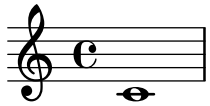
Header blocks may appear before and after the actual music in a score.

`header-score-reordered.ly`

Note: expect piece and opus.

Piece correct (set in score)

Opus correct (superseded at score level)
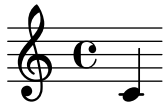


A second top-level header block shall not entirely replace a first header block, but only changed variables.

`header-toplevel-multiple.ly`

Title correct (superseded at top level)

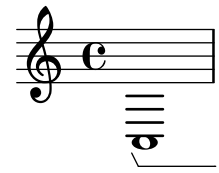
Note: expect title and piece.

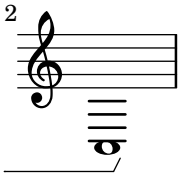
Piece correct (set at top level)



Horizontal brackets connect over line breaks.

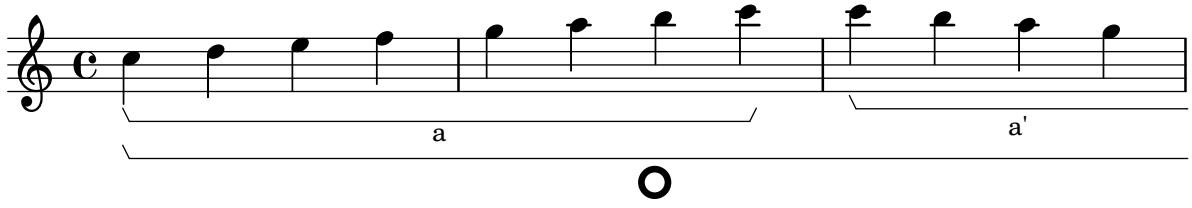
`horizontal-bracket-break.ly`

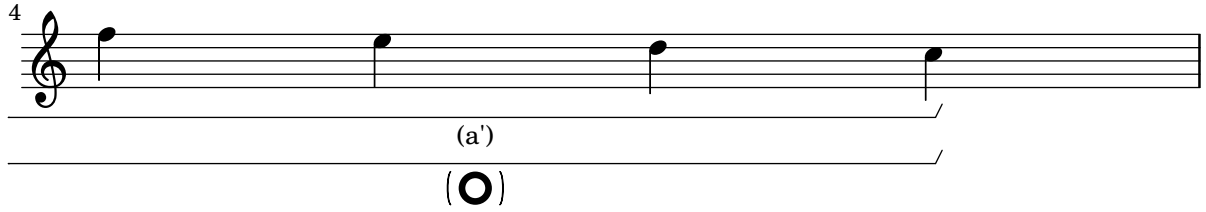




Text is parenthesized when analysis brackets cross line breaks.

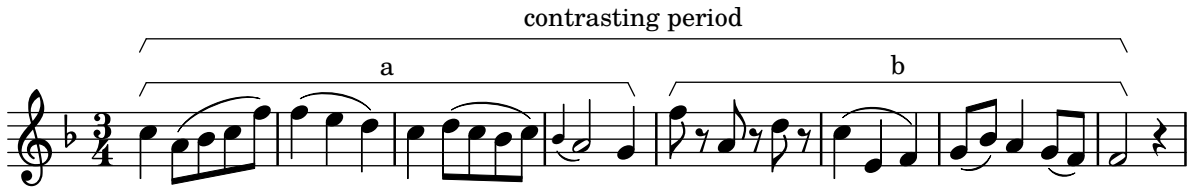
`horizontal-bracket-broken-texted.ly`





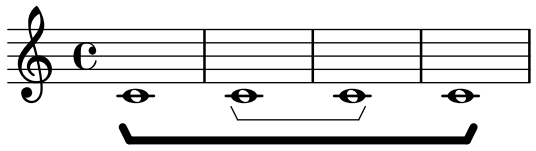
Labels may be added to analysis brackets through the `text` property of the `HorizontalBracketText` object. Use of the `weak` command is necessary for assigning text uniquely to brackets beginning at the same moment. Text assignments reflect the usual nesting order of brackets.

`horizontal-bracket-texted.ly`



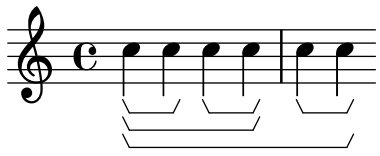
Horizontal brackets are created with the correct event-cause, ensuring tweaks are applied to the correct spanner.

`horizontal-bracket-tweak.ly`



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

`horizontal-bracket.ly`
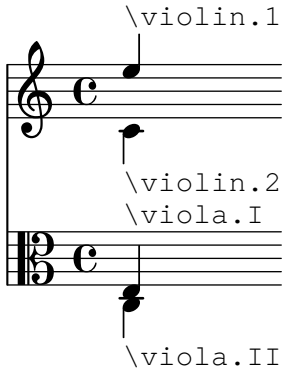


Shows the id property of a grob being set. This should have no effect.

`id.ly`
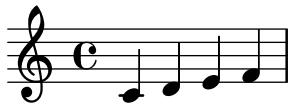


Music variables may be structured into alists indexed by numbers or symbols.

identifier-alists.ly



Identifiers following a chordmode section are not interpreted as chordmode tokens. In the following snippet, the identifier ‘m’ is not interpreted by the lexer as a minor chord modifier.

identifier-following-chordmode.ly
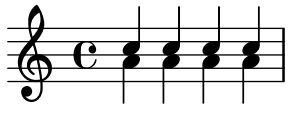


Music identifiers containing arbitrary characters may be initialized using

```
"violin1" = { c''4 c'' c'' c'' }
```

and used as:

```
\new Voice { \"violin1" }
```

identifier-quoted.ly
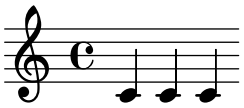


test identifiers.

identifiers.ly

title

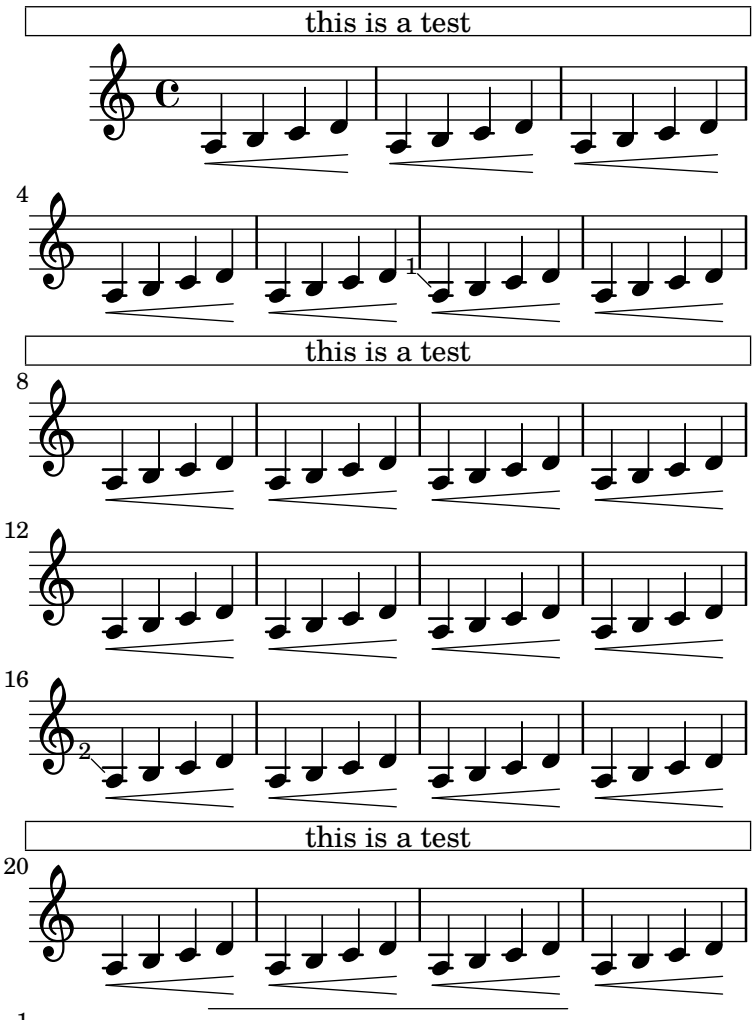
Composer

hoi *polloi*
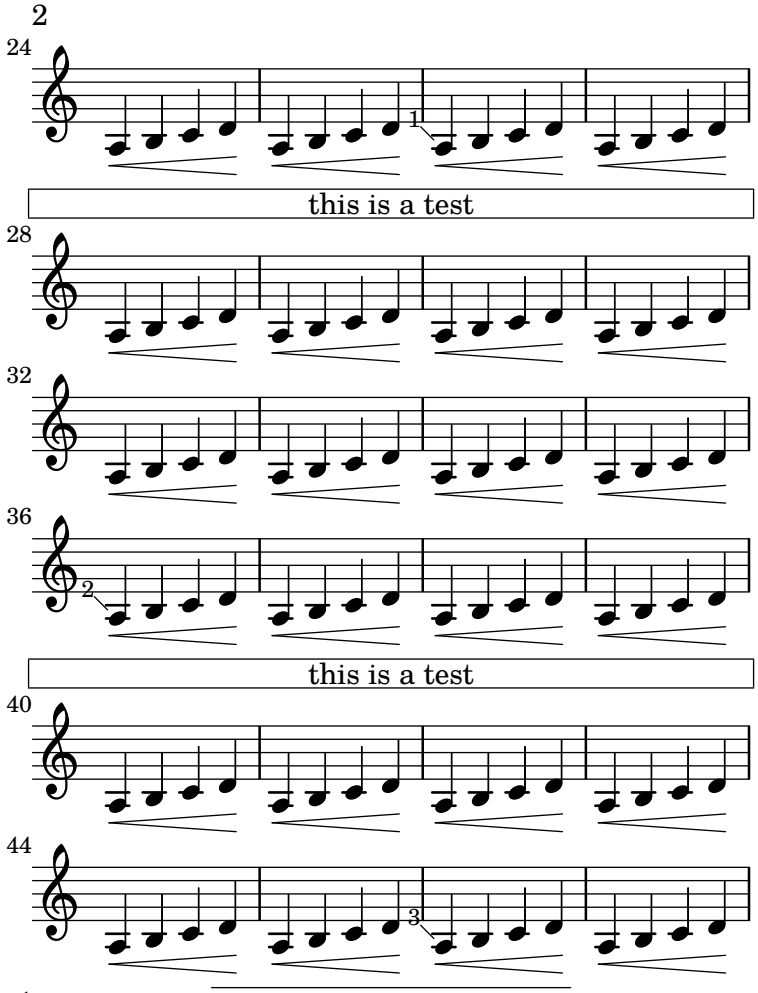


LilyPond does in-notes.

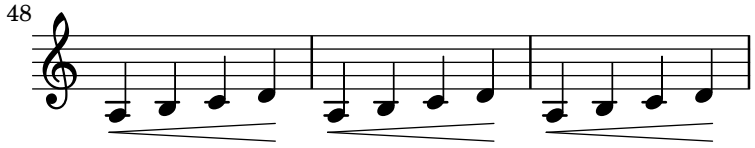
`in-note.ly`



[1]foobar
[2]foobar
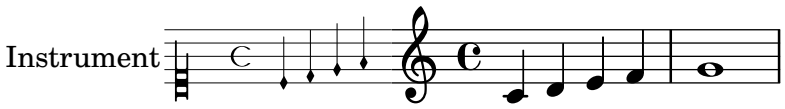


[1]foobar
[2]foobar
[3]foobar



Music engraving by LilyPond 2.21.1—www.lilypond.org
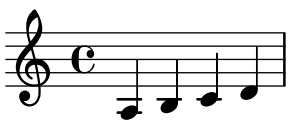
Incipits can be printed using an `InstrumentName` grob.

`incipit.ly`



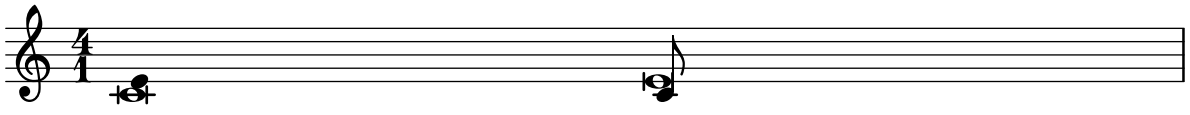
`ly:parser-include-string` should include the current string like a file `\include`.
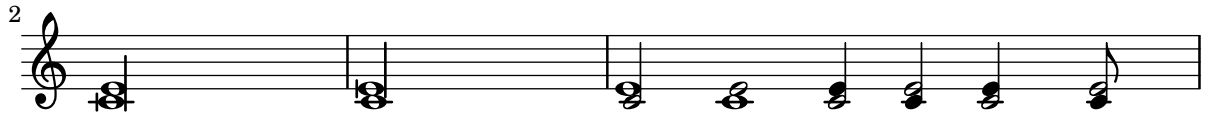
`include-string.ly`



Combine several kinds of stems in parallel voices.
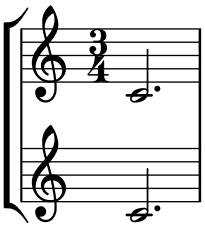
`incompatible-stem-warning.ly`





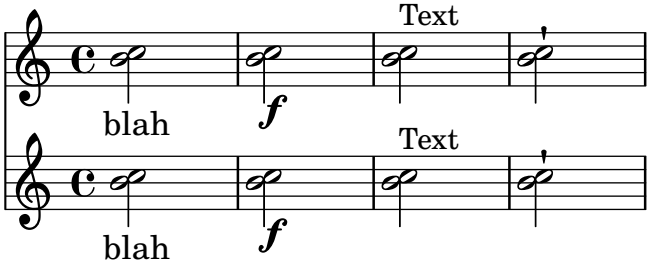
`\inherit-acceptability` allows for one context def to be accepted wherever an existing one is.

`inherit-acceptability.ly`
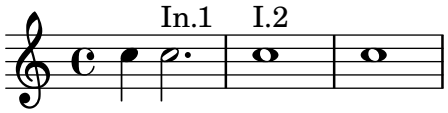


Alignment of lyrics, dynamics, textscripts and articulations attached to chords with suspended notes doesn't depend on input order. All these items are aligned on the "main" notehead (the one at the end of the stem).

`input-order-alignment.ly`



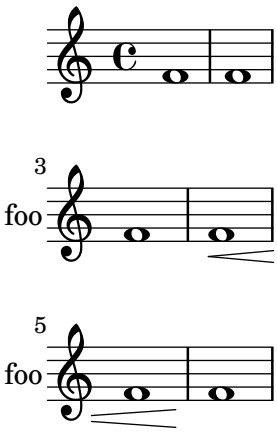
The `Voice.instrumentCueName` property generates instrument names for cue notes. It can also be unset properly.

`instrument-cue-name.ly`



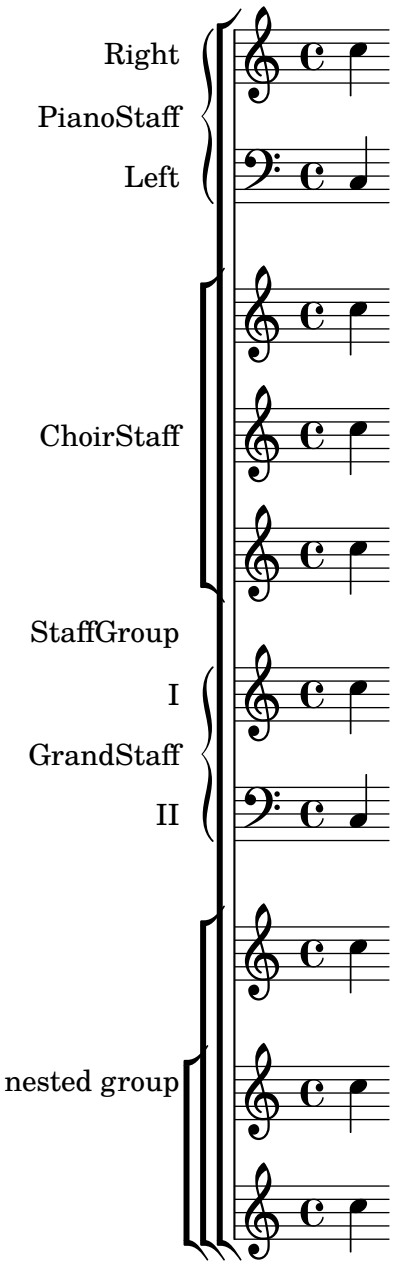
Instrument names (aligned on axis group spanners) ignore dynamic and pedal line spanners.

`instrument-name-dynamic.ly`



Instrument names can also be attached to staff groups.

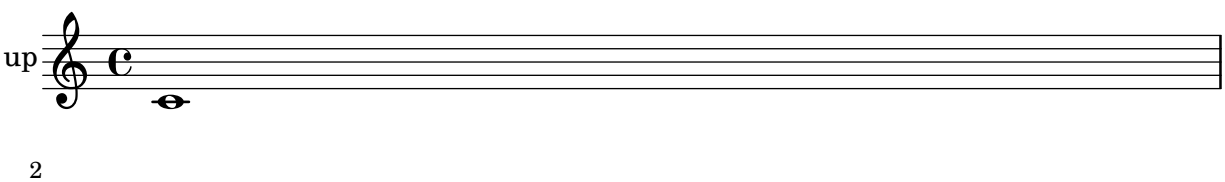
`instrument-name-groups.ly`



Instrument names are removed when the staves are killed off.

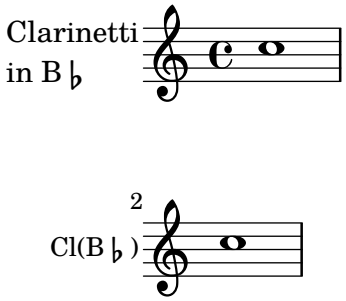
In this example, the second staff (marked by the bar number 2) disappears, as does the instrument name.

`instrument-name-hara-kiri.ly`
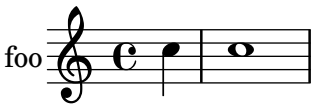


Instrument names are set with `Staff.instrument` and `Staff.instr`. You can enter markup texts to create more funky names, including alterations.

`instrument-name-markup.ly`



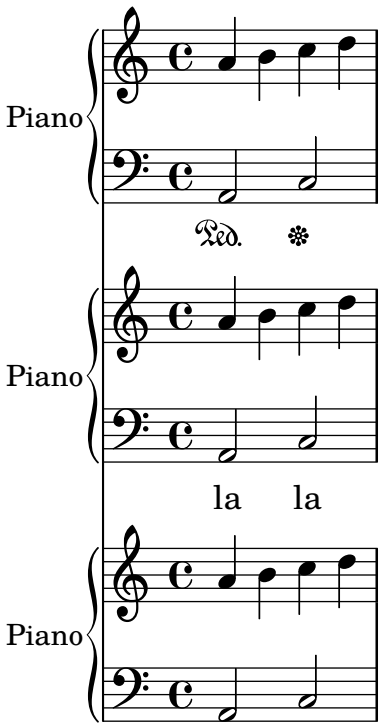
Instrument names are also printed on partial starting measures.

`instrument-name-partial.ly`



`Dynamics` and `Lyrics` lines below a `PianoStaff` do not affect the placement of the instrument name.

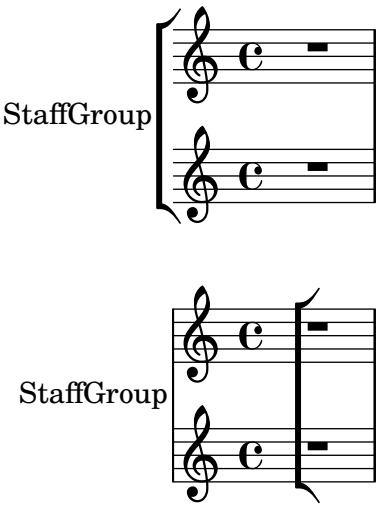
`instrument-name-pedal-lyrics.ly`



`InstrumentName` is reasonable positioned even for unusual system-start-delimiters.

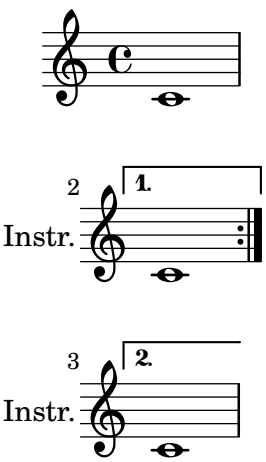
Below, the `instrumentName` neither collides with the `SystemStartBracket` nor moves to far to the left.

`instrument-name-system-start-delimiter.ly`



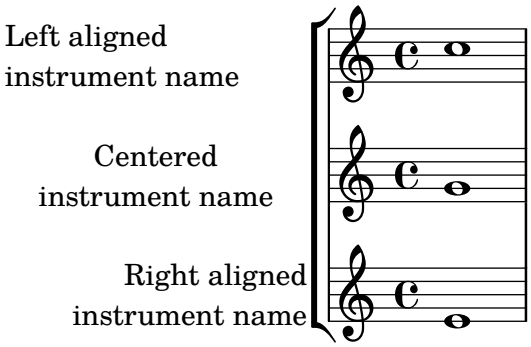
Moving the `Volta_engraver` to the `Staff` context does not affect `InstrumentName` alignment.

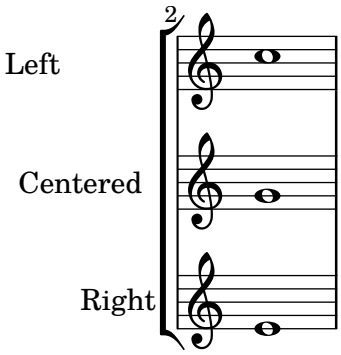
`instrument-name-volta.ly`



Instrument names horizontal alignment is tweaked by changing the `Staff.InstrumentName.self-alignment-X` property. The `\layout` variables `indent` and `short-indent` define the space where the instrument names are aligned before the first and the following systems, respectively.

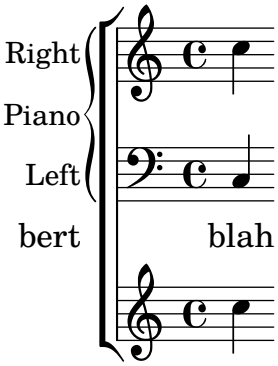
`instrument-name-x-align.ly`
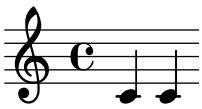




Staff margins are also markings attached to barlines. They should be left of the staff, and be centered vertically with respect to the staff. They may be on normal staves, but also on compound staves, like the PianoStaff.

`instrument-name.ly`
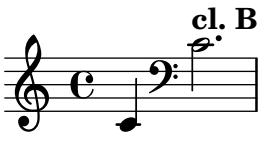


The `switchInstrument` music function prints a warning if the given instrument definition does not exist.

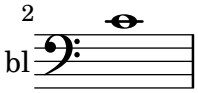
`instrument-switch-invalid-warning.ly`

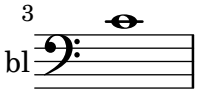


The `switchInstrument` music function modifies properties for an in staff instrument switch.

`instrument-switch.ly`







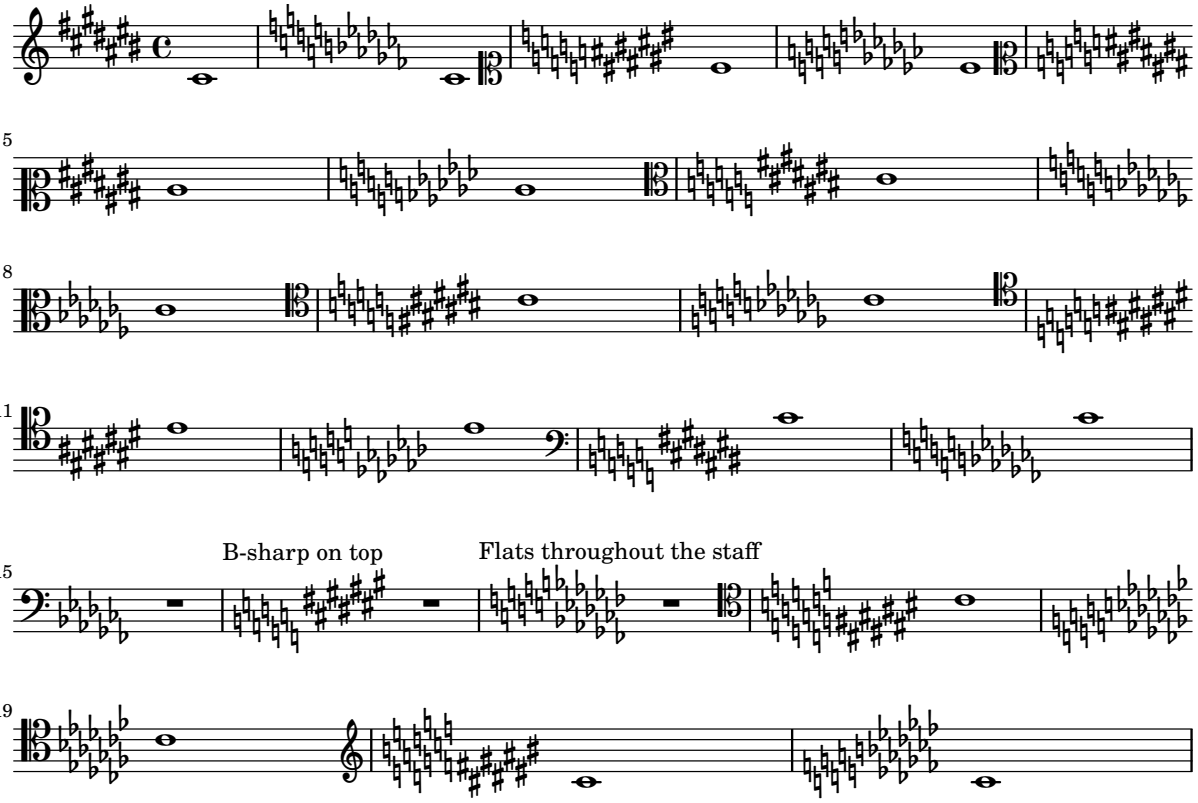
Engravers which do not exist produce a warning.

`invalid-engraver.ly`
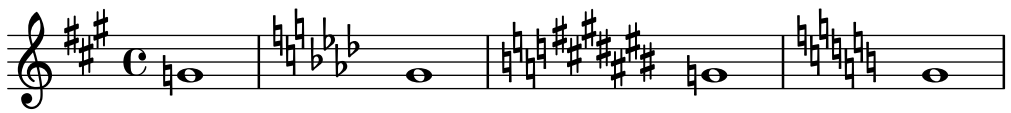


Each clef has its own accidental placing rules, which can be adjusted using `sharp-positions` and `flat-positions`.

`key-clefs.ly`
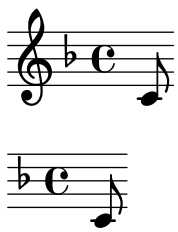


Key cancellation signs consists of naturals for pitches that are not in the new key signature. Naturals get a little padding so the stems don't collide.

`key-signature-cancellation.ly`



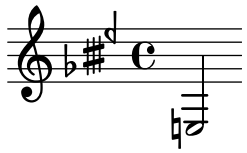
If the clef engraver is removed, the key signature shall use a proper padding > 0 to the start of the staff lines.

`key-signature-left-edge.ly`



With the `padding-pairs` property, distances between individual key signature items can be adjusted.
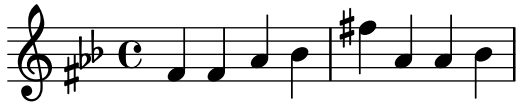
`key-signature-padding.ly`



When a custom key signature has entries which are limited to a particular octave, such alterations should persist indefinitely or until a new key signature is set.
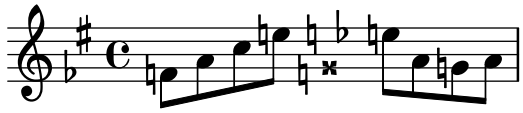
Here, only the fis' shows an accidental, since it is outside the octave defined in `keyAlterations`.

`key-signature-scordatura-persist.ly`



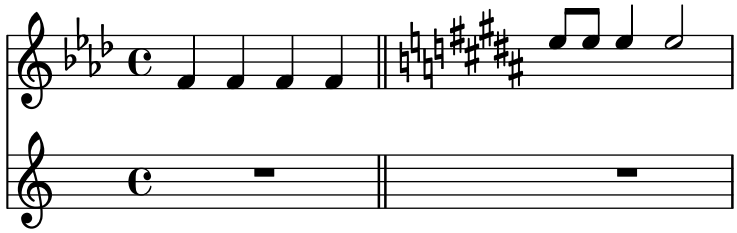
By setting `Staff.keyAlterations` directly, key signatures can be set invidually per pitch.

`key-signature-scordatura.ly`
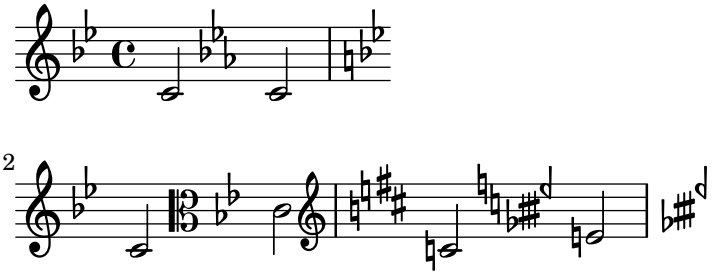


Key signatures get the required amount of horizontal space.

`key-signature-space.ly`
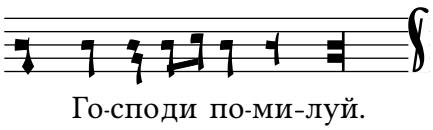


Key signatures may appear on key changes, even without a barline. In the case of a line break, the restoration accidentals are printed at end of a line. If `createKeyOnClefChange` is set, key signatures are created also on a clef change.

`keys.ly`



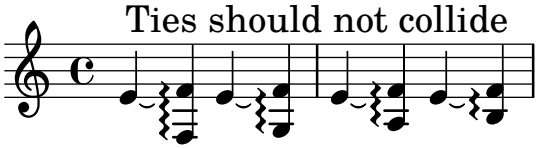
LilyPond typesets Kievan notation.

`kievan-notation.ly`
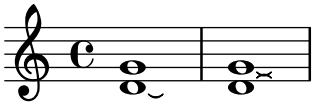


l.v. ties should not collide with arpeggio indications.

`laissez-vibrer-arpeggio.ly`
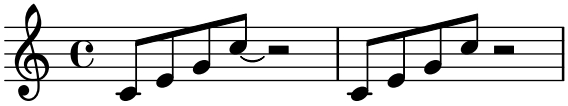


`\laissezVibrer` ties should also work on individual notes of a chord.

`laissez-vibrer-chords.ly`
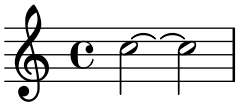


`\laissezVibrer` ties on beamed notes don't trigger premature beam slope calculation.

`laissez-vibrer-tie-beam.ly`
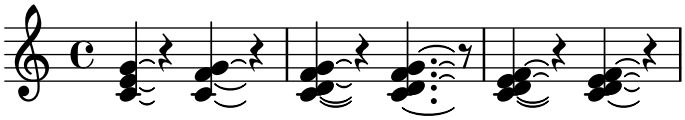


The 'head-direction of a LaissezVibrerTieColumn should be able to be set without causing a segmentation fault.

`laissez-vibrer-tie-head-direction.ly`



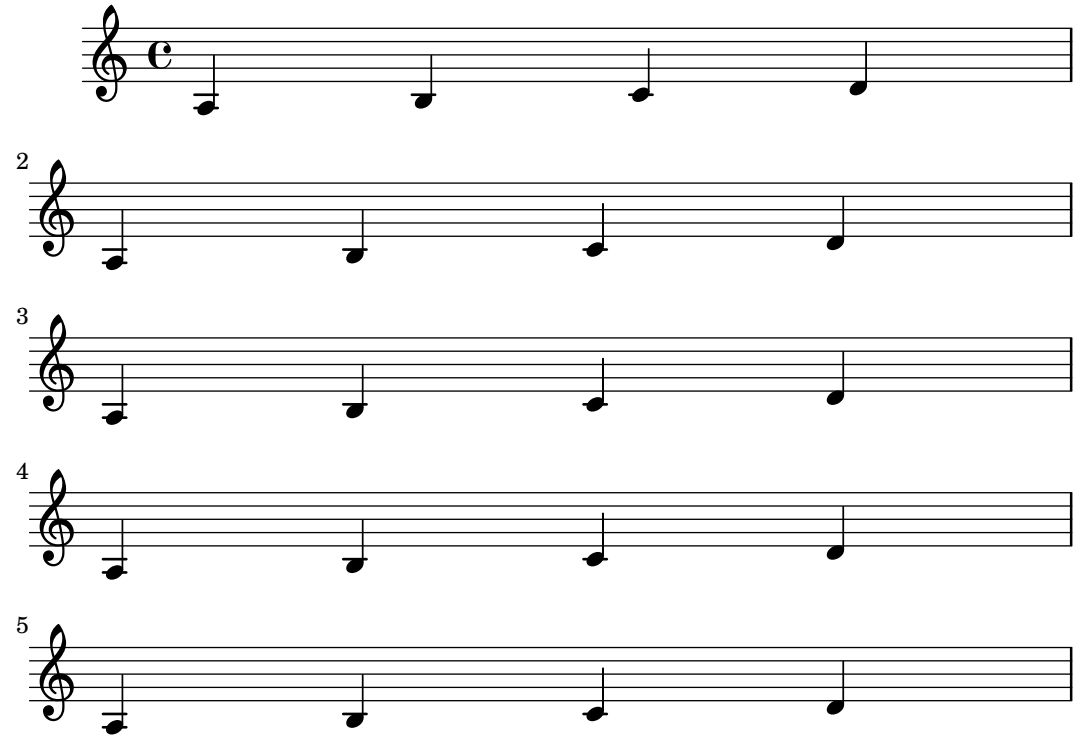
l.v. ties should avoid dots and staff lines, similar to normal ties. They have fixed size. Their formatting can be tuned with `tie-configuration`.
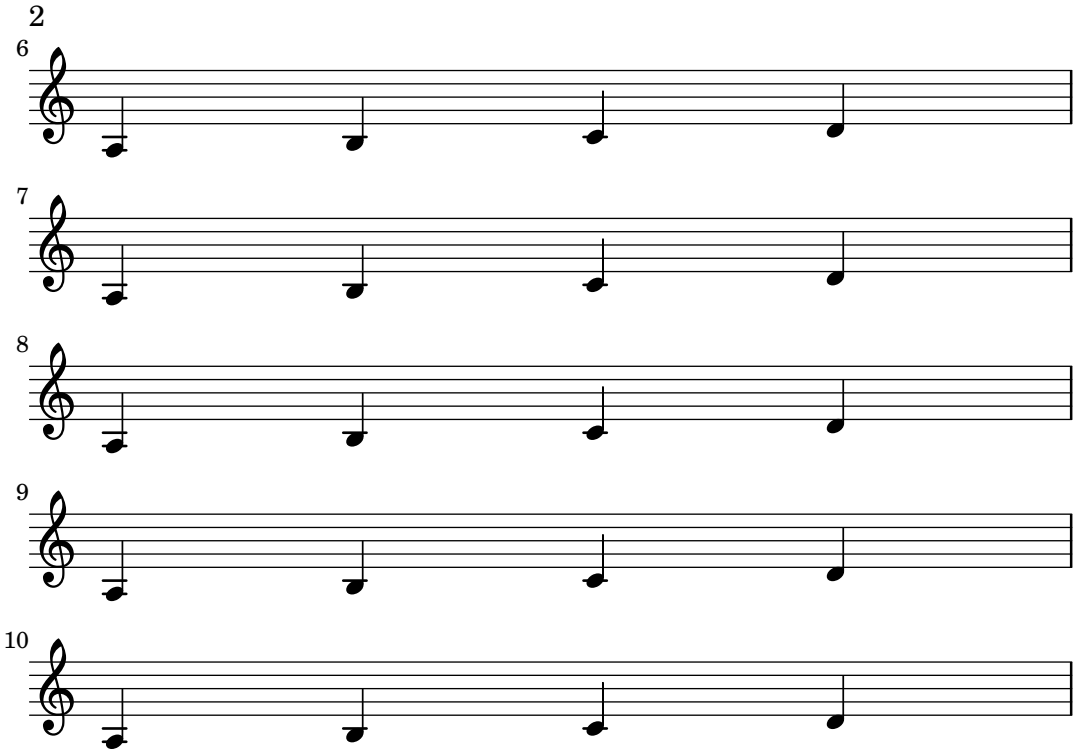
`laissez-vibrer-ties.ly`

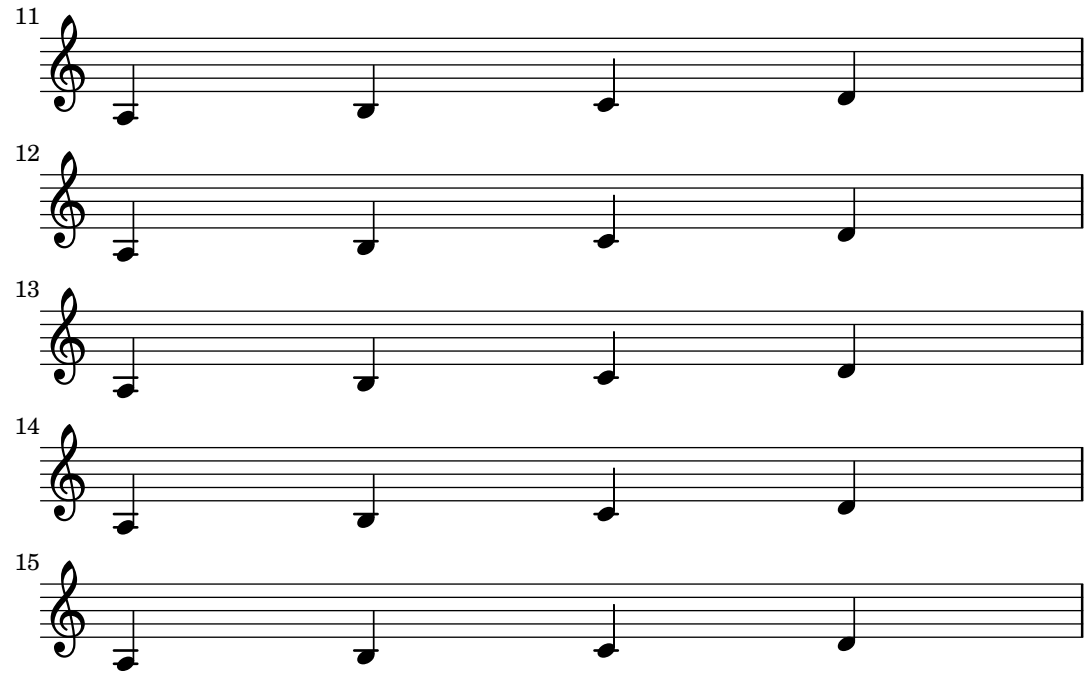


Scores may be printed in landscape mode.

landscape.ly
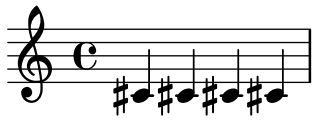






Music engraving by LilyPond 2.21.1—www.lilypond.org
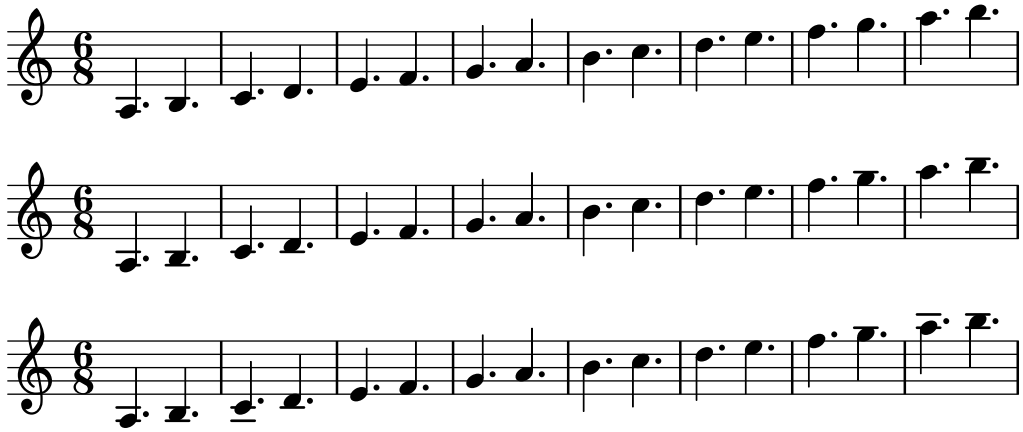
Inside of output definitions like `\layout` or `\midi`, music is harvested for layout definitions in order to turn them into context modifications.

`layout-from.ly`
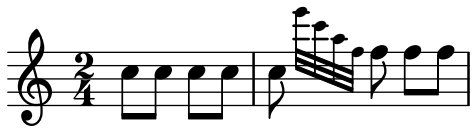


The ledger-extra grob property increases the number of ledger lines drawn, but they are not drawn on top of staff lines.

`ledger-extra.ly`
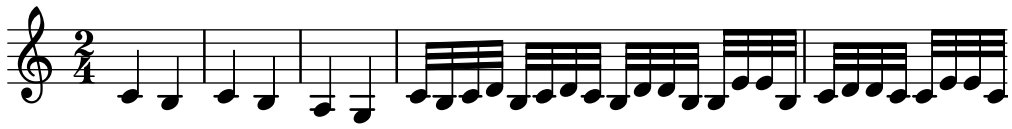


When ledgered notes are very close, for example, in grace notes, they are kept at a minimum distance to prevent the ledgers from disappearing.

`ledger-line-minimum.ly`
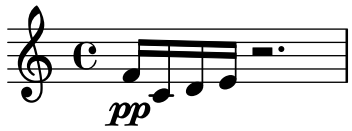


Ledger lines are shortened when they are very close. This ensures that ledger lines stay separate.

`ledger-line-shorten.ly`
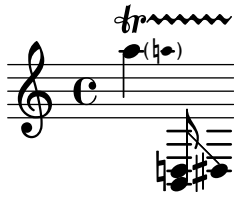


Dynamics and other outside staff objects avoid ledger lines.

`ledger-lines-dynamics.ly`
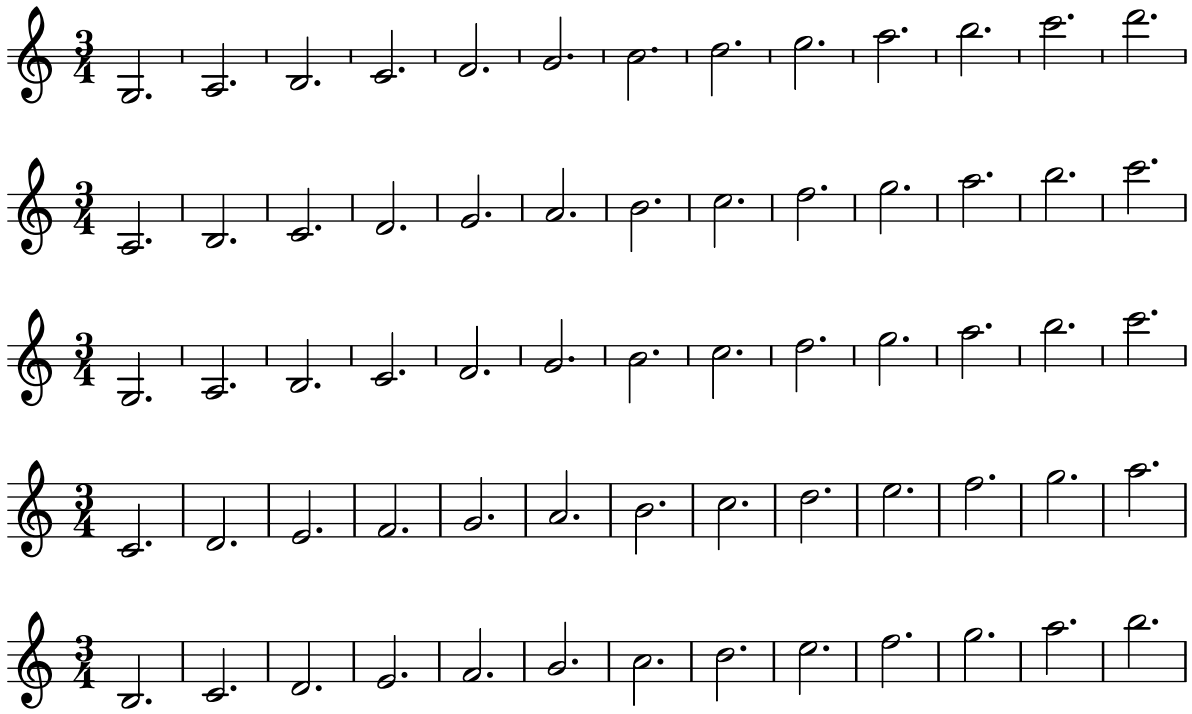


In some rare cases like these the extents of two ledger lines at the same vertical position in the same note column do not overlap horizontally, and they should not be merged into a single ledger line. See LSR 505: Displaying complex chords http://lsr.di.unimi.it/LSR/Item?id=505

`ledger-lines-non-merging.ly`
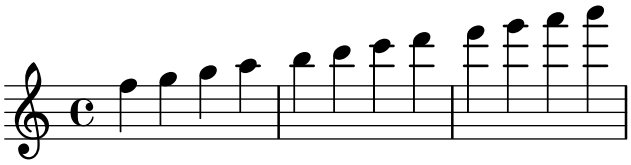


Ledger lines should appear at every other location for a variety of staves using both `line-count` and `line-positions`.

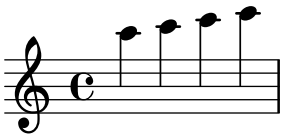
`ledger-lines-varying-staves.ly`

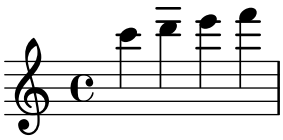


3 ways to customize ledger line positions.

`ledger-positions-customization.ly`







Highly tweaked example of lilypond output
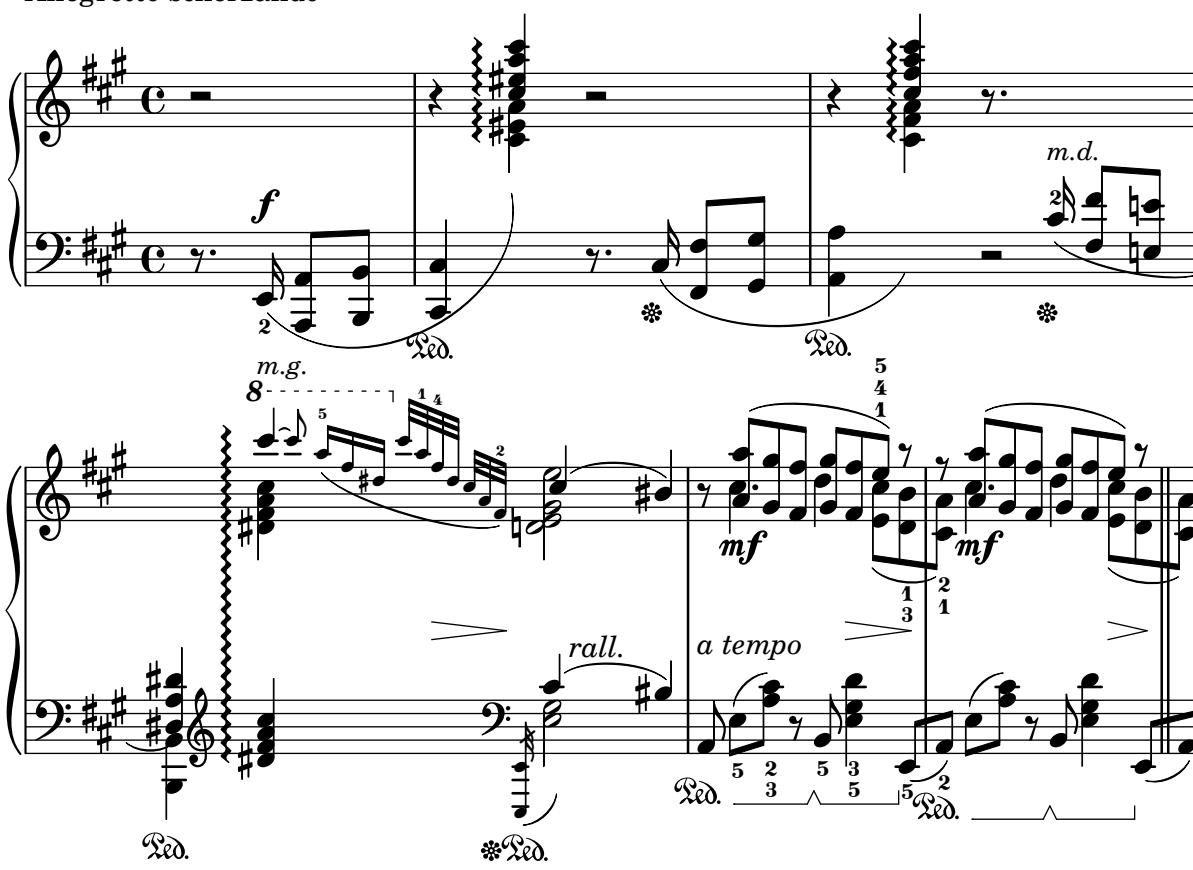
les-nereides.ly

LES NÉRÉIDES

THE NEREIDS

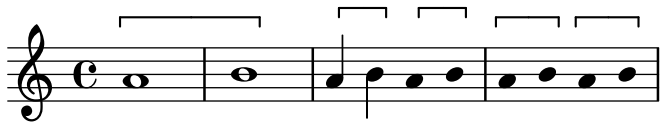
ARTHUR GRAY

Allegretto scherzando



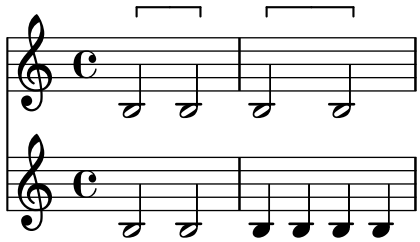
Ligature brackets should align to visible or transparent stems only. For stemless notes they should span the whole note width.

`ligature-bracket-X-positions.ly`



The ligature bracket right-end is not affected by other voices.

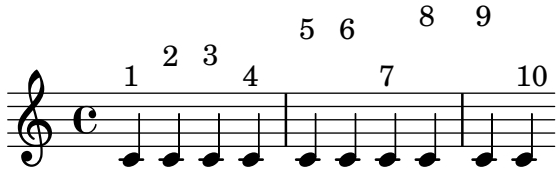
`ligature-bracket.ly`



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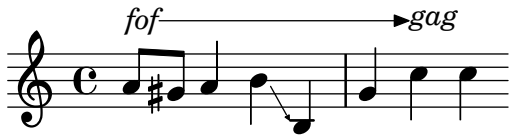
In this example, the `\withpaddingA`, `\withpaddingB` and `\withpaddingC` music functions set different kinds of padding on the `TextScript` grob.

`lily-in-scheme.ly`



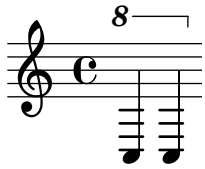
Arrows can be applied to text-spanners and line-spanners (such as the Glissando)

`line-arrows.ly`



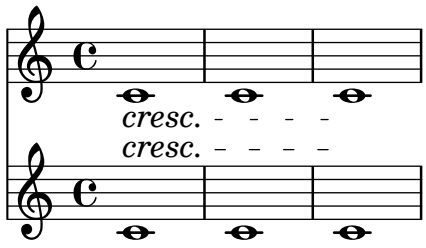
Generate valid postscript even if dash-period is small compared to line thickness.

`line-dash-small-period.ly`



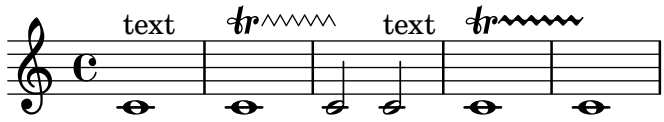
The period of a dashed line is adjusted such that it starts and ends on a full dash.

`line-dashed-period.ly`
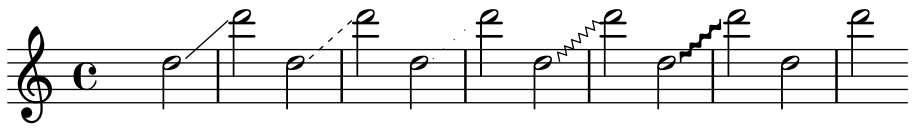


Setting `'zigzag` style for spanners does not cause spacing problems: in this example, the first text markup and zigzag trillspanner have the same outside staff positioning as the second markup and default trillspanner.

`line-style-zigzag-spacing.ly`



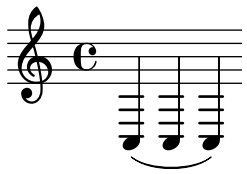
Cover all line styles available.

`line-style.ly`
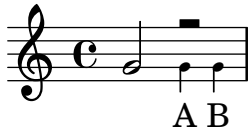


Test the different loglevels of lilypond. Run this file with –loglevel=NONE, ERROR, WARNING, PROGRESS, DEBUG to see the different loglevels. The errors are commented out. Comment them in to check the output manually.

`loglevels.ly`



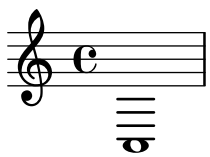
For Voice-derived contexts like CueVoice, the lyrics should still start with the first note.

`lyric-combine-derived-voice.ly`



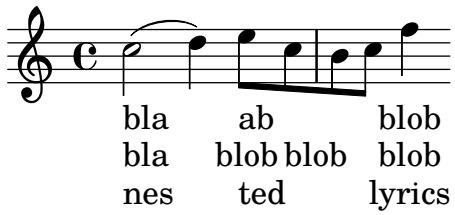
If lyrics are assigned to a non-existing voice, a warning should be printed. However, if the lyrics context does not contain any lyrics, then no warning should be printed.

`lyric-combine-empty-warning.ly`



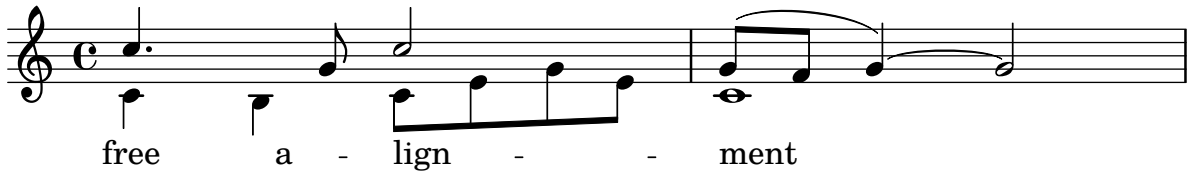
With the `\lyricsto` mechanism, individual lyric lines can be associated with one melody line. Each lyric line can be tuned to either follow or ignore melismata.

`lyric-combine-new.ly`
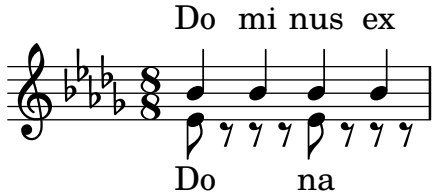


Lyrics can be aligned to a `NullVoice` context, which prints no notes, with the usual mechanisms for melismata.

`lyric-combine-nullvoice.ly`
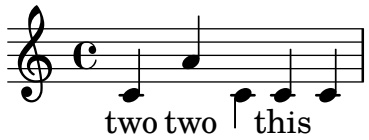


Polyphonic rhythms and rests do not disturb `\lyricsto`.

`lyric-combine-polyphonic.ly`



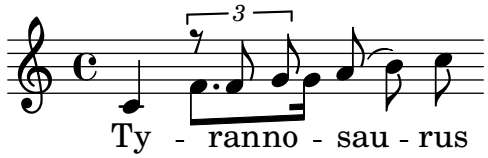
switching voices in the middle of the lyrics is possible using `lyricsto`.

`lyric-combine-switch-voice-2.ly`



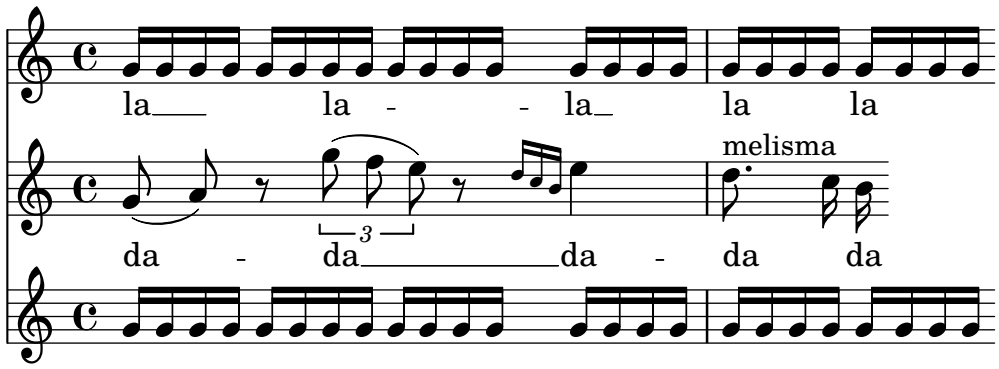
Switching the melody to a different voice works even if the switch occurs together with context instantiation.

`lyric-combine-switch-voice.ly`



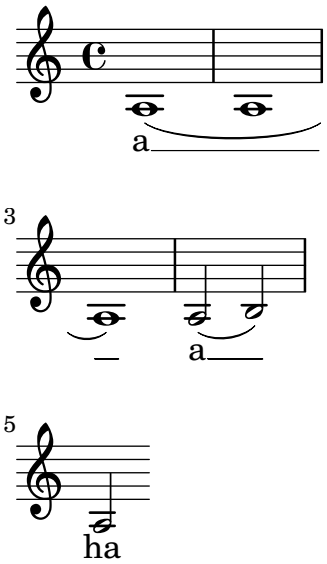
Lyrics can be set to a melody automatically. Excess lyrics will be discarded. Lyrics will not be set over rests. You can have melismata either by setting a property melismaBusy, or by setting automaticMelismas (which will set melismas during slurs and ties). If you want a different order than first Music,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

`lyric-combine.ly`



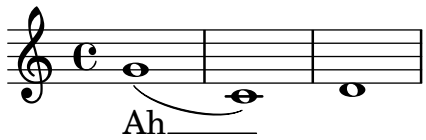
Lyric extenders run to the end of the line if it continues the next line. Otherwise, it should run to the last note of the melisma.

`lyric-extender-broken.ly`
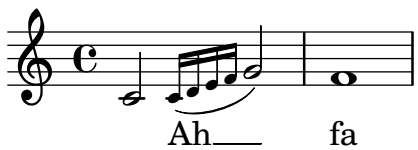


A LyricExtender should end at the right place even if there are more notes in the voice than lyrics.

`lyric-extender-completion.ly`
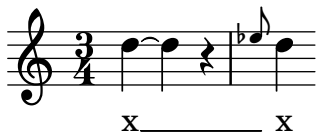


If `includeGraceNotes` is enabled, lyric extenders work as expected also for syllables starting under grace notes.

`lyric-extender-includegraces.ly`
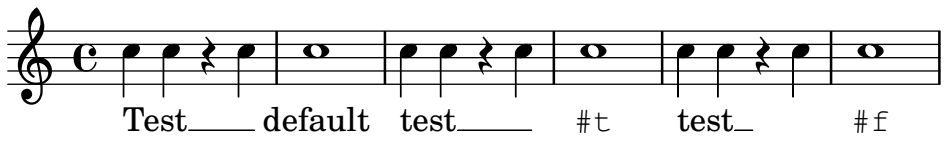


Extender engraver also notices the lack of note heads. Here the extender ends on the 2nd quarter note, despite the grace note without a lyric attached.

`lyric-extender-no-heads.ly`



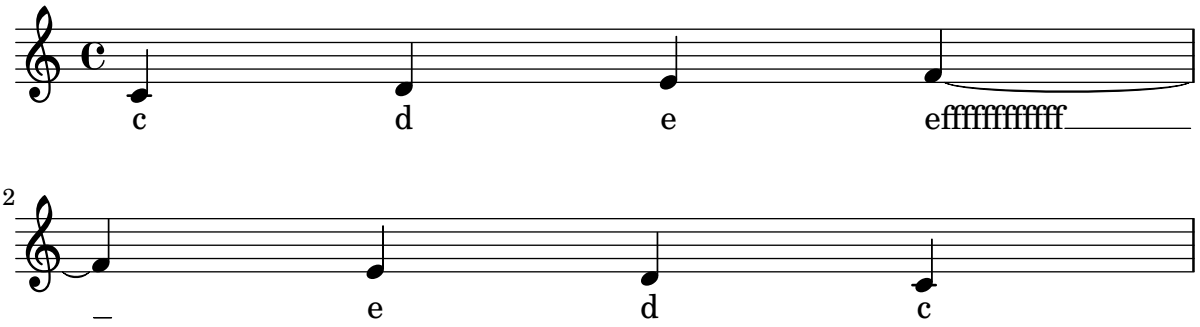
If `extendersOverRests` is set, an extender is not terminated upon encountering a rest.

`lyric-extender-rest.ly`
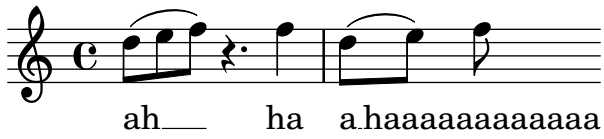


Extenders will not protrude into the right margin

`lyric-extender-right-margin.ly`
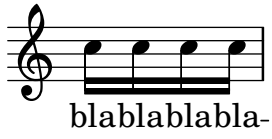


A LyricExtender may span several notes. A LyricExtender does not extend past a rest, or past the next lyric syllable.
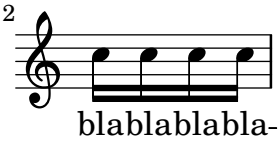
`lyric-extender.ly`
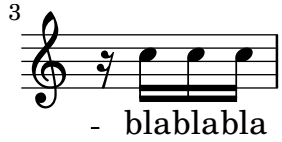


Hyphens are printed at the beginning of the line only when they go past the first note.

`lyric-hyphen-break.ly`

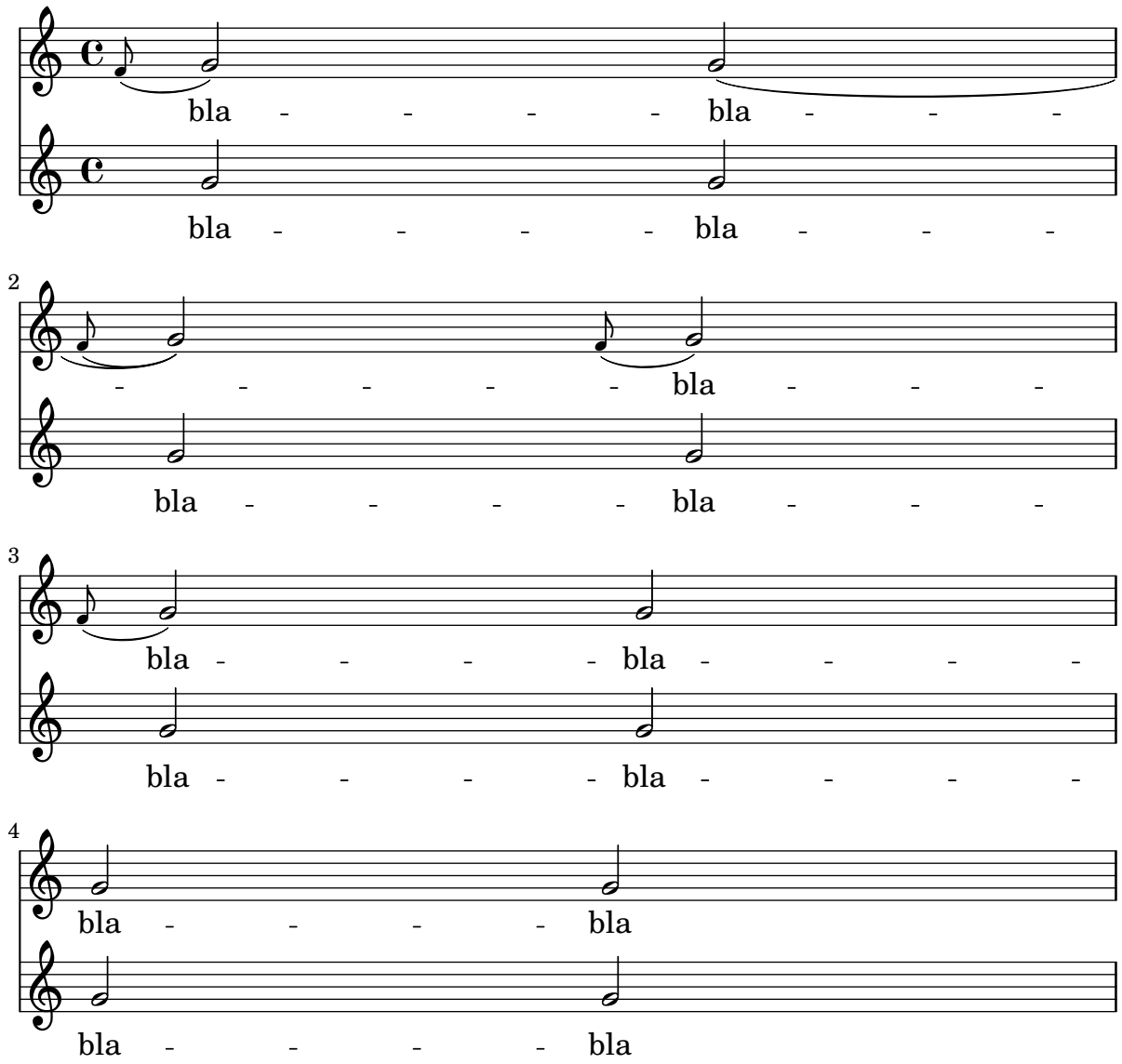






No hyphen should be printed under a grace note at the start of a line if the grace's main note starts a new syllable.

`lyric-hyphen-grace.ly`
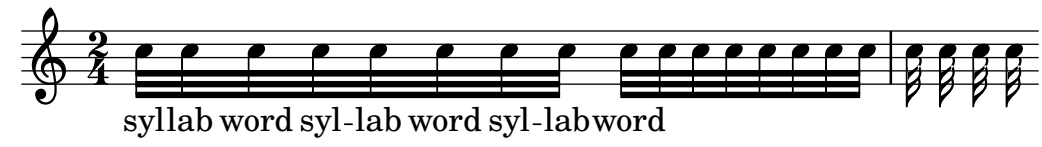


The minimum distance between lyrics is determined by the `minimum-distance` of `LyricHyphen` and `LyricSpace`.

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.

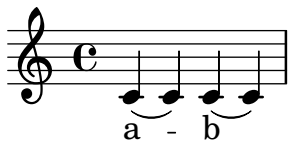
Like all overrides within `\lyricsto` and `\addlyrics`, the effect of a setting is delayed is one syllable.

`lyric-hyphen-retain.ly`



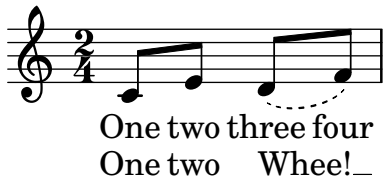
In lyrics, hyphens may be used.

`lyric-hyphen.ly`



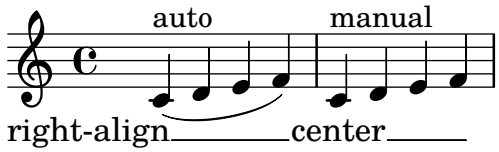
If `ignoreMelismata` is set, lyrics should remain center-aligned.

`lyric-ignore-melisma-alignment.ly`



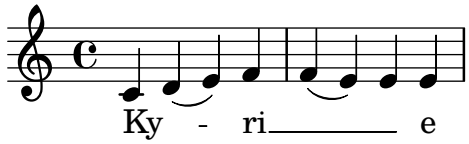
`lyricMelismaAlignment` sets the default alignment for melismata. It works with both automatic and manual melismata.

`lyric-melisma-alignment.ly`
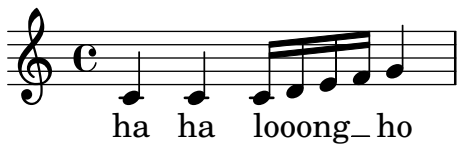


Melismata may be entered manually by substituting _ for lyrics on notes that are part of the melisma.

`lyric-melisma-manual.ly`
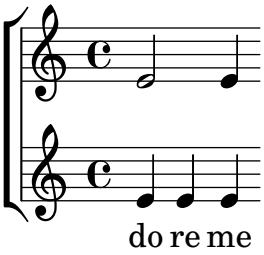


A syllable aligned with a melisma delimited with `\melisma` and `\melismaEnd` should be left-aligned.

`lyric-melisma-melisma.ly`
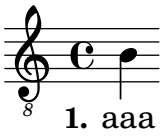


When lyrics are not associated with specific voices, the lyric placement should follow lyric rhythms. In particular, the second syllable here should not be attached to the first note of the first staff.

`lyric-no-association-rhythm.ly`
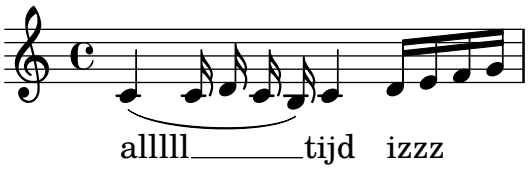


Lyrics should still slide under `TimeSignature` when an `OctaveEight` is present.

`lyric-octave-eight.ly`



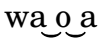
Normally, the lyric is centered on the note head. However, on melismata, the text is left aligned on the left-side of the note head.

`lyric-phrasing.ly`



Tildes in lyric syllables are converted to tie symbols.

`lyric-tie.ly`
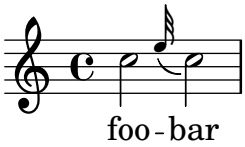


The `\tweak` function can be used in Lyrics.

`lyric-tweak.ly`



Lyrics are ignored for aftergrace notes.

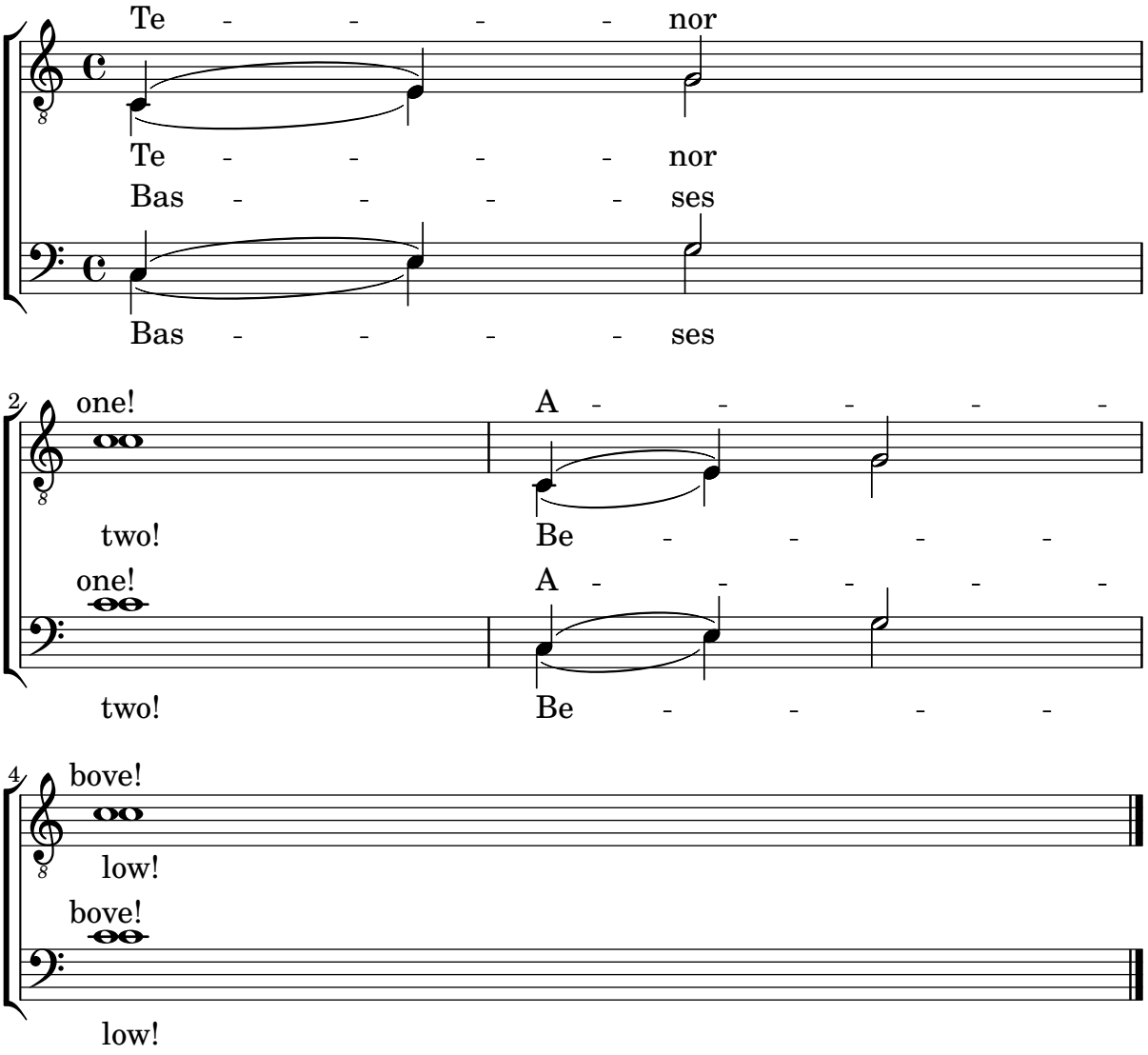
`lyrics-after-grace.ly`



Lyrics aligned above a context should stay close to that context when stretching. The Bass I lyric line stays with the Bass staff.

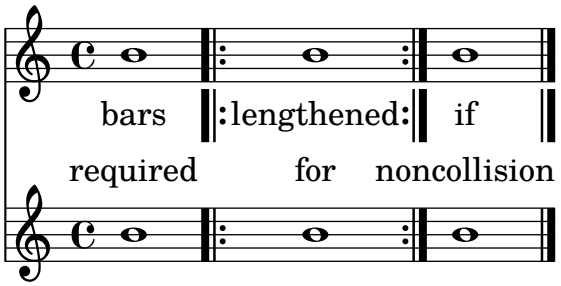
`lyrics-aligned-above-stay-close-to-staff.ly`

Aligned-above lyrics should stay close to their staff
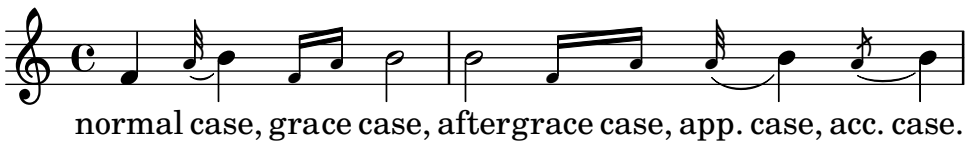


Adding a `Bar_engraver` to the Lyrics context makes sure that lyrics do not collide with barlines.

`lyrics-bar.ly`
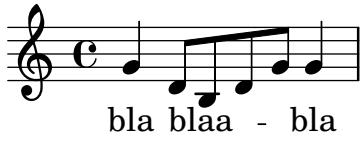


Setting `includeGraceNotes` enables lyrics syllables to be assigned to grace notes.

`lyrics-includegraces.ly`



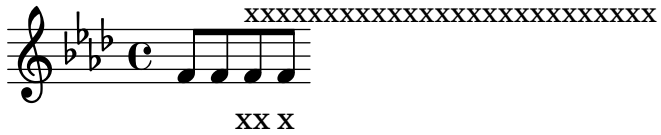
Melismata are triggered by manual beams. Notes in a melisma take their natural spacing over a long syllable.

`lyrics-melisma-beam.ly`
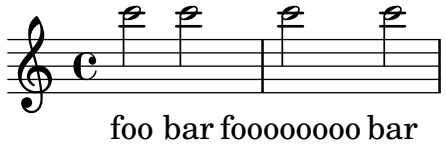


Lyric syllables without note attachment are aligned correctly even if the paper column is very wide.

`lyrics-no-notes.ly`



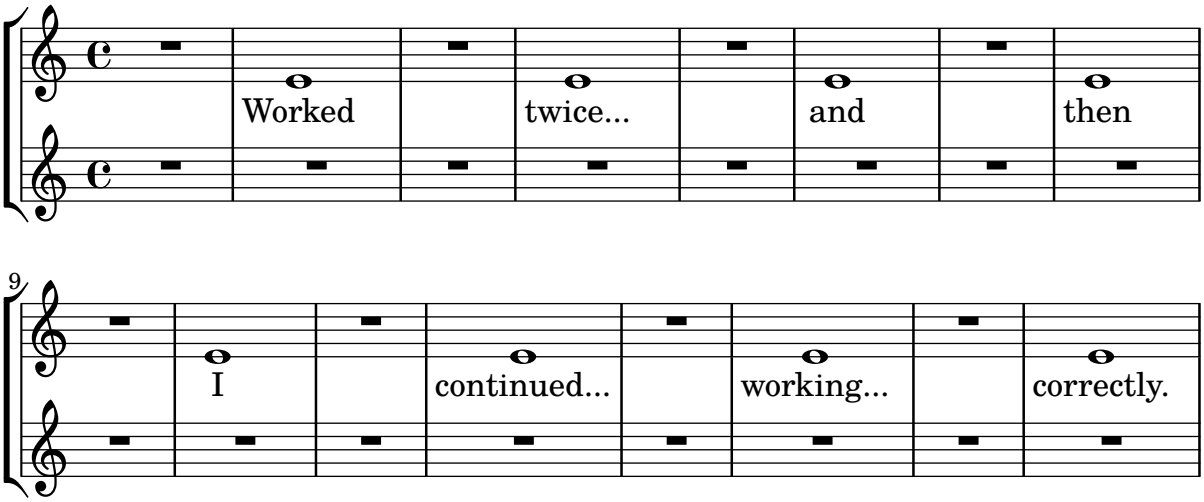
Long lyrics should be allowed to pass under the bar line.

`lyrics-pass-under-bar.ly`
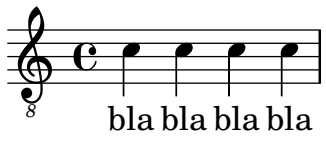


Empty measures do not confuse `SpanBarStub`. These lyrics should remain clear of the span bars.

`lyrics-spanbar.ly`
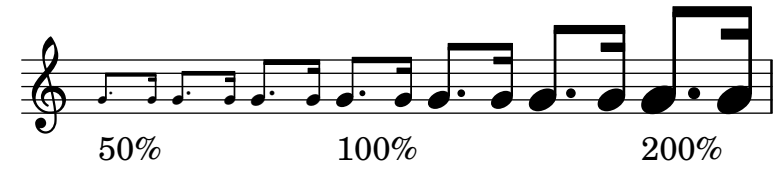


Lyrics are not lowered despite the presence of a clef transposition (8 below the clef).

`lyrics-tenor-clef.ly`



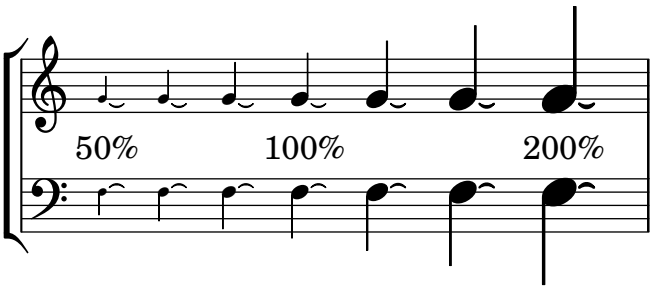
Dot size and beamlet length should be scaled along with notation size when using the `\magnifyMusic` command.

`magnifyMusic-dots-beamlets.ly`
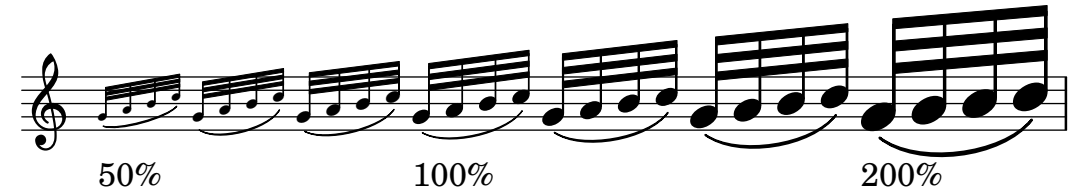


Laissez vibrer ties should be scaled along with notation size when using the `\magnifyMusic` command. They can get thicker than the default, but not thinner.

`magnifyMusic-laissez-vibrer-ties.ly`



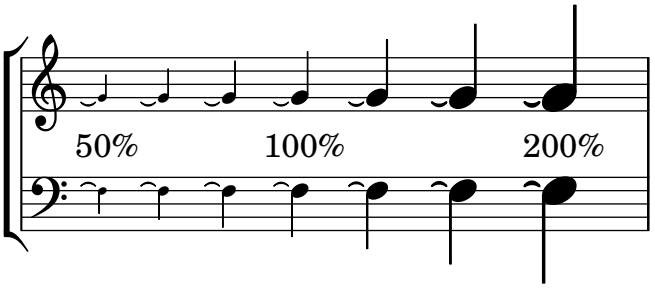
Phrasing slurs should be scaled along with notation size when using the `\magnifyMusic` command. They can get thicker than the default, but not thinner.

`magnifyMusic-phrasing-slurs.ly`
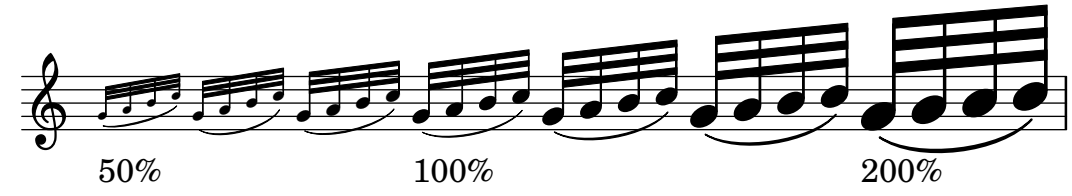


Repeat ties should be scaled along with notation size when using the `\magnifyMusic` command. They can get thicker than the default, but not thinner.

`magnifyMusic-repeat-ties.ly`



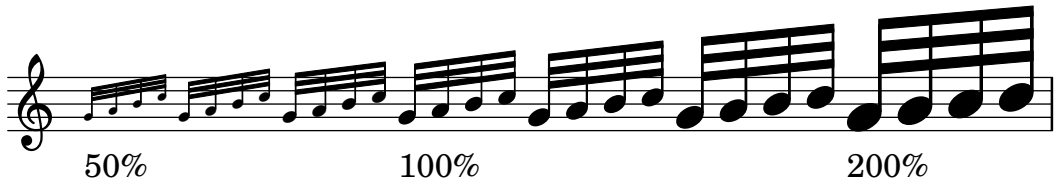
Slurs should be scaled along with notation size when using the `\magnifyMusic` command. They can get thicker than the default, but not thinner.

`magnifyMusic-slurs.ly`



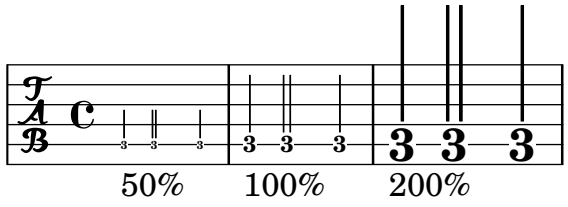
Stem length/thickness, beam spacing/thickness, and horizontal spacing should be scaled along with notation size when using the `\magnifyMusic` command. Stems can get thicker than the default, but not thinner.

`magnifyMusic-stem-beam-spacing.ly`



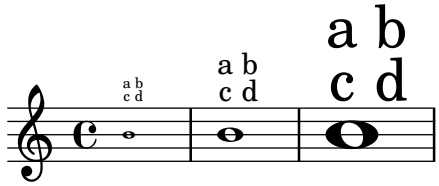
Tablature half-note double-stems should be scaled along with notation size when using the `\magnifyMusic` command.

`magnifyMusic-tablature-double-stems.ly`



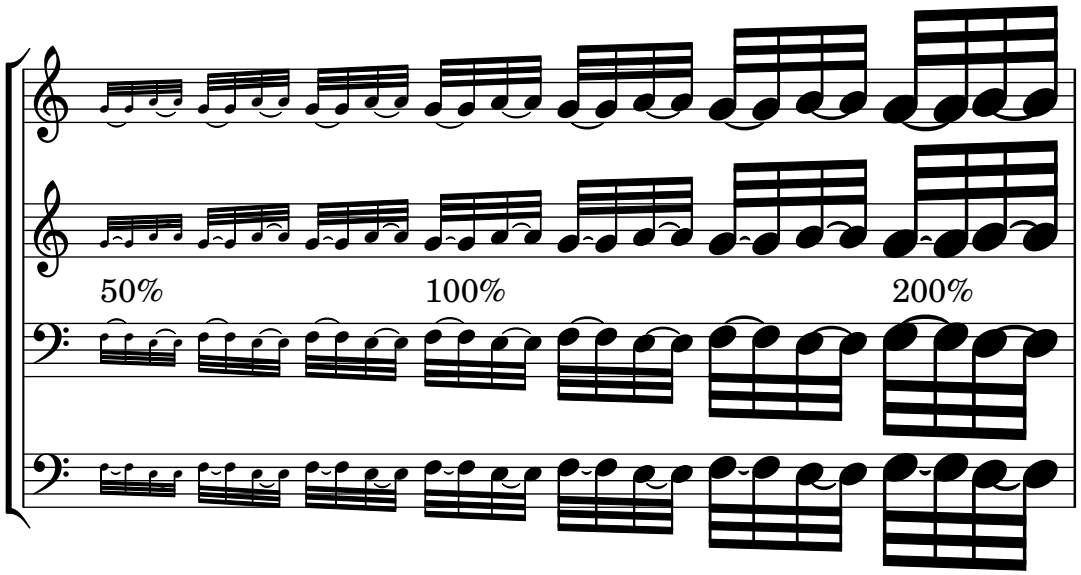
All text-interface grobs should have `baseline-skip` and `word-space` values scaled along with notation size when using the `\magnifyMusic` command.

`magnifyMusic-text-interface.ly`



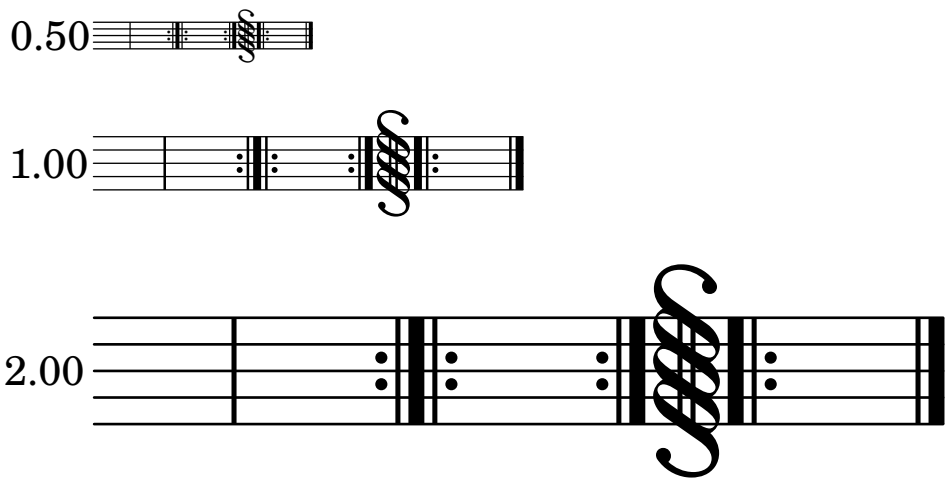
Ties should be scaled along with notation size when using the `\magnifyMusic` command. They can get thicker than the default, but not thinner.

`magnifyMusic-ties.ly`
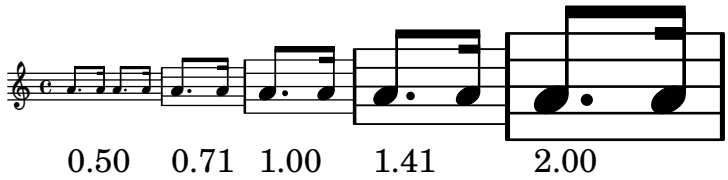


Bar line thickness and spacing should be scaled along with notation size when using the `\magnifyStaff` command.

`magnifyStaff-bar-lines.ly`



Dot size and beamlet length should be scaled along with notation size when using the `\magnifyStaff` command.
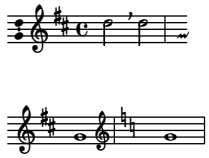
`magnifyStaff-dots-beamlets.ly`
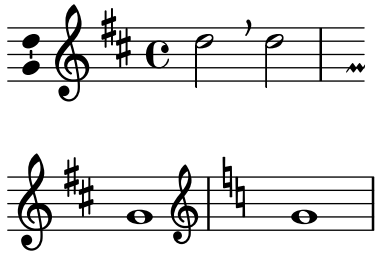


`space-alist` values should be scaled along with notation size when using the `\magnifyStaff` command.

`magnifyStaff-space-alist.ly`
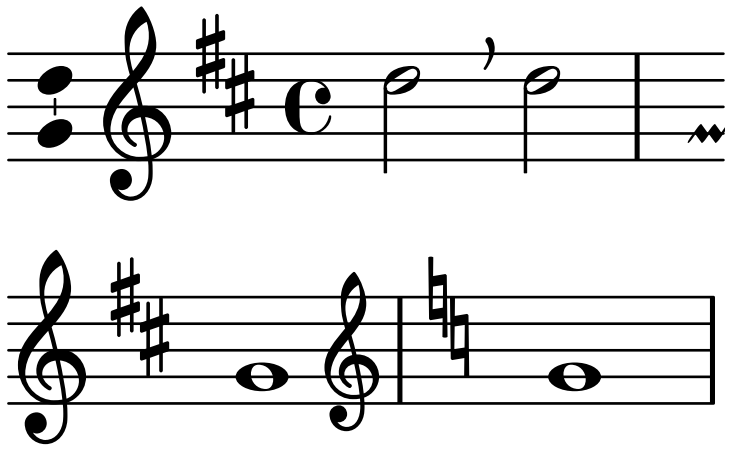
0.50:



1.00:

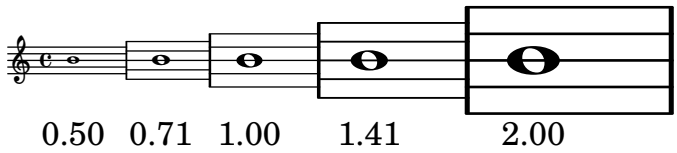


2.00:



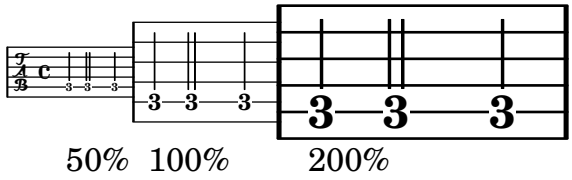
Staff line thickness should be scaled along with staff size when using the `\magnifyStaff` command. Staff lines can get thicker than the default, but not thinner.

`magnifyStaff-staff-line-thickness.ly`



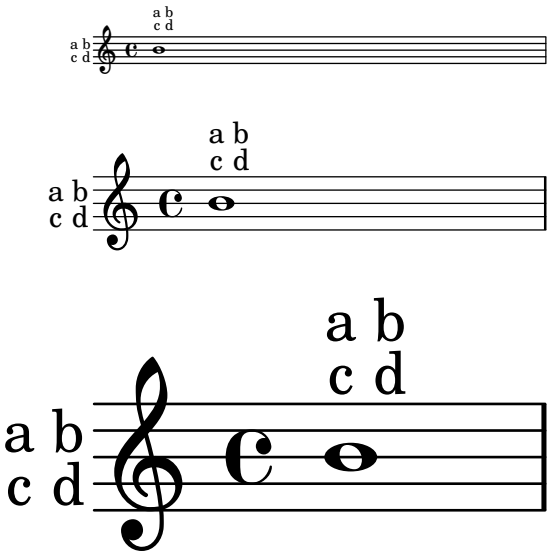
Tablature half-note double-stems should be scaled along with notation size when using the `\magnifyStaff` command.

`magnifyStaff-tablature-double-stems.ly`



All text-interface grobs that are within the Staff context should have `baseline-skip` and `word-space` values scaled along with notation size when using the `\magnifyStaff` command.
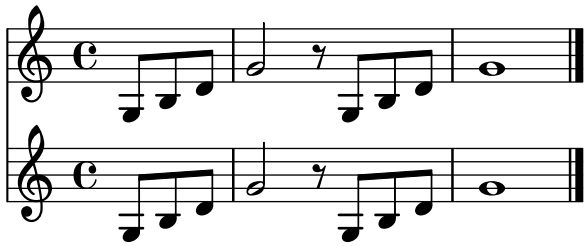
`magnifyStaff-text-interface.ly`



`make-relative` has to copy its argument expressions in case the generated music expression is getting copied and modified.

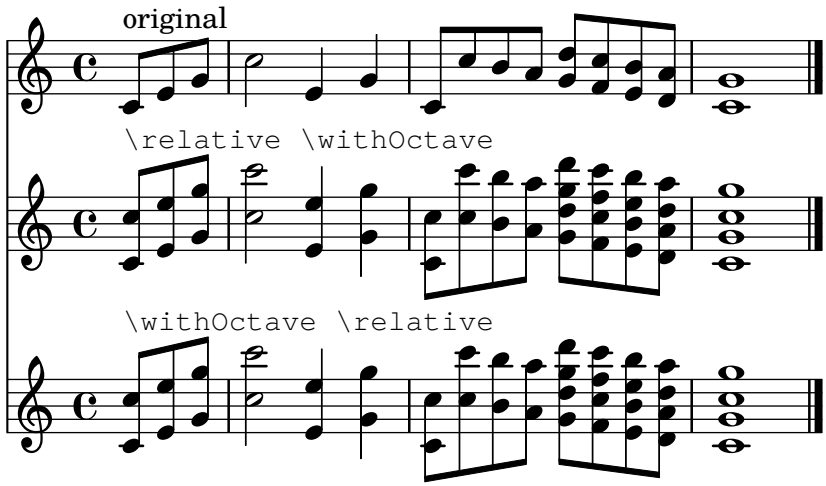
The code here defines a `\reltranspose` function working inside of `\relative` and uses it. Both staves should appear identical.

`make-relative-copies.ly`
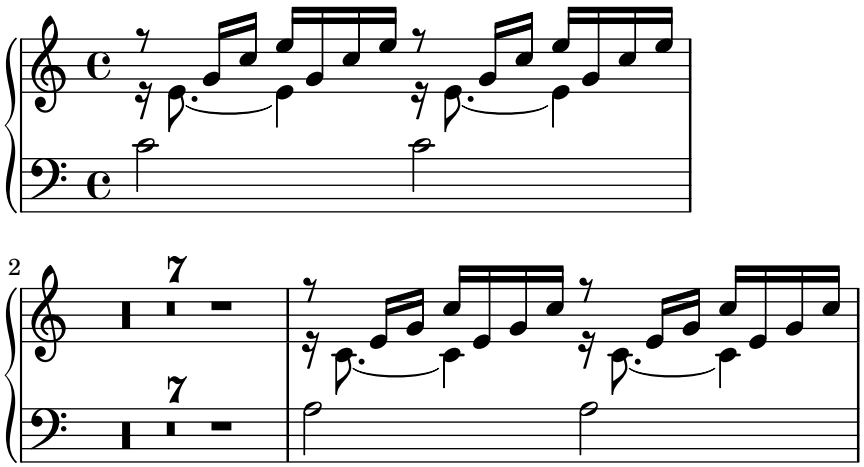


`make-relative` can make relativization on music function calls behave as one would expect from looking at the function's arguments rather than at the actually resulting expressions. This regtest defines an example function `\withOctave` which works equally well inside and outside of `\relative`.
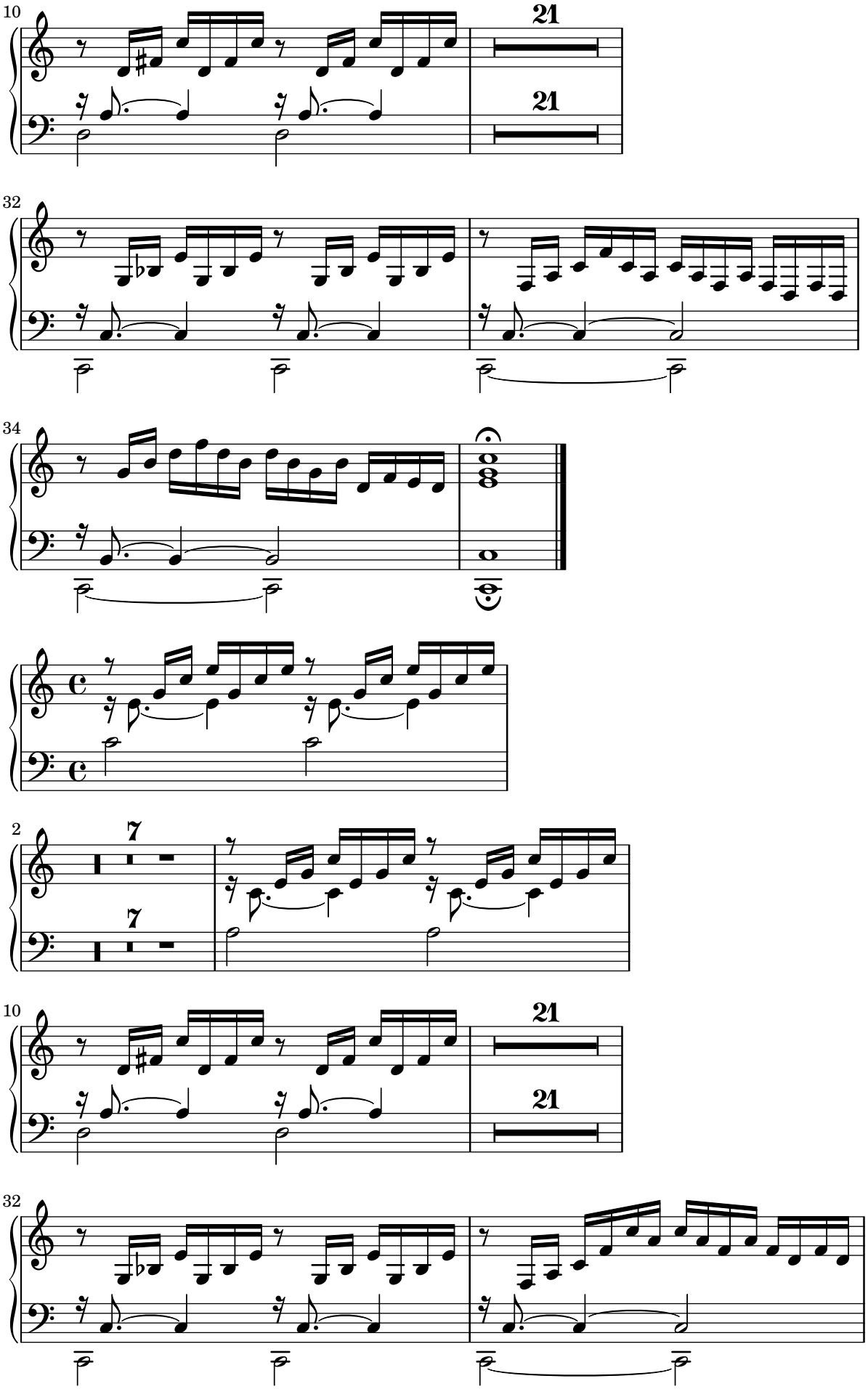
`make-relative-music.ly`

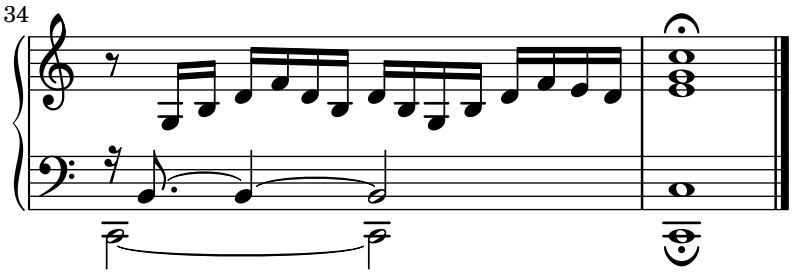


`make-relative` is a Scheme utility macro mainly useful for creating music functions accepting pitches as arguments. Its purpose is to make music functions taking pitch arguments for producing complex music fragments integrate nicely within a `\relative` section. This regtest typesets a short music fragment twice, once without using `\relative`, once using it. The fragment should appear identical in both cases.

`make-relative.ly`

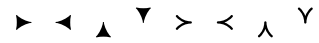



10
21
21
32
34
2
7
7
10
21
21
32



The feta font has arrow heads

`markup-arrows.ly`



The explicit directional embedding codes, U+202A and U+202B, are supported in single-line markup strings. The embeddings must be terminated with the pop directional formatting character, U+202C.

`markup-bidi-explicit-embedding.ly`

אבה אבה "!ABC" אבה אבה!
אבה אבה "ABC!" אבה אבה!

abc def "אבה!" ghi jkl!
abc def "!אבה" ghi jkl!

The explicit directional override codes, U+202D and U+202E, are supported in single-line markup strings. The overrides must be terminated with the pop directional formatting character, U+202C.

`markup-bidi-explicit-overrides.ly`

אבג דהו זחט יךכ
כדי טחז והד גבא

abc def ghi jkl
lkj ihg fed cba

The implicit directional marks, U+200E and U+200F, are supported in single-line markup strings.

`markup-bidi-implicit-marks.ly`

אבה "!ABC" אבה
אבה "ABC!" אבה

abc "אבה!" def
abc "!אבה" def

A single Pango string is processed according to the Unicode Bidirectional Algorithm. The strong Hebrew characters in this example are set right-to-left, and the Latin numerals, space character, and punctuation are set according to the rules of the algorithm.

`markup-bidi-pango.ly`

.לל1ללל, רר2רר

If `\left-brace` or `\right-brace` cannot find a match for the given point size, it should default gracefully to either `brace0` or `brace575` and display a warning.

`markup-brace-warning.ly`

{

The markup command `\left-brace` selects a `fetaBraces` glyph based on point size, using a binary search. `\right-brace` is simply a `\left-brace` rotated 180 degrees.

`markup-braces.ly`



Text markup using `center-column` shall still reserve space for its whole width and not overwrite the previous stencil.

`markup-center-align-nocollision.ly`

XXX + XXX
Y Y
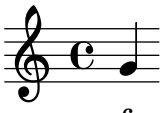
Fixed horizontal alignment of columns of text can be set using \left-column, \center-column and \right-column.

`markup-column-align.ly`

one	one	one
two	two	two
three	three	three

test various markup commands.

`markup-commands.ly`



foo **foo** LOWER **normal** normal Small-Caps SMALL-CAPS
LOWER

justify:
This is a field containing text. Blah blah blah. This is a field containing text. Blah blah blah. This is a field containing text. Blah blah blah. This is a field containing text. Blah blah blah. This is a field containing text. Blah blah blah.

wordwrap:
This is a field containing text. Blah blah blah.
This is a field containing text. Blah blah blah.
This is a field containing text. Blah blah blah.
This is a field containing text. Blah blah blah.
This is a field containing text. Blah blah blah.

draw-line:



underlined
multiple underlines

The `\compound-meter` markup command can produce various kinds of numeric time signature.

`markup-compound-meter.ly`

These are conventional time signatures: $\frac{3}{4}$ $\frac{4}{4}$ (Aren't they pretty?)

This is single-digit compound time signature: $2+3$ (Isn't it pretty?)

This is an unusual time signature: $6.22\frac{e}{1}23+\frac{4}{3}+3.14+9876\frac{+}{0}5432+-1$ (Isn't it pretty?)

Cyclic markup definitions should cause a warning, but not crash LilyPond with an endless loop

`markup-cyclic-reference.ly`

Markups have a maximum depth to prevent non-termination.

`markup-depth-non-terminating.ly`

Test:

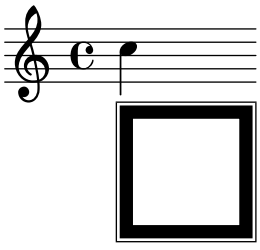
Diacritic marks are rendered and positioned correctly. The diacritic on line 1 looks like a lower-underline and is centered beneath the main character. The diacritic on line 2 is positioned to the left of the main character, with a tiny space of separation. The diacritic on line 3 is positioned directly above the main character, either centered or shifted slightly to the left.

`markup-diacritic-marks.ly`

בּ
וּ
וֹ
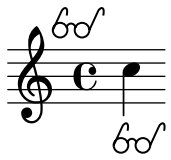
The epsfile markup command reads an EPS file

`markup-eps.ly`



The eyeglasses markup function prints out eyeglasses.

`markup-eyeglasses.ly`
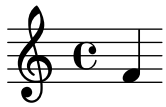


The markup command `\first-visible` uses the first argument that produces a non-empty stencil and ignores the rest.

The expected markup on this score is "Lame Songs for Testing" followed by a "C" time signature symbol.

`markup-first-visible.ly`

Lame Songs for Testing **C**



No elements:

One element (expect 111): 111

Single markup list (expect aaa): aaa

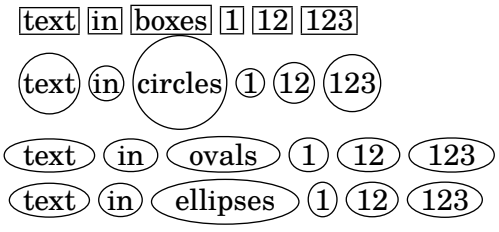
Multiple markup lists (expect ccc): ccc

Mixed markup and markup lists (expect fff): fff

Nested markup lists (expect jjj): jjj

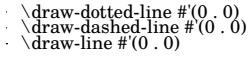
Text is framed properly with `\box`, `\circle`, `\oval` and `\ellipse`

`markup-frame-text.ly`



The markup-commands `\draw-dashed-line`, `\draw-dotted-line` and `\draw-squiggle-line` should print the same visual length as `\draw-line`. Also testing possible overrides for `\draw-squiggle-line`

`markup-line-styles.ly`



\draw-dotted-line #'(0.75 . 0)
\draw-dashed-line #'(0.75 . 0)
\draw-line #'(0.75 . 0)

\draw-dotted-line #'(1.5 . 0)
\draw-dashed-line #'(1.5 . 0)
\draw-line #'(1.5 . 0)

\draw-dotted-line #'(2.25 . 0)
\draw-dashed-line #'(2.25 . 0)
\draw-line #'(2.25 . 0)

\draw-dotted-line #'(3.0 . 0)
\draw-dashed-line #'(3.0 . 0)
\draw-line #'(3.0 . 0)

\draw-dotted-line #'(3.75 . 0)
\draw-dashed-line #'(3.75 . 0)
\draw-line #'(3.75 . 0)

\draw-dotted-line #'(4.5 . 0)
\draw-dashed-line #'(4.5 . 0)
\draw-line #'(4.5 . 0)

\draw-dotted-line #'(5.25 . 0)
\draw-dashed-line #'(5.25 . 0)
\draw-line #'(5.25 . 0)

\draw-dotted-line #'(6.0 . 0)
\draw-dashed-line #'(6.0 . 0)
\draw-line #'(6.0 . 0)

\draw-dotted-line #'(6.75 . 0)
\draw-dashed-line #'(6.75 . 0)
\draw-line #'(6.75 . 0)

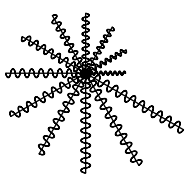
\draw-dotted-line #'(7.5 . 0)
\draw-dashed-line #'(7.5 . 0)
\draw-line #'(7.5 . 0)

\draw-dotted-line #'(8.25 . 0)
\draw-dashed-line #'(8.25 . 0)
\draw-line #'(8.25 . 0)

\draw-dotted-line #'(9.0 . 0)
\draw-dashed-line #'(9.0 . 0)
\draw-line #'(9.0 . 0)

\draw-dotted-line #'(9.75 . 0)
\draw-dashed-line #'(9.75 . 0)
\draw-line #'(9.75 . 0)

\draw-dotted-line #'(10.5 . 0)
\draw-dashed-line #'(10.5 . 0)
\draw-line #'(10.5 . 0)



default

different orientation

"eq-end?" set #f
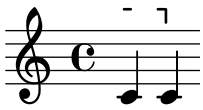
different height

different thickness

different angularity

The thickness setting between markup lines and other lines is consistent.

`markup-line-thickness.ly`



Text that can spread over pages is entered with the `\markuplist` command. It can be assigned to a variable and inserted at top-level with or without preceding it by `\markuplist`.

`markup-lines-identifier.ly`

Lorem ipsum dolor sit amet, consectetur adipisici elit,

sed eiusmod tempor incidunt ut labore et dolore

magna aliqua. ...

Lorem ipsum dolor sit amet, consectetur adipisici elit,

sed eiusmod tempor incidunt ut labore et dolore

magna aliqua. ...

Text that can spread over pages is entered with the `\markuplist` command. Widowed and orphaned lines are avoided at the begininng and end of a `\markuplist` containing more than one line.

`markup-lines.ly`

Il y avait en Westphalie, dans le château de M. le baron de Thunder-ten-tronckh, un jeune garçon à qui la nature avait donné les mœurs les plus douces. Sa physionomie annonçait son âme. Il avait le jugement assez droit, avec l'esprit le plus simple ; c'est, je crois, pour cette raison qu'on le nommait Candide. Les anciens domestiques de la maison soupçonnaient qu'il était fils de la sœur de monsieur le baron et d'un bon et honnête gentilhomme du voisinage, que cette demoiselle ne voulut jamais épouser parce qu'il n'avait pu prouver que soixante et onze quartiers, et que le reste de son

2
arbre généalogique avait été perdu par l'injure du temps. (not orphaned)
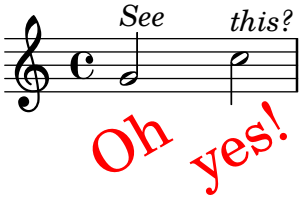
Monsieur le baron était un des plus puissants seigneurs de la Westphalie, car son château avait une porte et des fenêtres. Sa grande salle même était ornée d'une tapisserie. Tous les chiens de ses basses-cours composaient une meute dans le besoin ; ses palefreniers étaient ses piqueurs; le vicaire du village était son grand-aumônier. Ils l'appelaient tous monseigneur, et ils riaient quand il faisait des contes.

3

Madame la ... (may be orphaned)
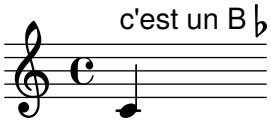
`\markupMap` can be used for applying a markup function to music properties throughout a music expressions, like the `text` of all contained lyric events.

`markup-map.ly`
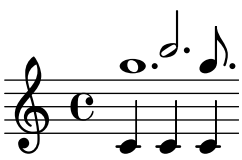


Reset fontname for musicglyph. For unknown glyphs, we print a warning.

`markup-music-glyph.ly`



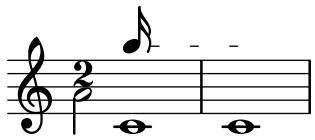
A dotted whole note displayed via the `\note` command must separate the note head and the dot. The dot avoids the upflag.

`markup-note-dot.ly`



The `'style` property from grobs such as `TimeSignature` and `TextSpanner` does not affect the default note head style for `\note` and `\note-by-number`.

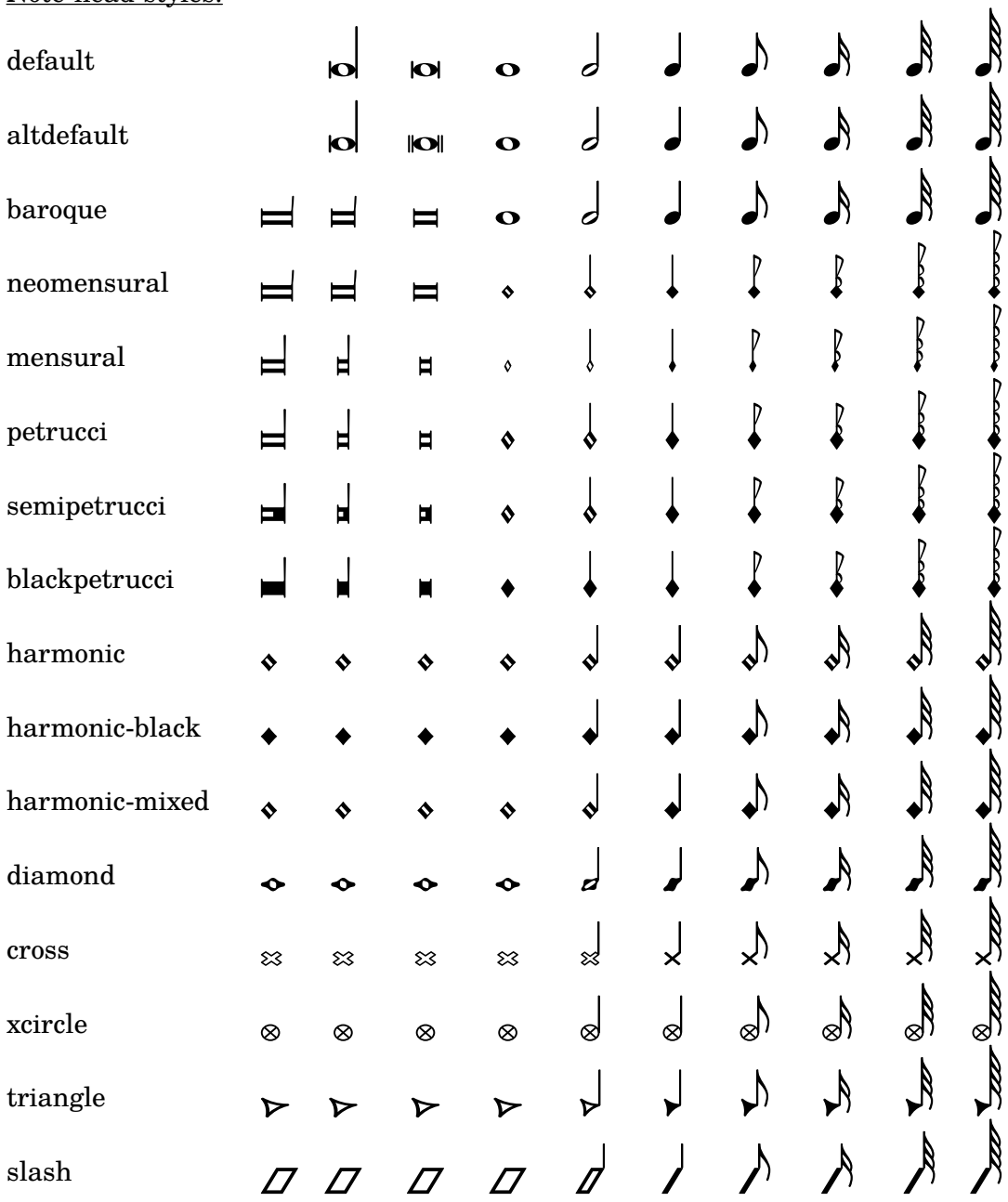
`markup-note-grob-style.ly`



`\note-by-number` and `\note` support all note head styles and all flag styles (default, straight, flat, mensural).

`markup-note-styles.ly`

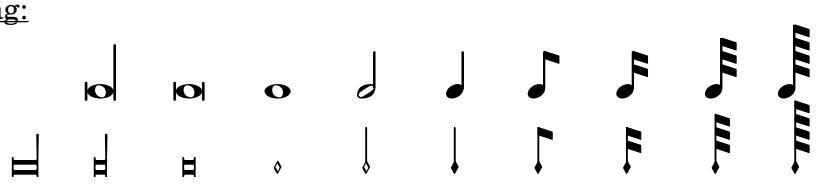
<u>Note-head-styles:</u>



Modern-straight-flag:

default

mensural



Old-straight-flag:

default



Flat-flag:

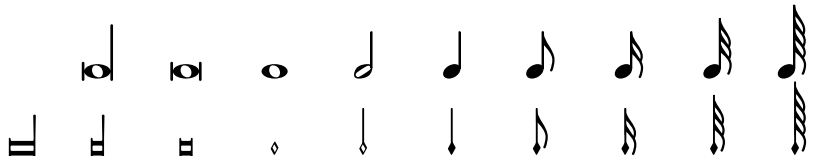
default



default-flag:
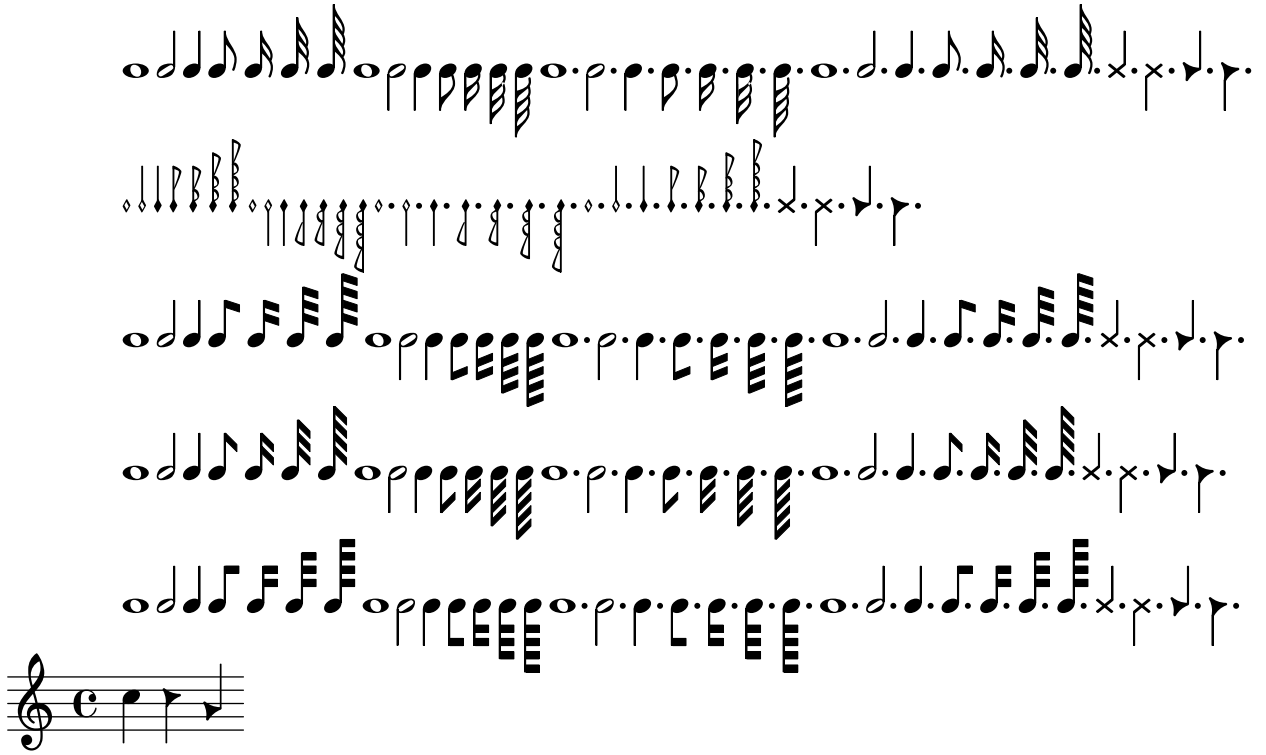
default

mensural



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

`markup-note.ly`



Partial markups acts as a chain of markup commands where everything but some arguments of the final markup command has already been supplied.

`markup-partial.ly`

Bold red.

Bold

red

in

a

list.

Italic green.

Italic

green

in

a

list.

3/8: 

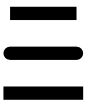
The `\path` markup command supports the `filled` property to toggle its fill.

`markup-path-fill.ly`



The `\path` markup command supports the `line-cap-style` property with values of `butt`, `round`, and `square`.

`markup-path-linecap.ly`



The `\path` markup command supports the `line-join-style` property with values of `bevel`, `round`, and `miter`.

`markup-path-linejoin.ly`



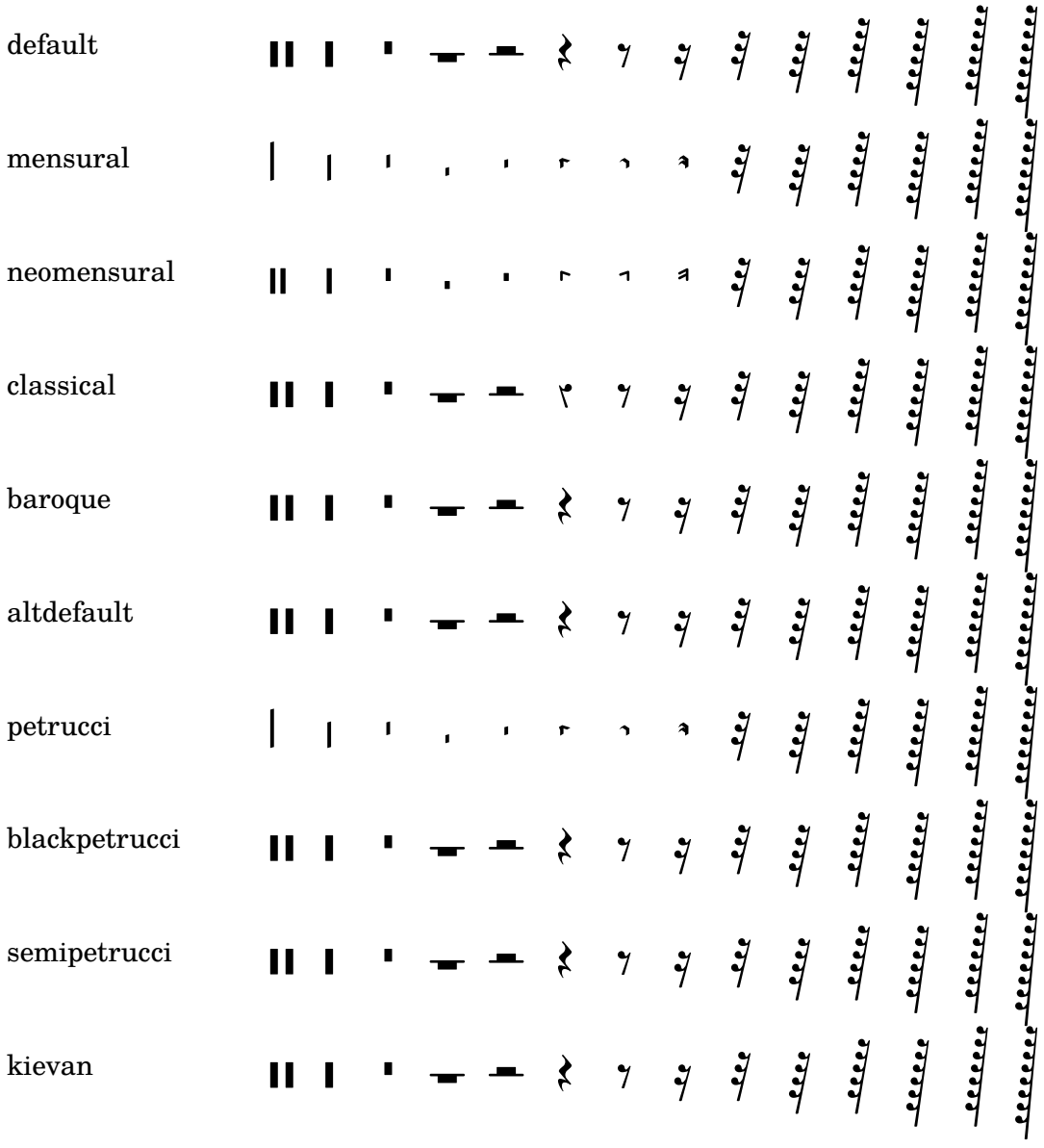
The `\path` markup command allows the user to draw arbitrary paths using a simple syntax. The two paths below should be identical.

`markup-path.ly`



`\rest-by-number` and `\rest` support all rest styles.

`markup-rest-styles.ly`

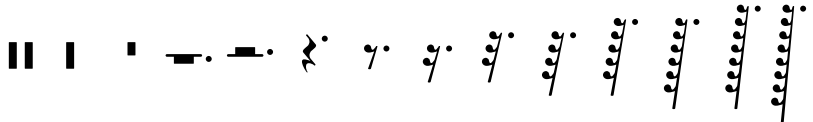


The rest markup function works for a variety of style, dot and duration settings.

`markup-rest.ly`

Simple Rests

default



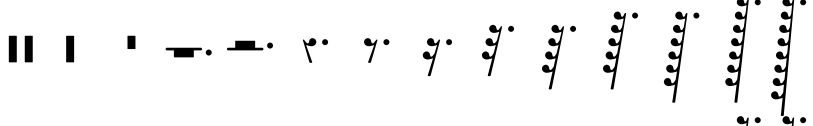
mensural



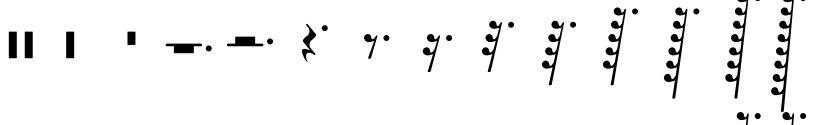
neomensural



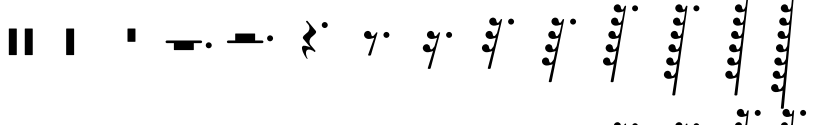
classical



baroque



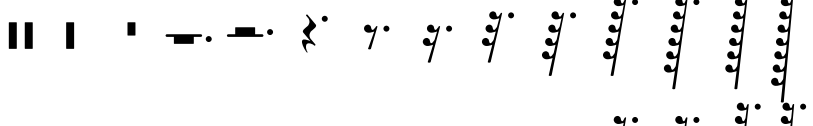
altdefault



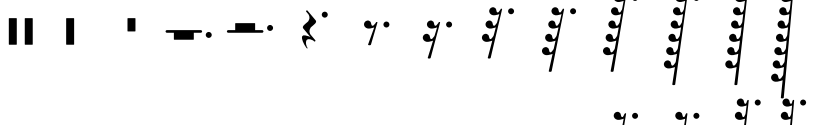
petrucci



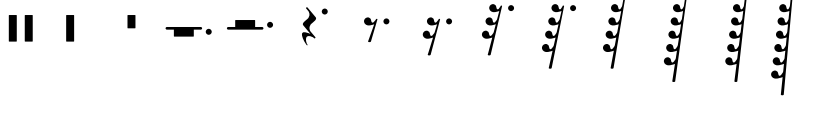
blackpetrucci



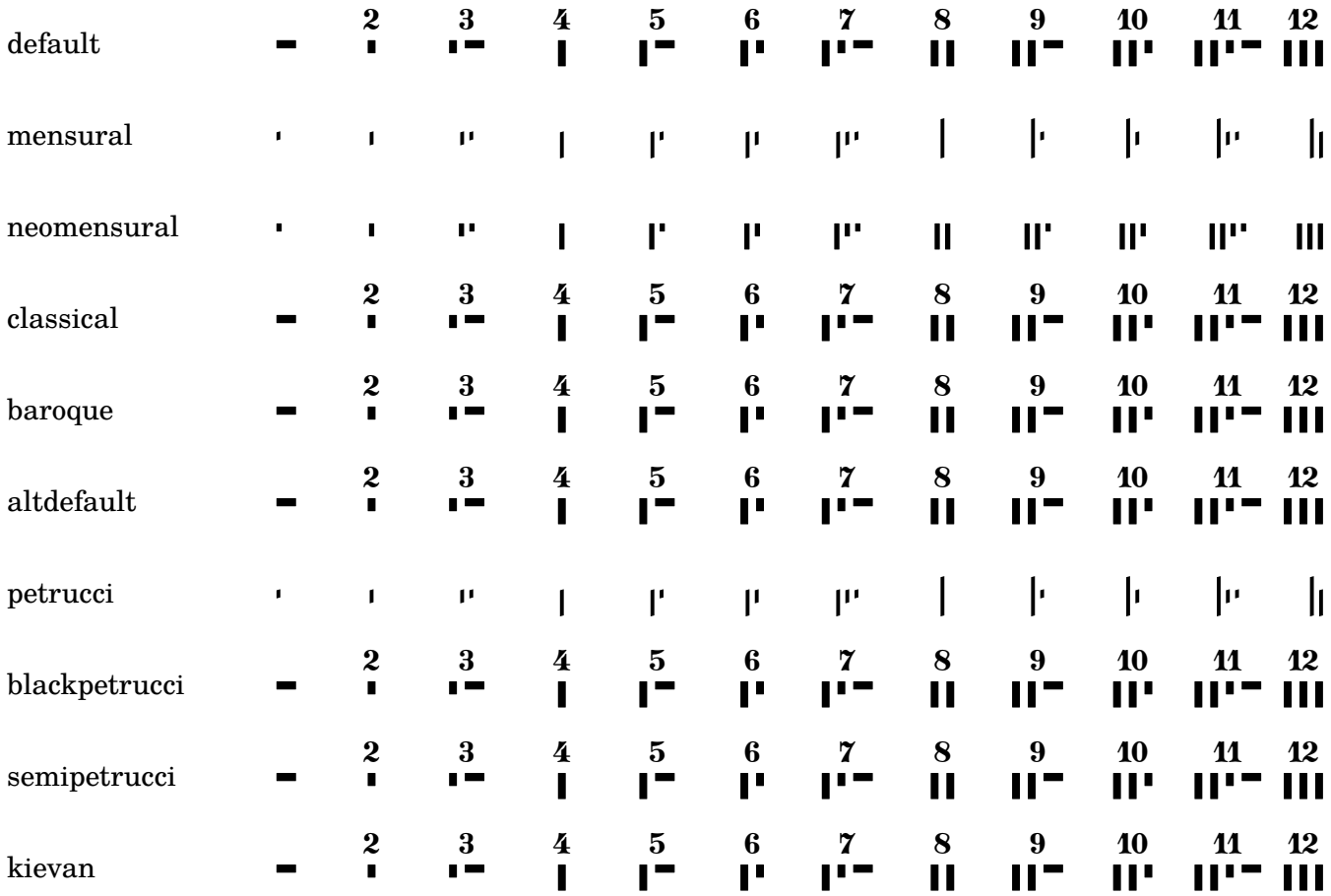
semipetrucci



kievan

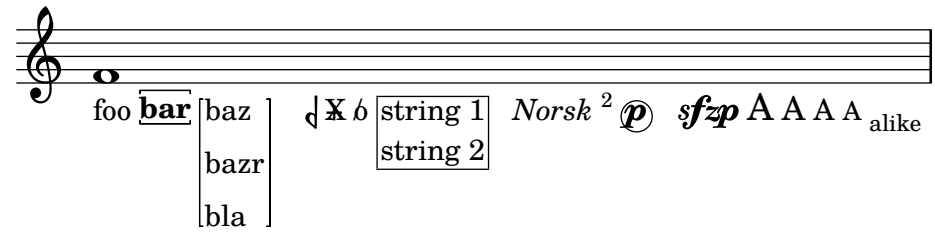


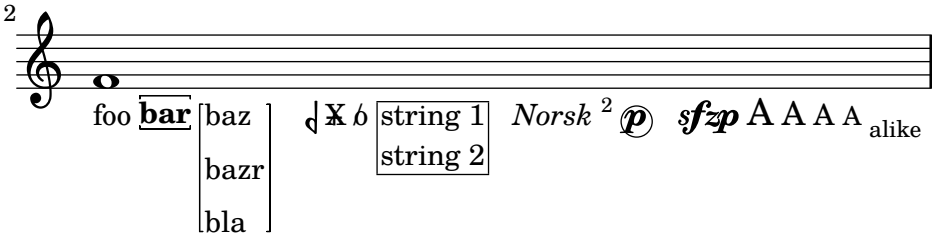
MultiMeasureRests



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

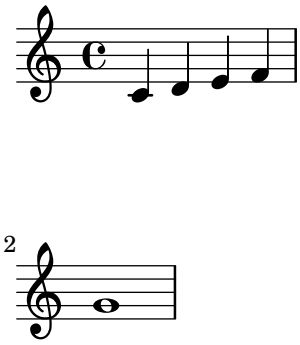
`markup-scheme.ly`





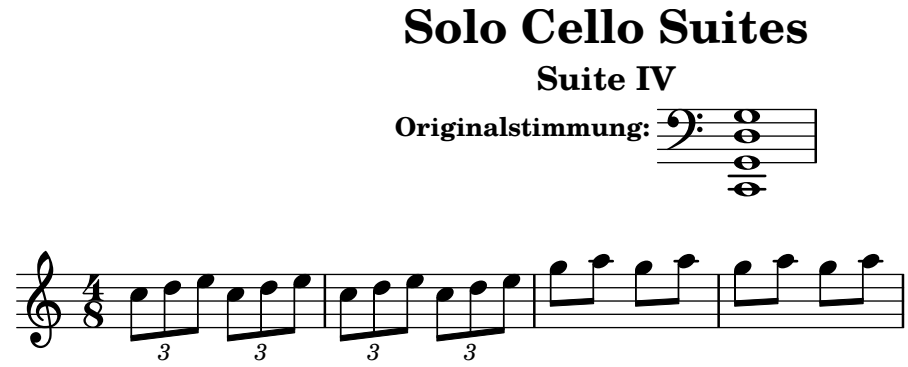
\markup \score displays all systems. Spacing between systems is set using `baseline-skip`.

markup-score-multi-system.ly



Use \score block as markup command.

markup-score.ly



A list of special character ASCII aliases can be easily included. This works for markups and lyrics.

markup-special-characters.ly

Markup example:

Input:

```
&numero;2 &ndash; &OE;dipe&hellip;
```

Output:

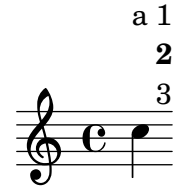
№2 – Œdipe…

Lyric example:

Ceſſez Infidèles, un cœur innocent ne craint rien ;

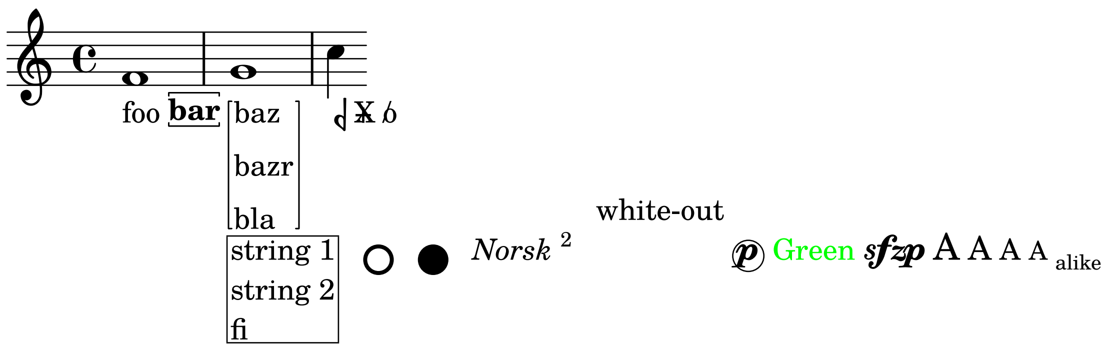
Markup scripts may be stacked.

markup-stack.ly
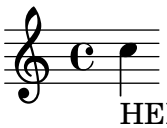


Demo of markup texts, using LilyPond syntax.

markup-syntax.ly



Users may define non-standard markup commands using the `define-markup-command` scheme macro.

markup-user.ly



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

markup-word-wrap.ly

this is normal text This is a test of the wordwrapping function. 1 This is a test continuing 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*

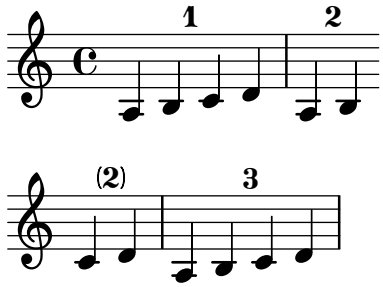
this is normal text This is a test of the wordwrapping continuing function, but with justification. 1 This is a test of the wordwrapping function, but with justification. 2 This is a test of $\frac{a}{b}$ the wordwrapping function, but with justification. 3 This is a test of the wordwrapping function,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

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.

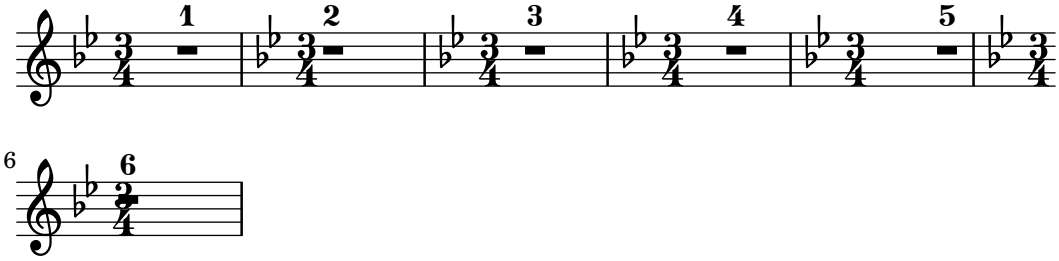
Measures split across line breaks may be numbered in a measure count. Each segment receives a number. The first number has its ordinary appearance, but numbers after the break are enclosed in parentheses.

`measure-counter-broken.ly`



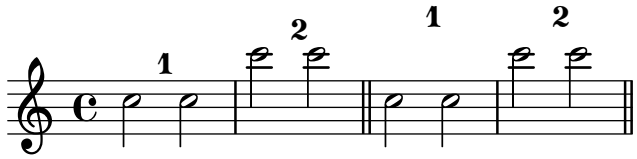
The `spacing-pair` property may be used to adjust the horizontal positioning of `MeasureCounter` objects relative to prefatory material. In the following example, the count should be aligned with the full-measure rests.

`measure-counter-spacing-pair.ly`



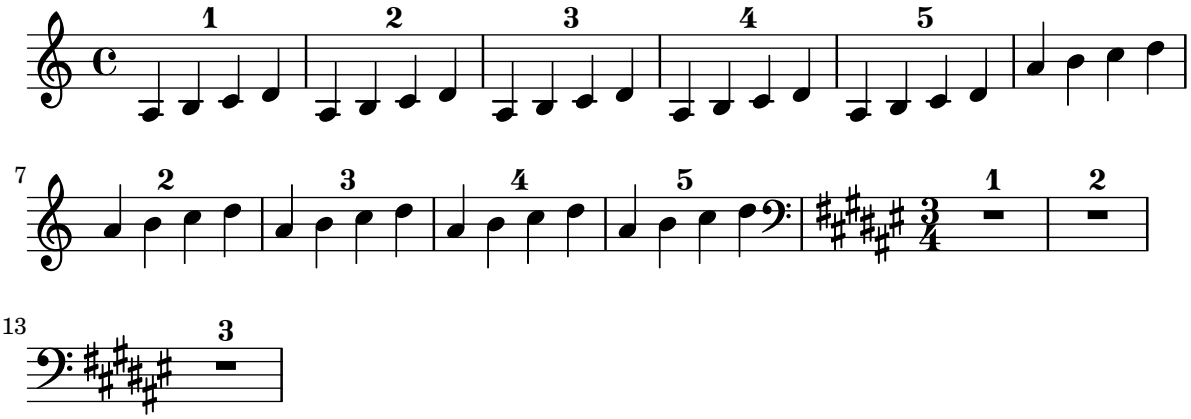
The `staff-padding` property may be used to adjust the distance of `MeasureCounter` objects from the staff. The following example uses `staff-padding` to align the count vertically.

`measure-counter-staff-padding.ly`



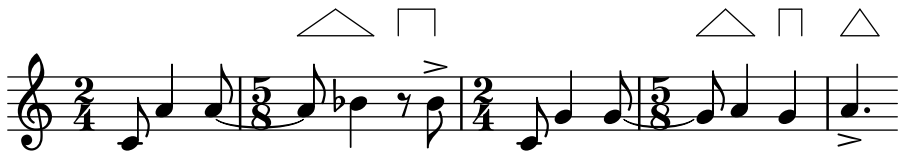
Measures can be numbered sequentially by enclosing them with `\startMeasureCount` and `\stopMeasureCount`.

`measure-counter.ly`



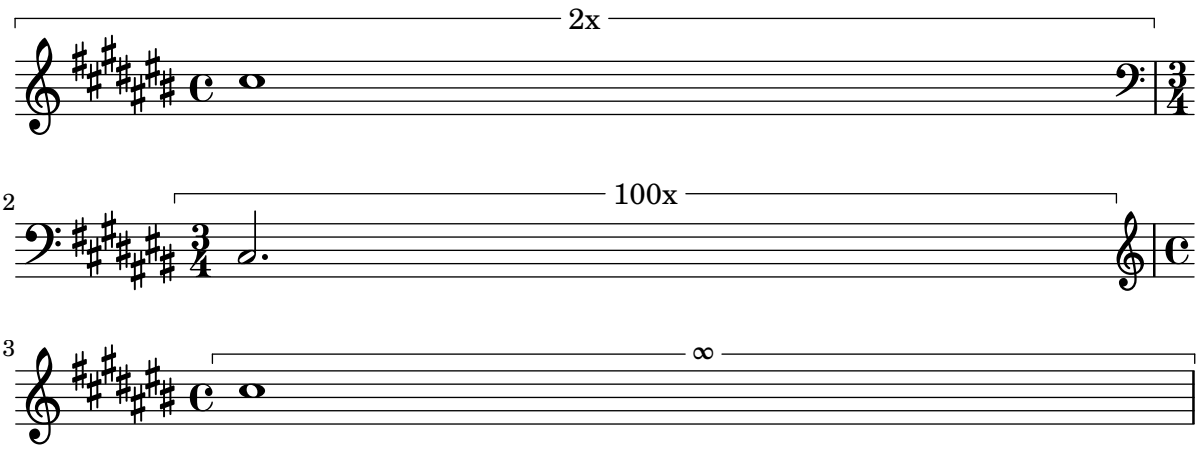
The `Measure_grouping_engraver` adds triangles and brackets above beats when the beats of a time signature are grouped.

`measure-grouping.ly`



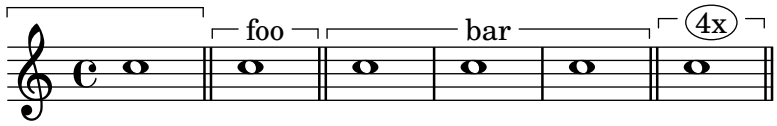
The ends of measure spanners may be aligned in various ways.

`measure-spanner-spacing-pair.ly`



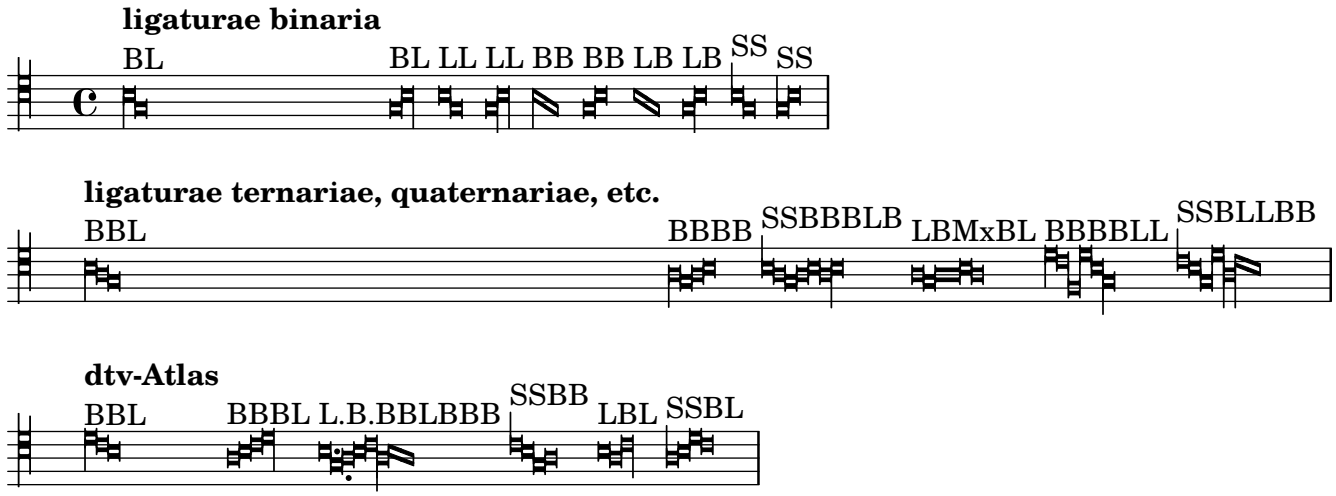
Measure spanners can span single and multiple measures. They may be texted or untexted and hold markups.

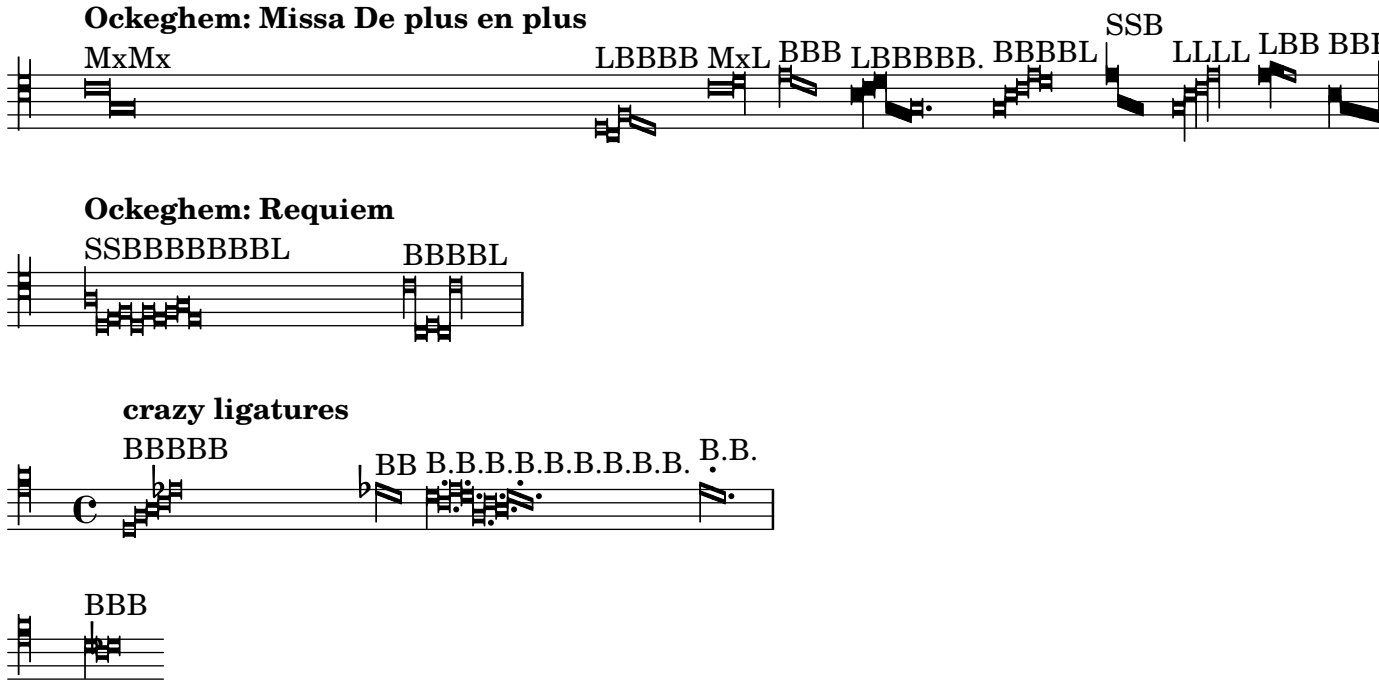
`measure-spanner.ly`



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

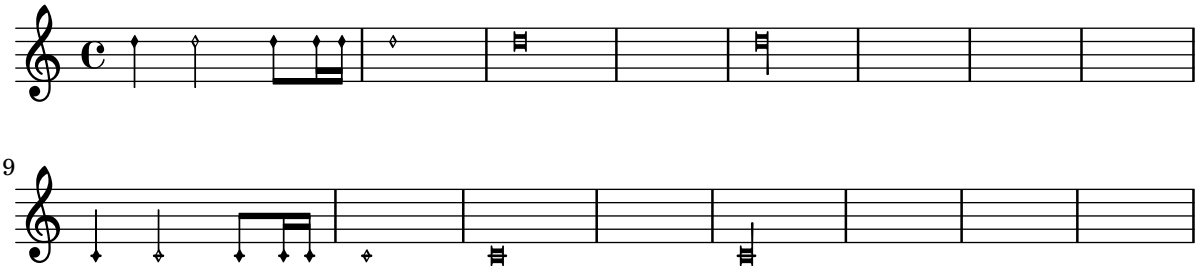
`mensural-ligatures.ly`





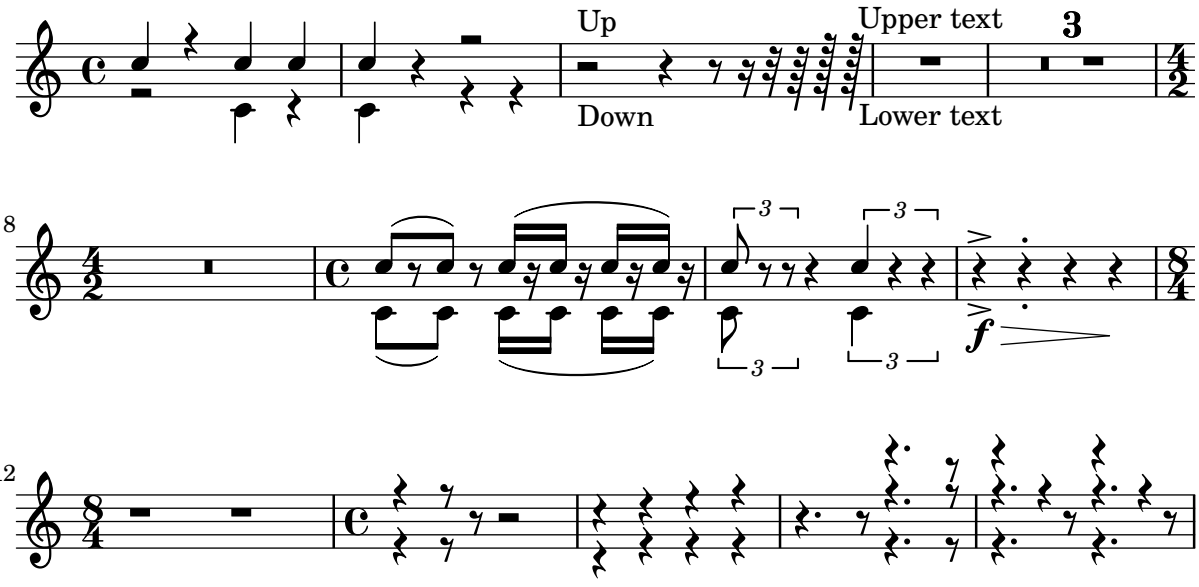
There is limited support for mensural notation: note head shapes are available. Mensural stems are centered on the note heads, both for up and down stems.

`mensural.ly`
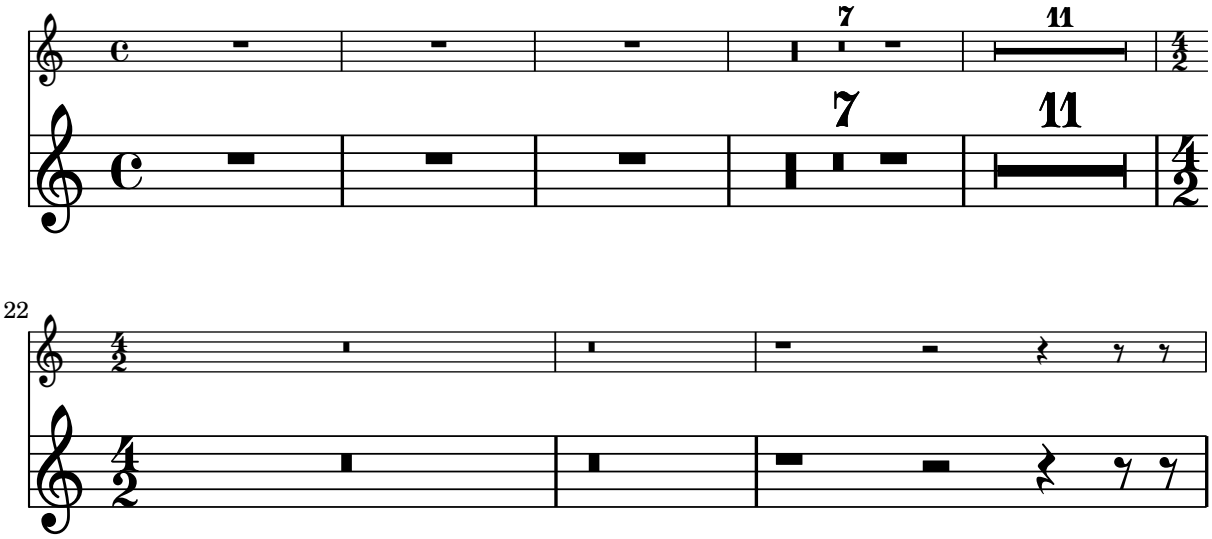


Test for merging rests in different voices.

`merge-rests-engraver.ly`



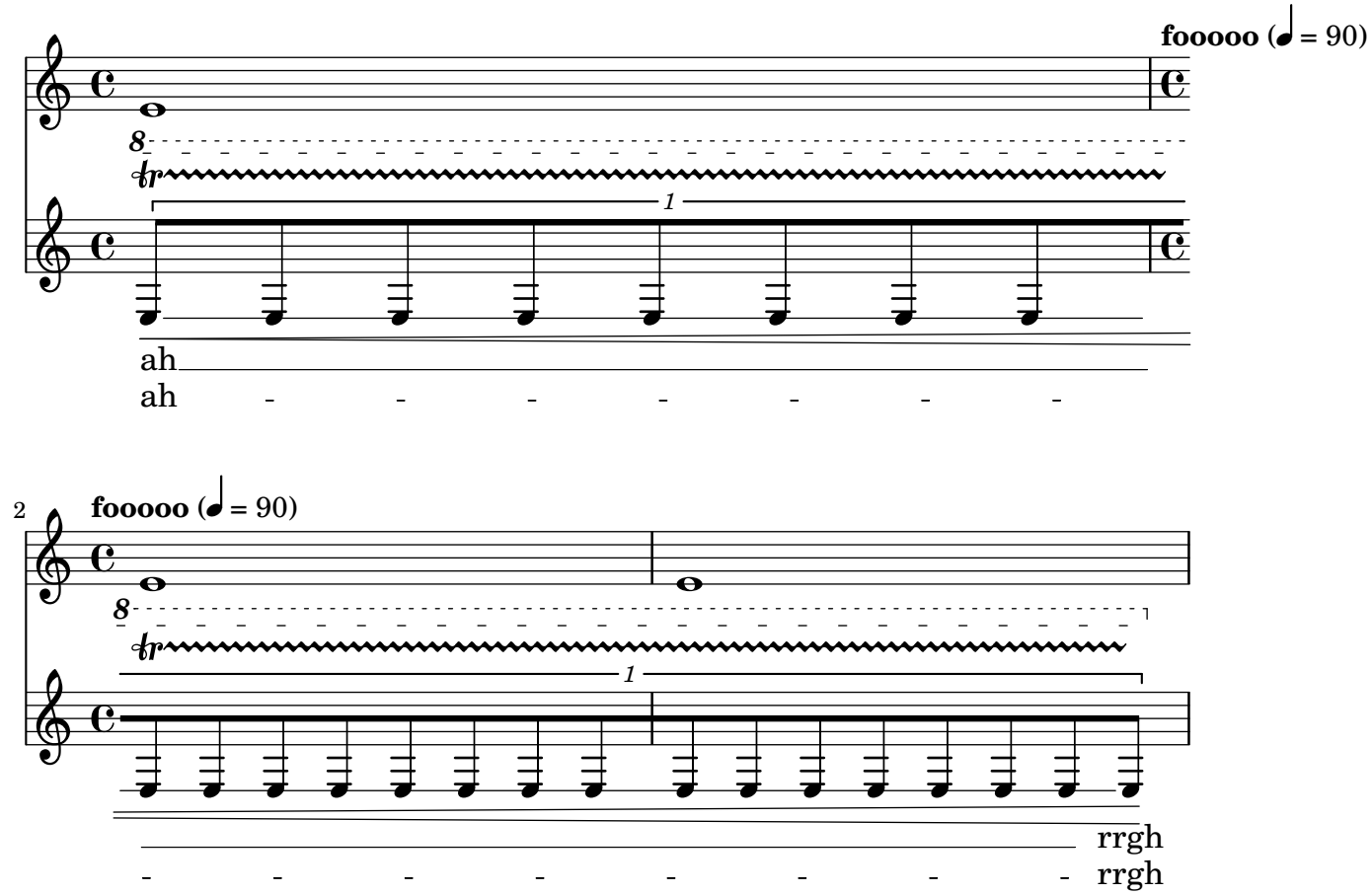
Test for vertical positions of merged rests in magnified staves.

merge-rests-magnify-staff.ly
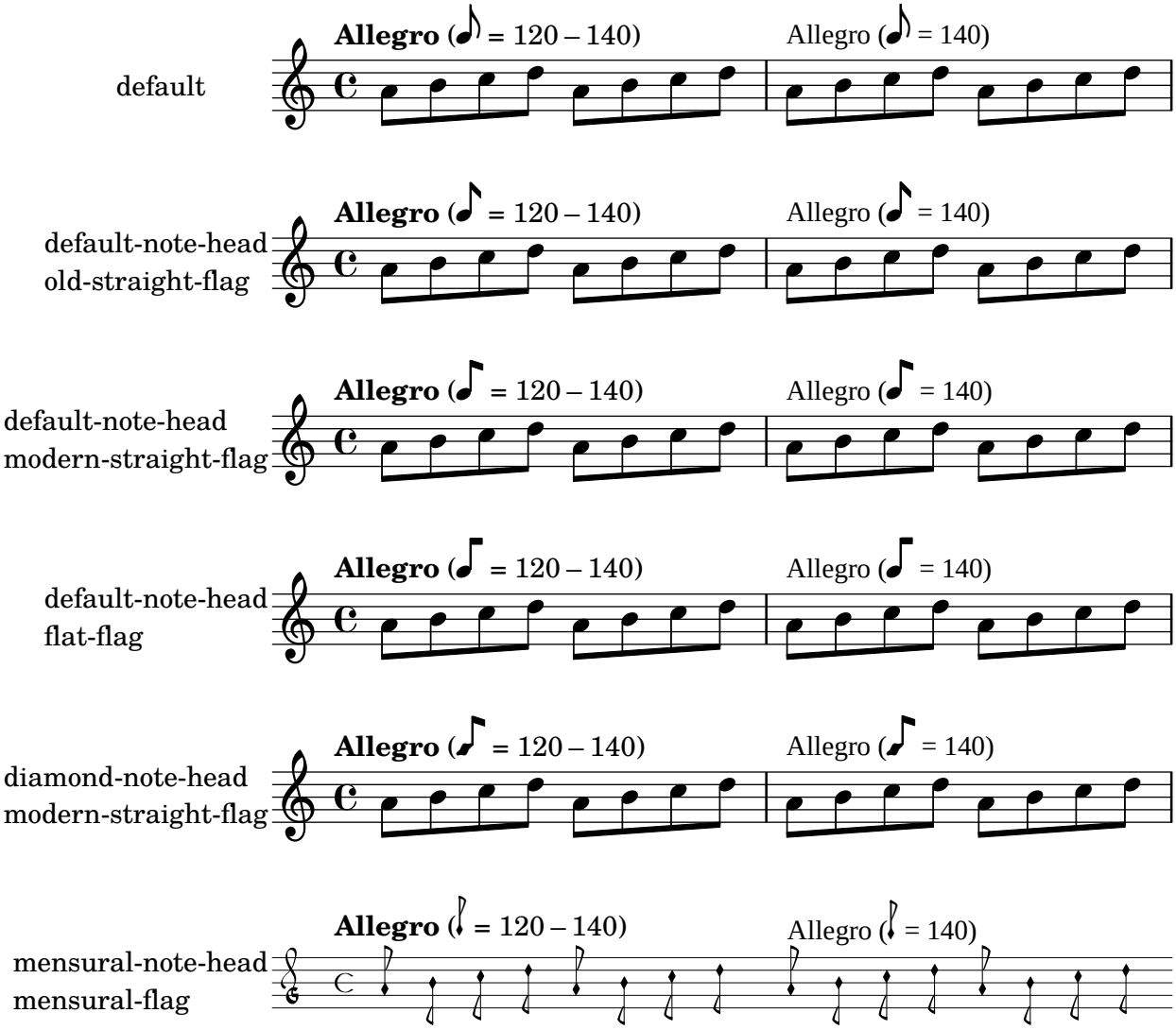


A `MetronomeMark`, `RehearsalMark` and `BarNumber` should not effect the starting point of spanners.

metronome-mark-broken-bound.ly



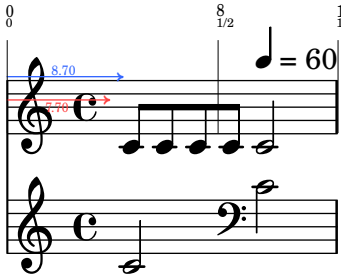
`metronomeMarkFormatter` supports all note head styles and flags styles. Setting `font-name` for `MetronomeMark` does not disturb the glyphs for note-head and flag.

`metronome-mark-formatter.ly`



Metronome marks aligned on notes do not interfere with the positioning of loose columns in other staves. Here the loose column supporting the clef is correctly placed immediately before the second note in the lower staff.

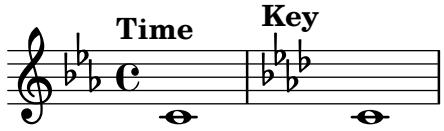
`metronome-mark-loose-column.ly`



Metronome marks respect symbol order in `break-align-symbols`.

In this example, the default is changed to `'(time-signature key-signature)`: since `key-signature` is second in the list, the mark should only be aligned with the key signature if there is no time signature present, as in the second measure.

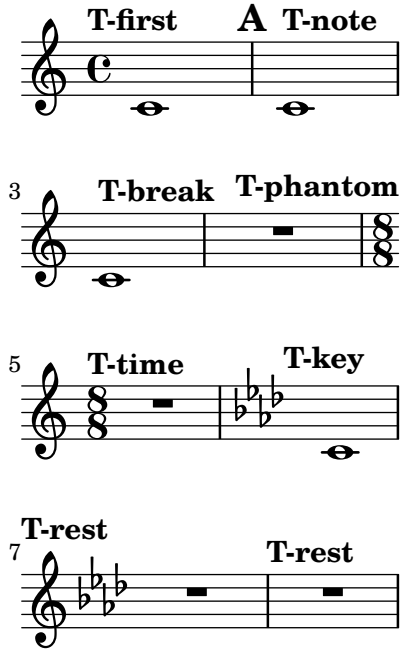
metronome-marking-align-order.ly



\tempo marks are aligned with the time signature or the position of the first note.

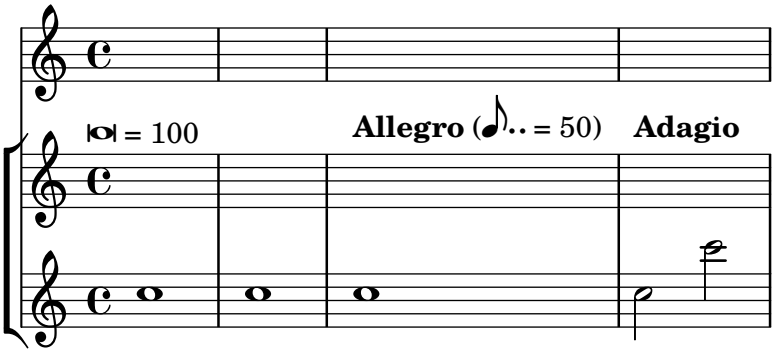
By overriding break-align-symbols the default alignment can be changed. If no symbol in break-align-symbols is present, the property non-break-align-symbols determines the alignment. If the alignment object is a multi-measure rest, the tempo mark is aligned with the preceding bar line.

metronome-marking-break-align.ly



Metronome marks are placed correctly if Metronome_mark_engraver is moved to StaffGroup context. Metronome marks should appear above the middle staff (the upper staff of the group) only.
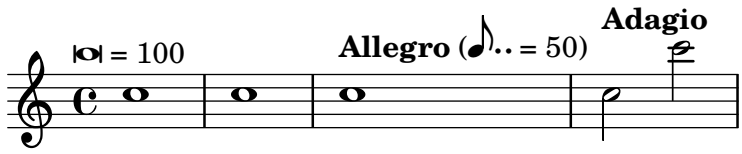
metronome-marking-staff-group-context.ly



Here \tempo directives are printed as metronome markings.

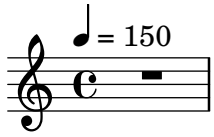
The marking is left aligned with the time signature, if there is one.

`metronome-marking.ly`



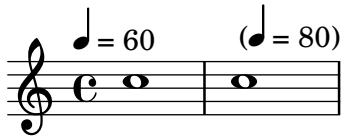
A metronome marking can be added to a multimeasure rest whose engraver was moved to the Staff, without segfaulting.

`metronome-multimeasure-rest-no-segfault.ly`



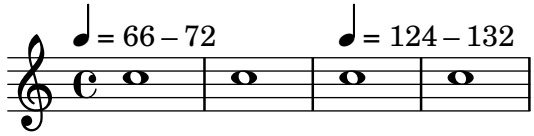
Using an empty text in the metronome marks, one can generate parenthesized tempo marks.

`metronome-parenthesized.ly`



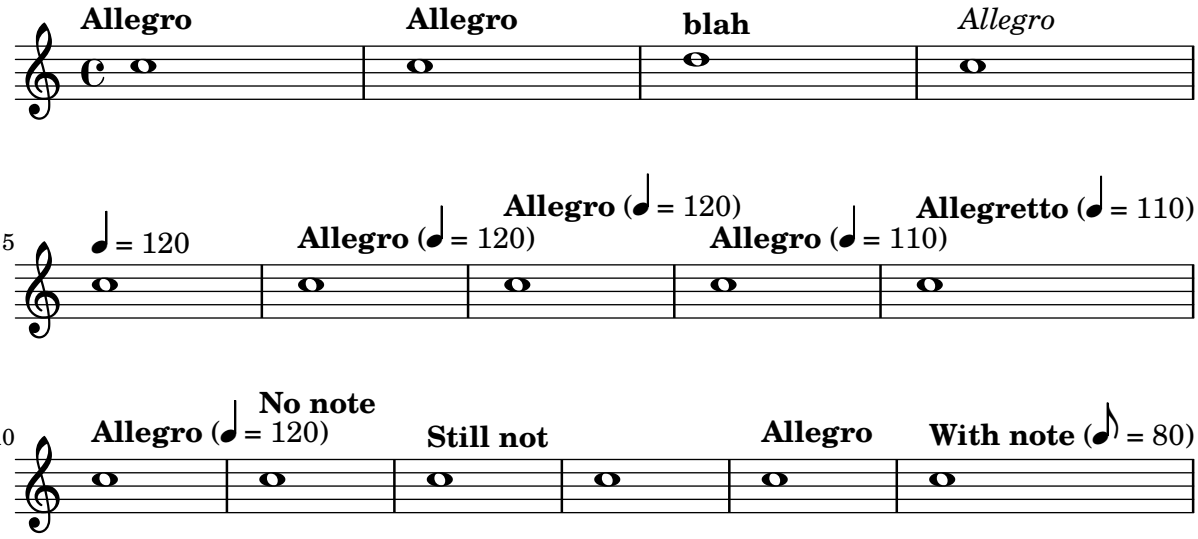
Tempo ranges are supported. By default, numbers are printed with an en-dash character, separated by thin-spaces.

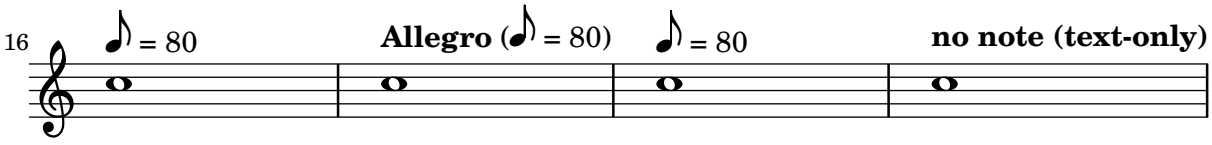
`metronome-range.ly`



The tempo command supports text markup and/or 'duration=count'. Using `Score.tempoHideNote`, one can hide the 'duration=count' in the tempo mark.

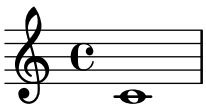
`metronome-text.ly`





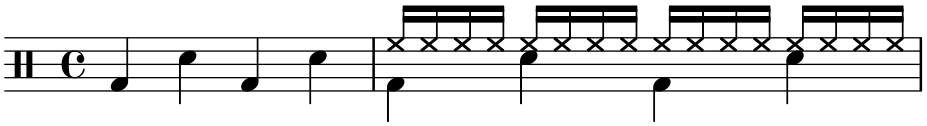
If `after-writing` is set in the `\midi` block, it is called after every MIDI file that is written. The visual and MIDI output are not important in this test.

midi-after-writing.ly



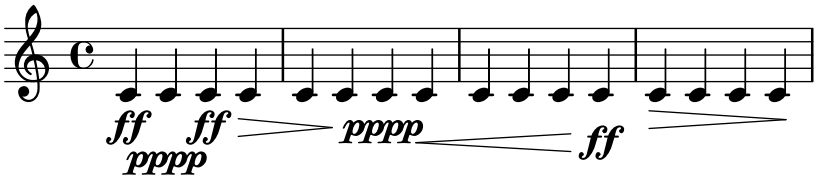
Midi can create drums.

`midi-drums.ly`



Midi also handles crescendo and decrescendo, either starting and ending from specified or unspecified sound level.

`midi-dynamics.ly`



Grace notes shorten previous notes only if they'd overlap them. The A should be a full quarter note, but the C should be shortened to 1/4 - 9/40 * 1/8 = 71/320 (rounded down to 340/384 in MIDI).

`midi-grace-after-rest.ly`

Tied notes sound as one note in MIDI. Grace notes following a tied note shorten the resulting single note in MIDI.
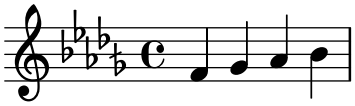
`midi-grace-after-tie.ly`

Grace notes don't intrroduce syncing problems: the last note off will appear at tick 768 (2 * 384).

`midi-grace.ly`

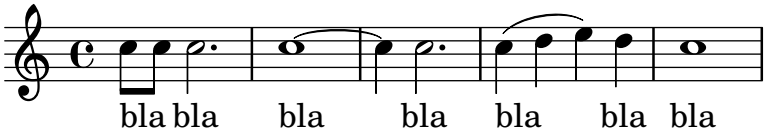
MIDI key signatures are output, using an approximate key signature if MIDI format cannot represent the true key signature

`midi-key-signature.ly`



Lyrics in MIDI are aligned to ties and beams: this examples causes no bar checks in MIDI.

`midi-lyric-barcheck.ly`



Microtonal shifts should be corrected before the start of the next (possibly grace) note.
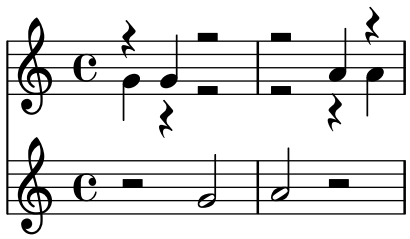
`midi-microtone-off.ly`

The pitch wheel is used for microtones.

`midi-microtone.ly`

A MIDI note-off event precedes a simultaneous note-on event for the same pitch in the same MIDI channel, so that all notes are heard. Run `timidity -idvvv file.midi |grep Midi` to see midi events.

`midi-notes.ly`
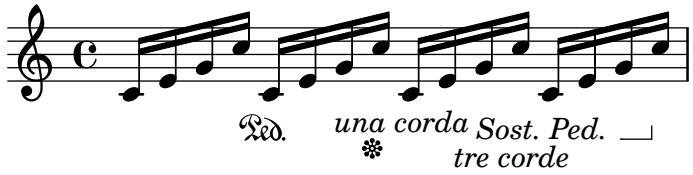


MIDI and partial measures work together.

`midi-partial.ly`
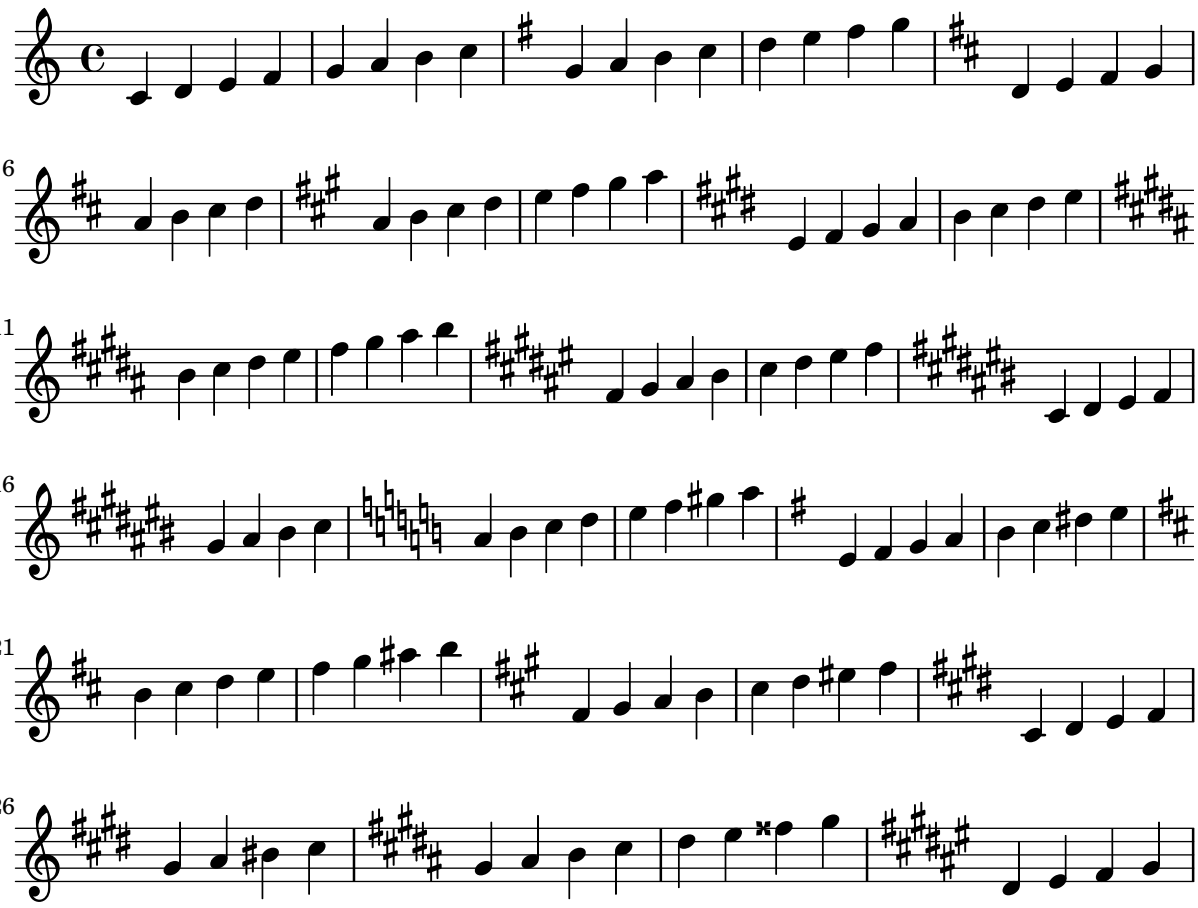
Pedals. Run `timidity -idvvv file.midi |grep Midi` to see midi events.

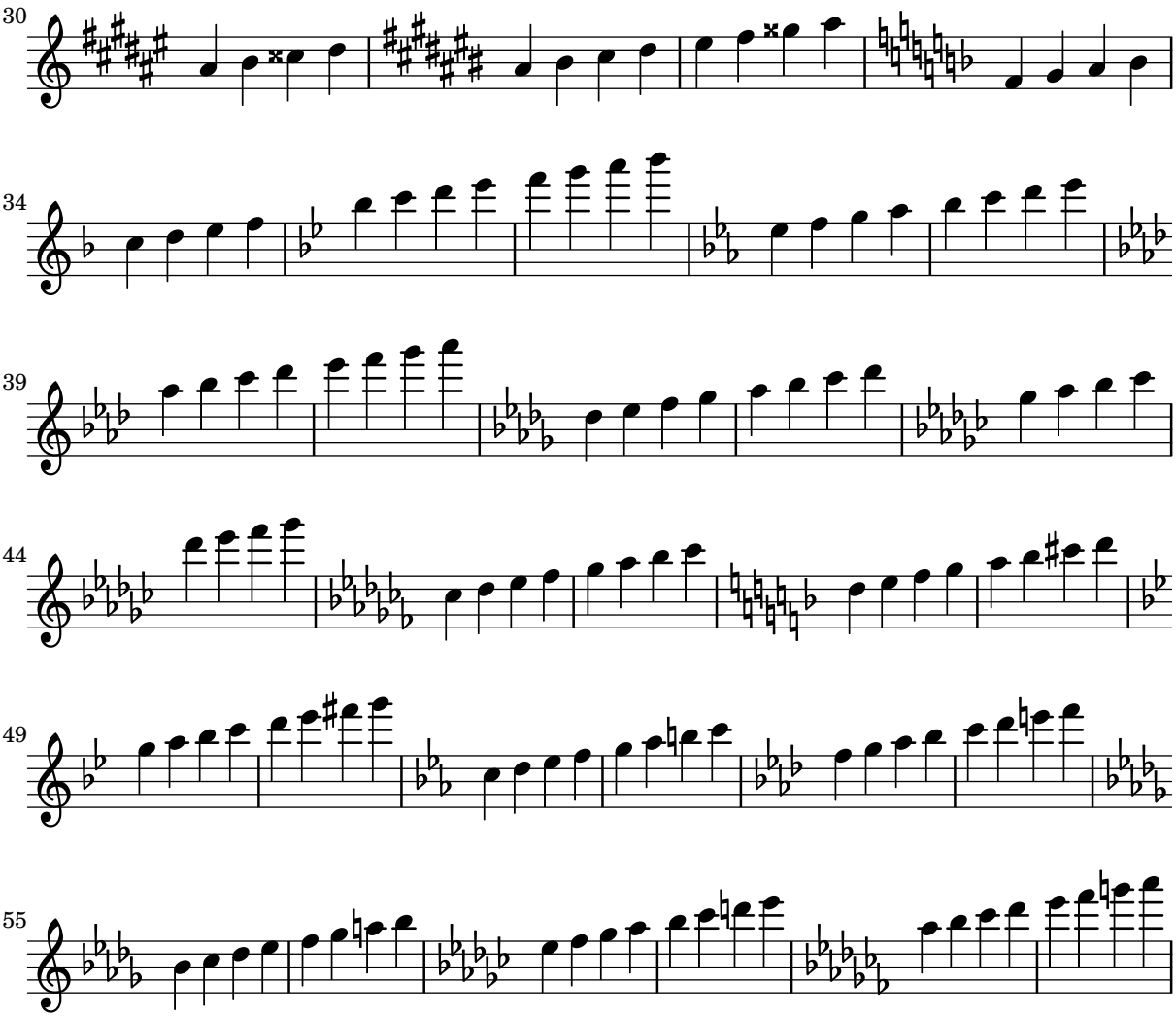
`midi-pedal.ly`



Converting LilyPond input to MIDI and then again back with `midi2ly.py` is a reversible procedure in some simple cases, which mean that the original `.ly` -file and the one converted back from the generated `.midi` -file do not differ. Here are produced some scales.

`midi-scales.ly`





should deliver f' in MIDI

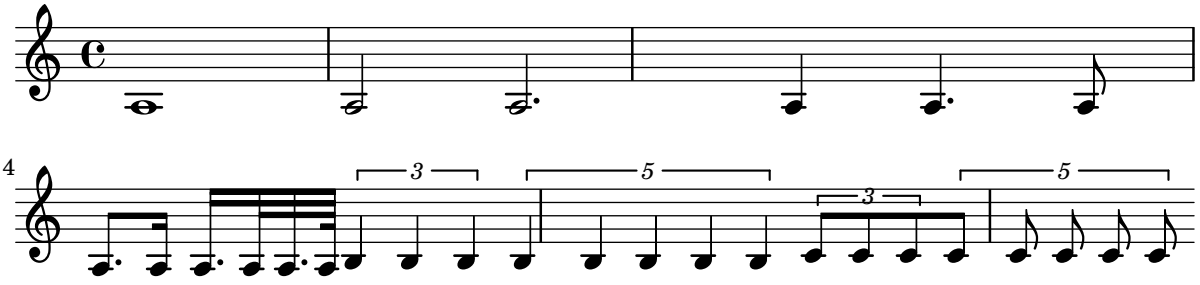
`midi-transposition.ly`



Midi2ly tuplet test.
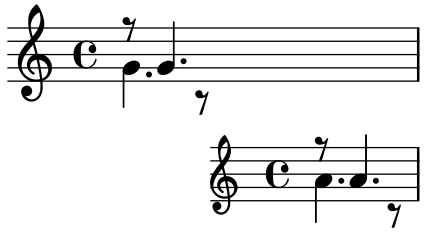
```
python scripts/midi2ly.py --duration-quant=32 \
    --allow-tuplet=4*2/3 \
    --allow-tuplet=8*2/3 \
    --allow-tuplet=4*3/5 \
    --allow-tuplet=8*3/5 \
    tu.midi
```

`midi-tuplets.ly`



In overlapping unisons, within a single MIDI channel, either the first note is truncated, or the notes are merged if `midiMergeUnisons` is `#t`. Run `timidity -idvvv file.midi |grep Midi` to see midi events.

`midi-unisons.ly`

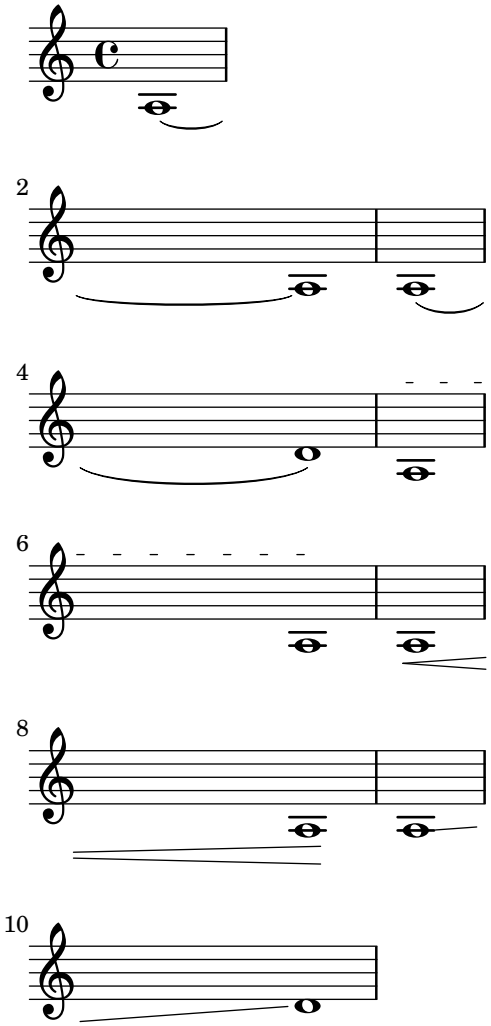


The full orchestra plays a note, where groups stop one after another. Use this to tune equalizer settings.

`midi-volume-equaliser.ly`

The property `minimum-length-after-break` can be used to stretch broken spanners starting after a line break. The following example shows usage for a variety of spanners.

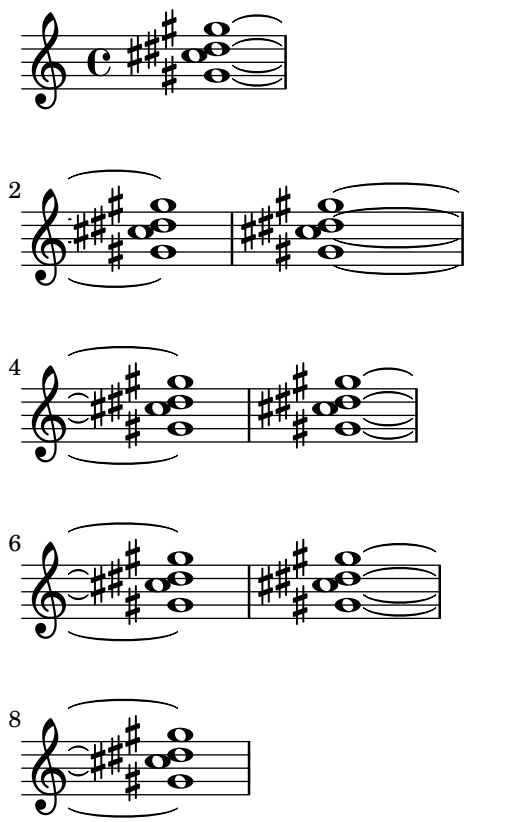
`minimum-length-after-break.ly`



The following shows the interaction between the properties `minimum-length` and `minimum-length-after-break`. When `minimum-length` is used alone, both segments of the

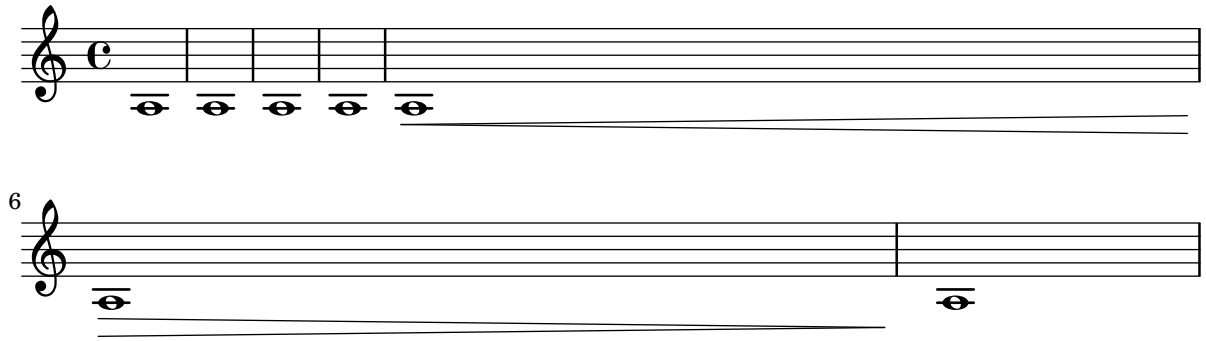
tie are affected. The properties `minimum-length-after-break` only affects the sibling starting a line. Both properties may be used together to create independent changes of both siblings. This example shows that both properties have an identical effect on the sibling after the break.

`minimum-length-broken-ties.ly`



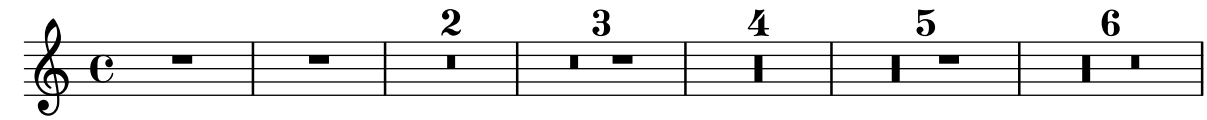
Long spanners at the end of the lines stretch measures correctly.

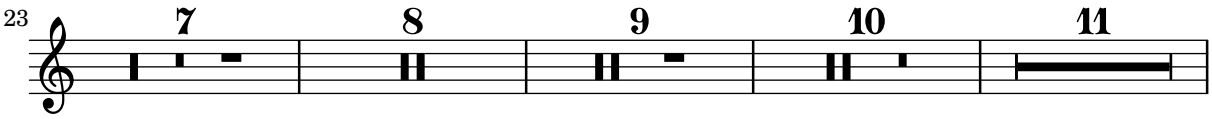
`minimum-length-end-line.ly`



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.
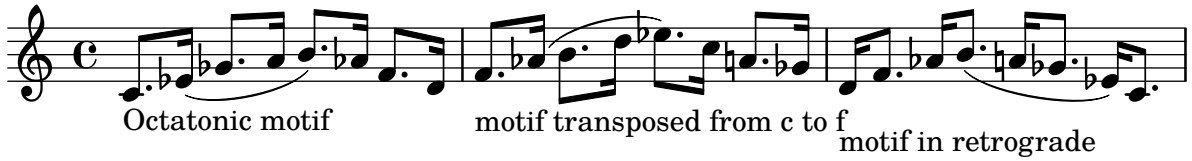
`mm-rests2.ly`
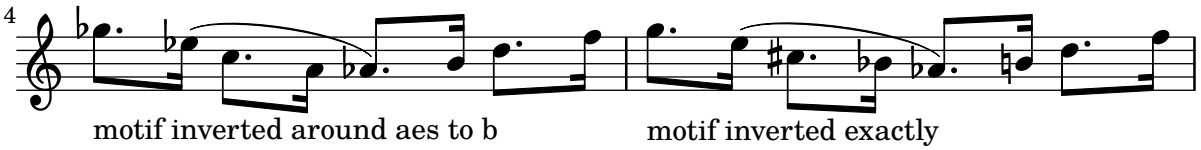




\modalTranspose, \retrograde, \inversion and \modalInversion work for an octatonic motif.

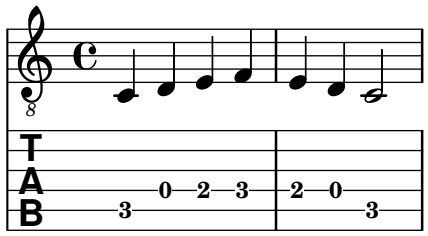
`modal-transforms.ly`

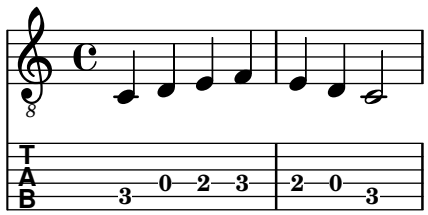




The sans serif style tab clef is automatically adjusted to different string spacings.

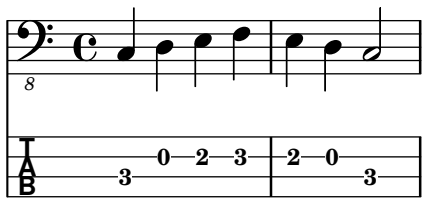
`modern-tab-clef-scaled.ly`

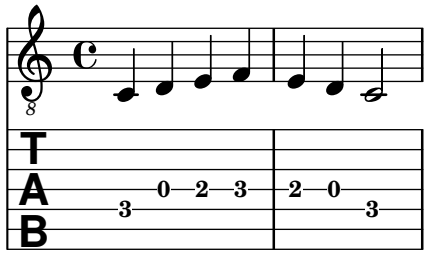




Sans serif style tab clefs are supported by `\clef moderntab`. This alternative clef supports four- to seven-stringed instruments and is scaled automatically.

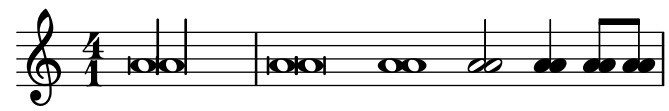
`modern-tab-clef.ly`
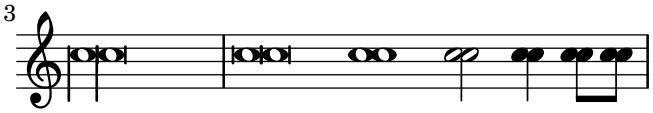




Whole notes in a monochord must be properly offset so that the heads just touch each other. On the other hand, a stem should touch both notes.

`monochords.ly`
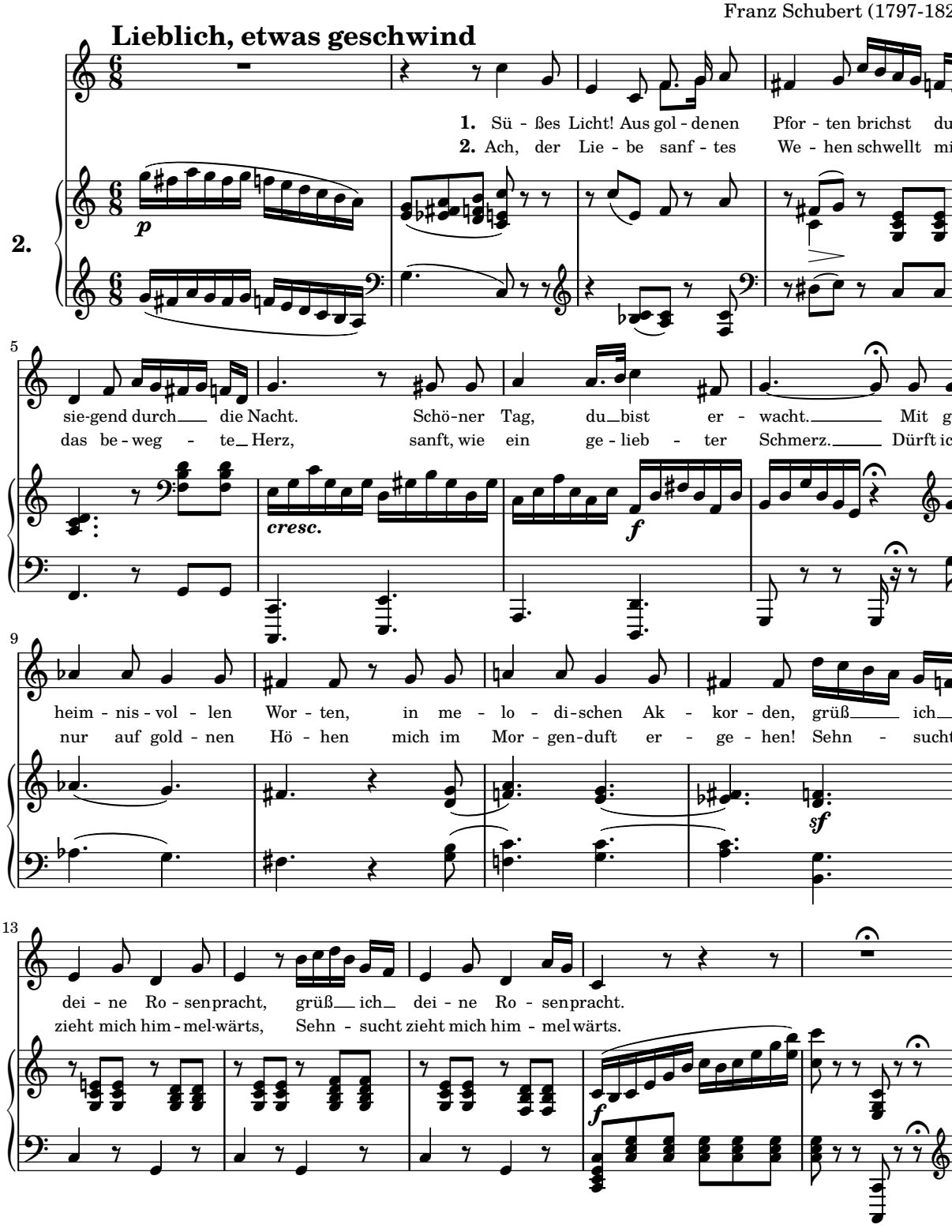




The source is a rather tightly set Peters in Edition is a heavy font. The Peters edition (4622c) was 'herausgegeben' by Paul Losse, whose name also appears on a 1956 edition of some other music. Strictly speaking, his edititorial enhancements will not be in the PD - but I am assuming there are no notable ones in this small piece.

The original compresses the entire music onto a single page, in 4 systems. Lily does so too if you tune down spacing-increment, but chooses line breaks differently.

Further manual tweaks: the slur in measure 12 has been flattened manually. The beam in measure 3, left-hand, technically is wrong, but has been added following the original. The crescendo in measure 4 has been lowered

Sängers Morgenlied

Franz Schubert (1797-182
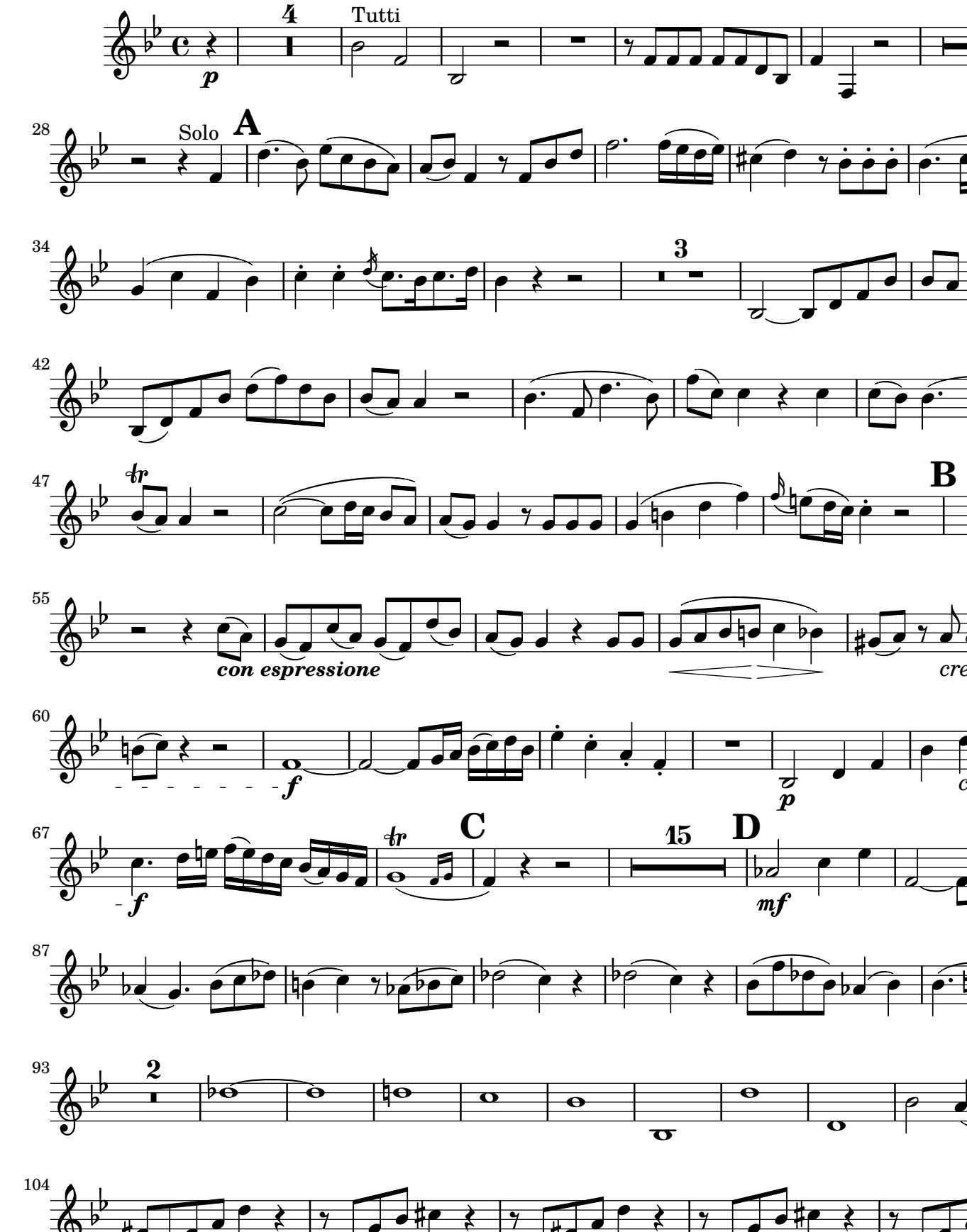


This is the Mozart 3 for horn. It's from an Edition Breitkopf EB 2563, edited by Henri Kling. Henri Kling (1842-1918) was a horn virtuoso that taught in Geneva.

mozart-hrn-3.ly

Konzert Nr. 3 Es dur

für Horn und Orchester

Horn in F Wolfgang Amadeus Mozart (1



Horn in F

122

128 **F** 3

137 3 **G**

145

152 *f* *ff* *sempre f*

157 **H**

163 *f*

171 8 tutti *f*

Cadenza ad lib.
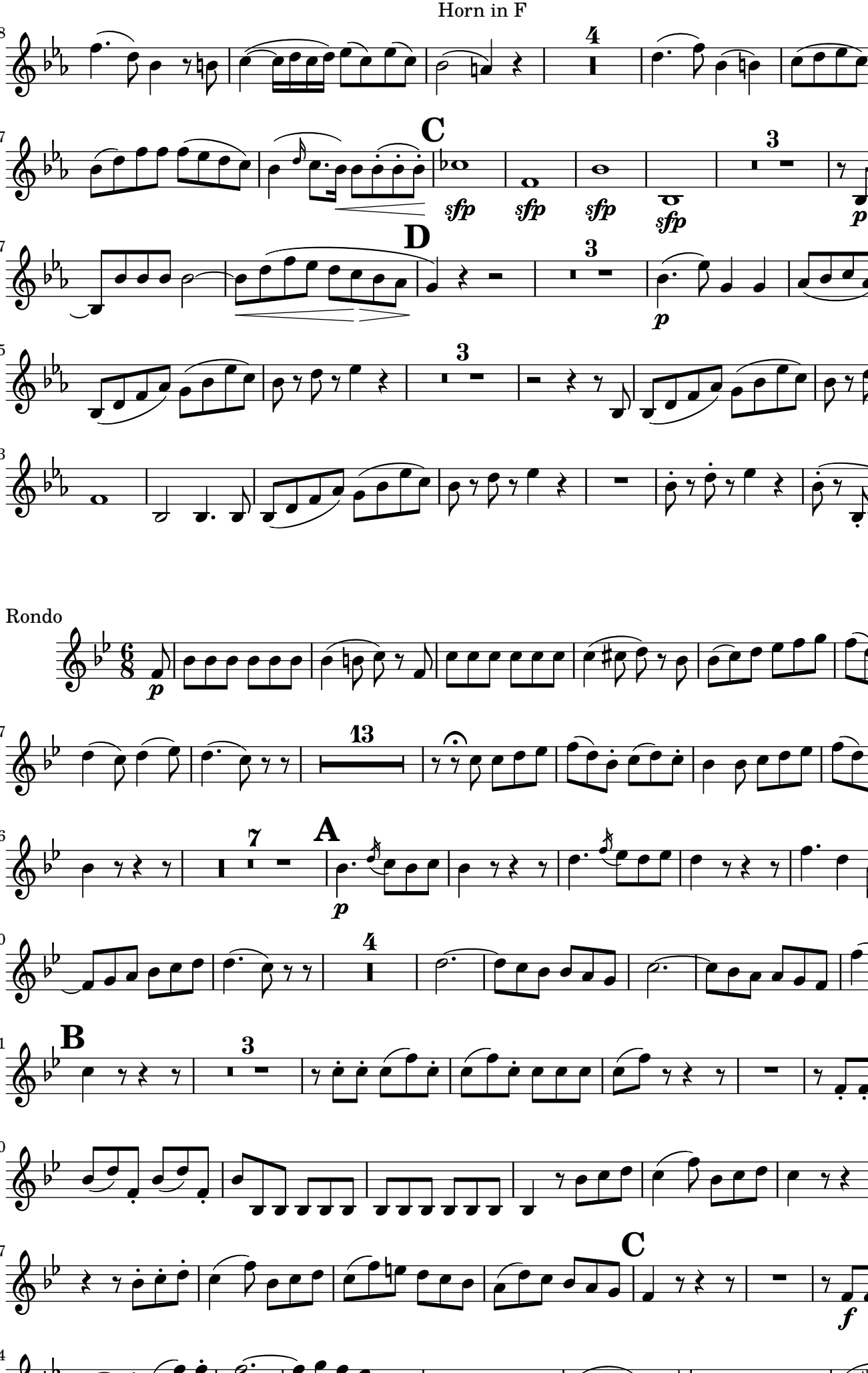
Romanze
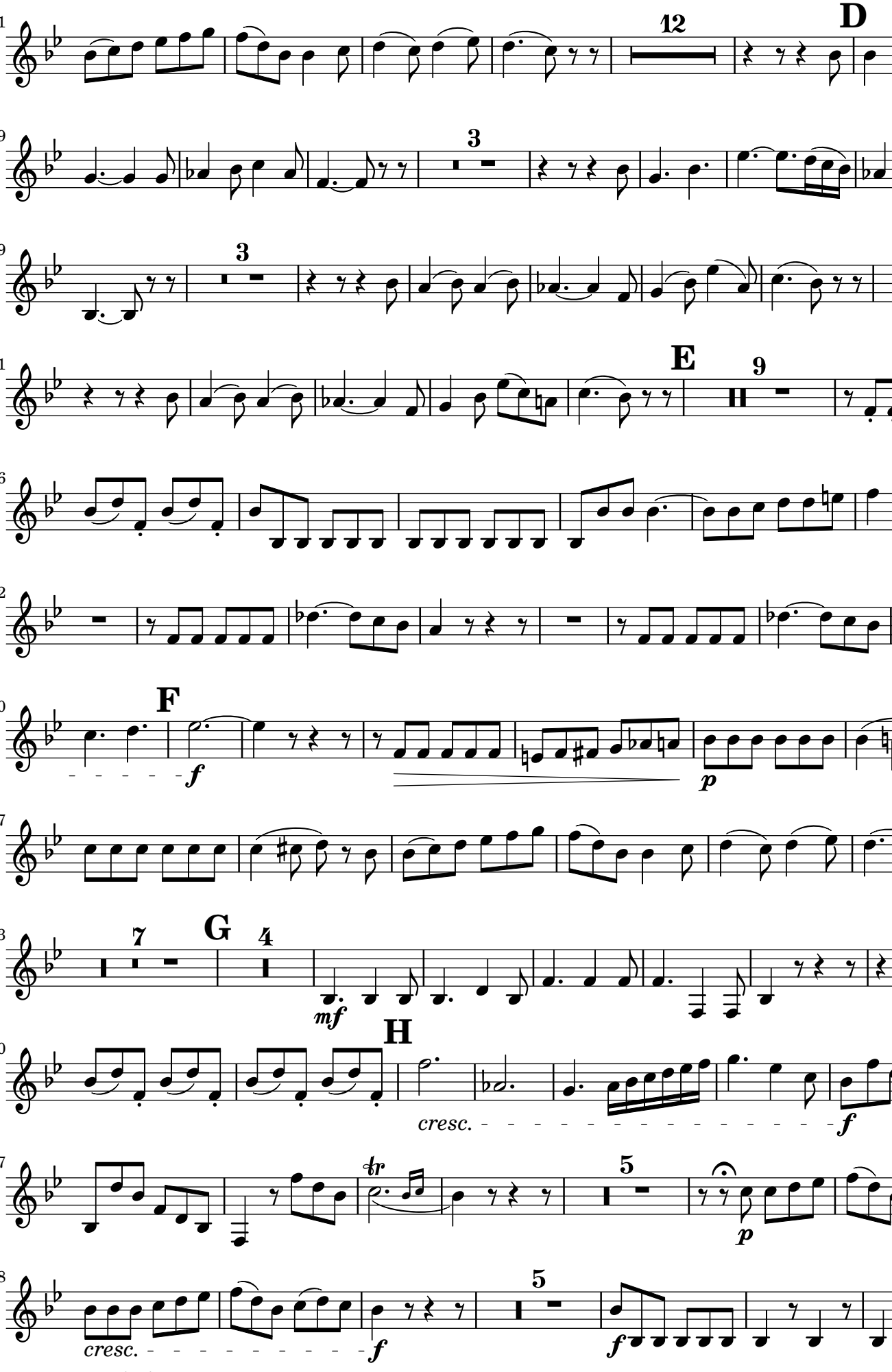
p con molto espressione

6 **A** 8 *mf*

18 2

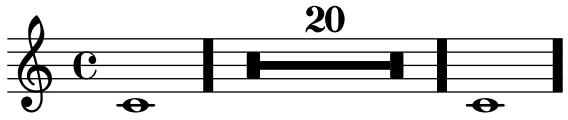
25 **B** 9


Horn in F
38
4
47
C
sfp
sfp
sfp
sfp
3
p
57
D
3
p
65
3
73
Rondo
p
7
13
26
7
A
p
40
4
51
B
3
60
67
C
f
74


81
12
D
99
3
109
3
121
E
9
136
142
150
F
f
p
157
163
7
G
4
mf
180
H
cresc.
f
187
5
p
198
cresc.
f
5
f
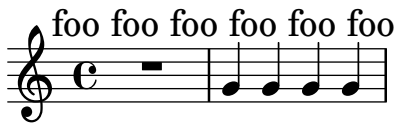
The multimeasure rest is centered exactly between bar lines.

`multi-measure-rest-center.ly`
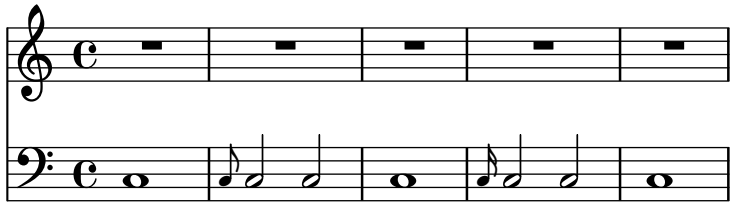


The existence of a text mark does not affect the placement of a multimeasure rest.

`multi-measure-rest-center2.ly`



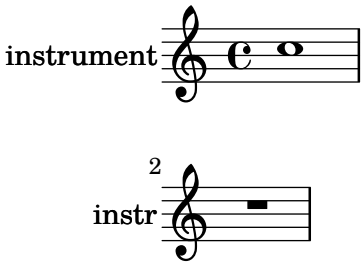
Multi-measure rests are centered also in the case of grace notes.

`multi-measure-rest-grace.ly`



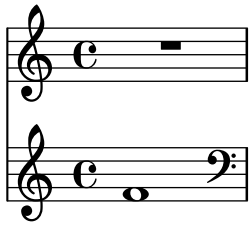
There are both long and short instrument names. Engraving instrument names should not be confused by the multimeasure rests.

`multi-measure-rest-instr-name.ly`



Though the default spacing for multi-measure rests is affected by prefatory matter in other staves, centering can be restored by overriding `spacing-pair`.

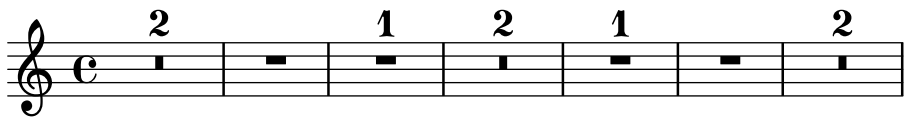
`multi-measure-rest-multi-staff-center.ly`



Setting `restNumberThreshold` affects only future multi measure rests. Unsetting it works without crashes.

The rests should be numbered 2, (none), 1, 2, 1, (none), and 2.

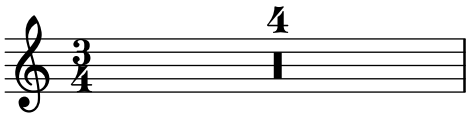
multi-measure-rest-number-threshold.ly

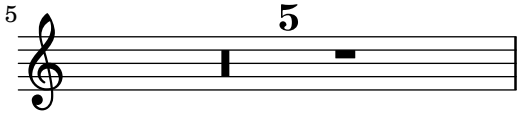


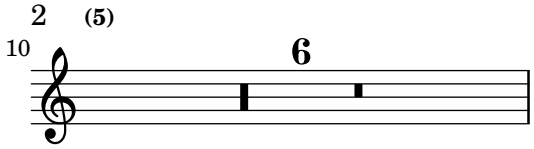
A multi measure rest reminder is a reminder printed at the top of the page, to remember how many measures you were counting.

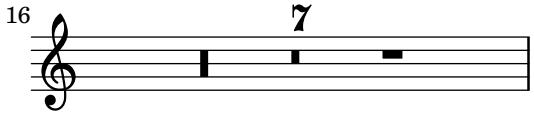
This is a demo of user-defined engravers, and defining grobs using `ly:make-grob-properties`.

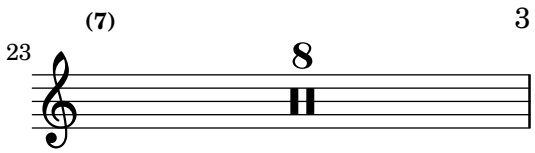
multi-measure-rest-reminder.ly







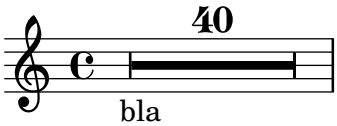




Music engraving by LilyPond 2.21.1—www.lilypond.org
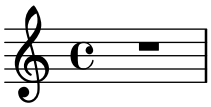
By setting texts starting with a multi-measure rest, an extra spacing column is created. This should not cause problems.

`multi-measure-rest-spacing.ly`
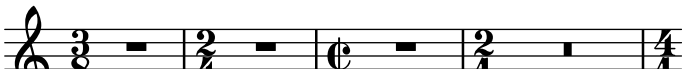


Multi measure rest staff position can be overridden to 0.

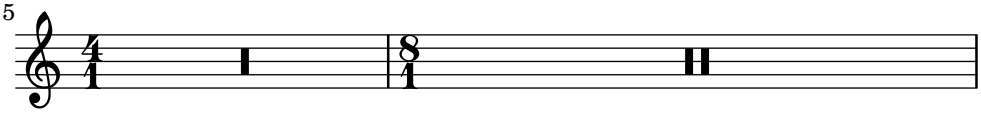
`multi-measure-rest-staff-position.ly`



Only whole, breve, longa and maxima rests are used by default for multi-measure rests.

`multi-measure-rest-standard.ly`

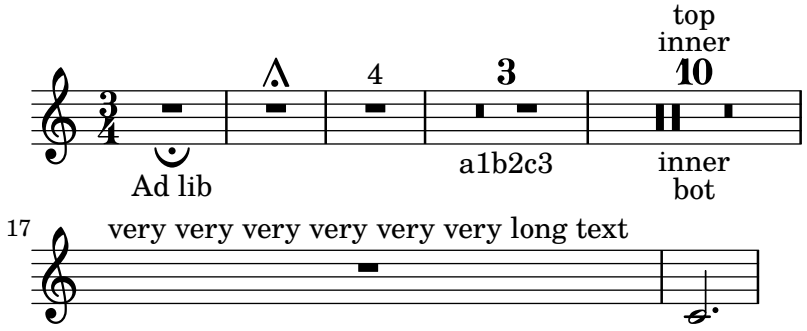




Scripts and texts may be added to the multi-measure rests.

By setting the appropriate `spacing-procedure`, we can make measures stretch to accommodate wide texts.
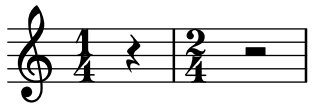
multi-measure-rest-text.ly
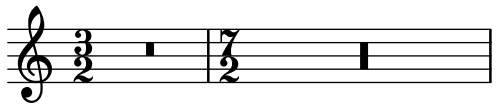


Multi-measure rests standard values can be tweaked.

`multi-measure-rest-tweaks.ly`

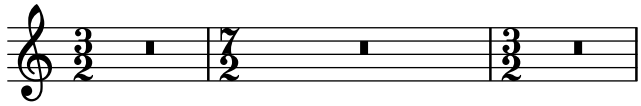
Use non-standard multi-measure rests:



Round up to the longer rest:



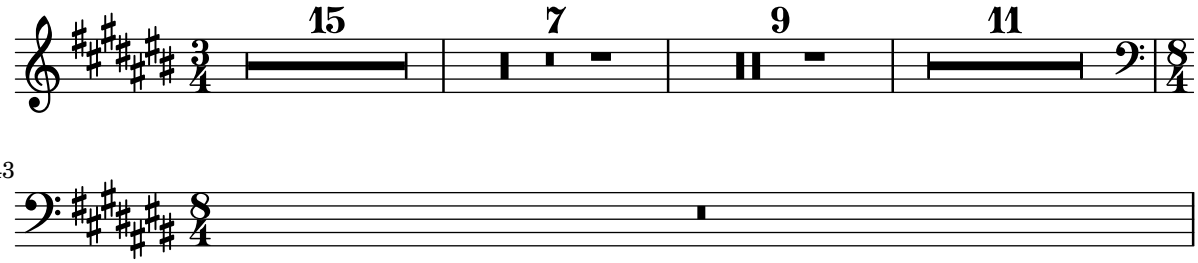
Round up to the longer rest only in specified time signatures:



Multi-measure rests do not collide with bar lines and clefs. They are not expanded when you set `Score.skipBars`. Although the multi-measure-rest is a Spanner, minimum distances are set to stop it colliding with bar lines.
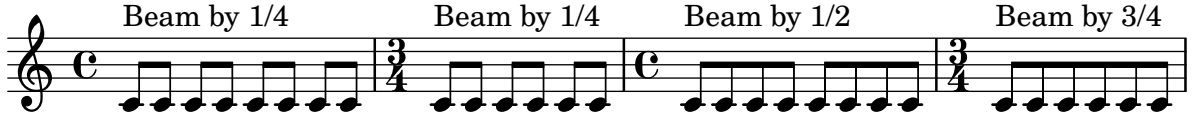
Rests over measures lasting longer than 2 wholes use breve rests. When more than 10 measures (tunable through `expand-limit`) are used then a different symbol is used.

`multi-measure-rest.ly`
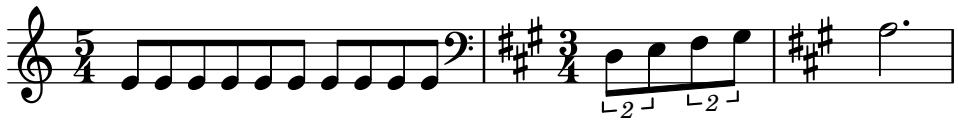


Multiple overrides to the default time signature settings can be added. In this example, notes should be beamed as indicated by the markups.

`multiple-time-sig-settings.ly`



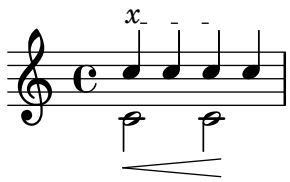
Music functions can be called directly from Scheme.

`music-function-direct-call.ly`
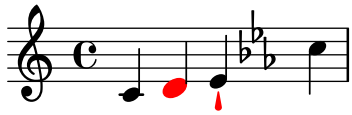


the `endSpanners` music function inserts end span events at the end of a note.

`music-function-end-spanners.ly`



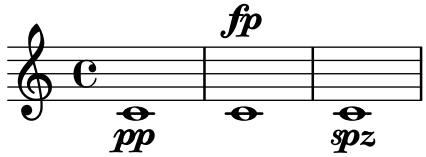
For defining a music function, one can supply one or several music function calls chained together, cutting the last call short using `\etc`. The remaining arguments are supplied when calling the music function defined in this manner.

`music-function-incomplete.ly`
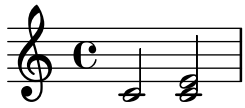


Music functions may be attached to notes; in this case they must be introduced by a direction indicator. If a non-neutral direction is given (i.e. anything else than a dash), then the 'direction property of the resulting object is set accordingly.

`music-function-post-event.ly`



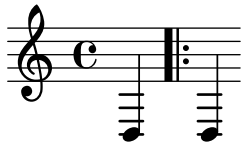
Music functions accept strings as markup arguments when using the type predicate `markup?`

`music-function-string-markup.ly`



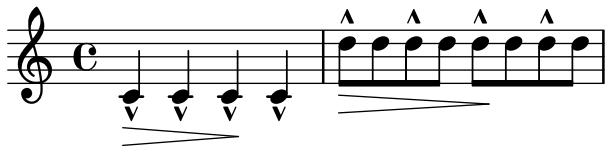
Music functions are generic music transformation functions, which can be used to extend music syntax seamlessly. Here we demonstrate a `\myBar` function, which works similar to `\bar`, but is implemented completely in Scheme.

`music-function.ly`



With `music-map`, you can apply functions operating on a single piece of music to an entire music expression. In this example, 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.
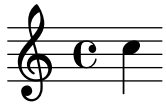
`music-map.ly`



Nested fill-lines should work properly. In this example, both occurrences of FOO should be centered.

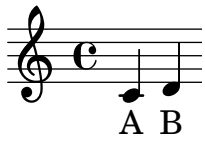
`nested-fill-lines.ly`

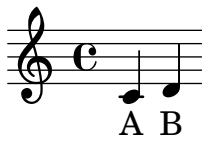
| FOO |
| FOO |



addlyrics do not need braces around their arguments, in particular if the arguments are variables.

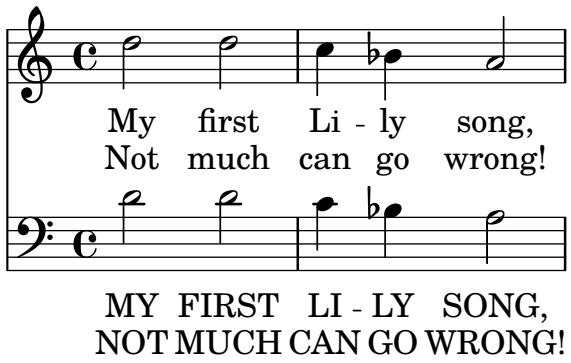
`newaddlyrics-music-identifiers.ly`





newlyrics, multiple stanzas, multiple lyric voices.

`newaddlyrics.ly`



`no-header.ly`

This regtest does not contain any header and paper blocks. Its purpose is to test

whether anything breaks if these blocks are absent.

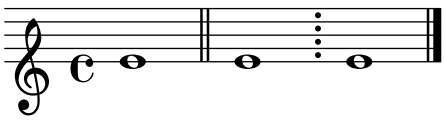
The printing of the staff lines may be suppressed by removing the corresponding engraver.

`no-staff.ly`
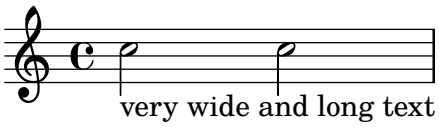


Bar lines are positioned correctly when using custom staves which are not centered around position 0.

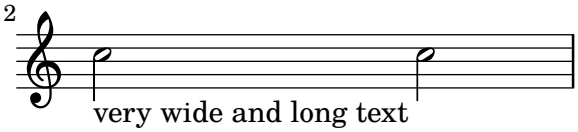
`non-centered-bar-lines.ly`



By default, text is set with empty horizontal dimensions. The property `extra-spacing-width` in `TextScript` is used to control the horizontal size of text.

`non-empty-text.ly`
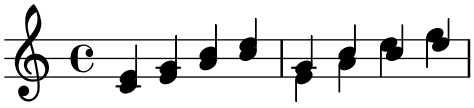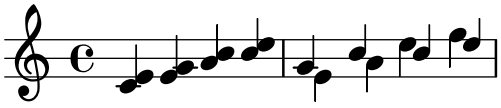




Whether simultaneous notes are identified as vertically colliding or not depends on the value of the `note-collision-threshold` property of the `Stem` grob (for notes in the same voice) and the `NoteCollision` grob (for notes in different voices).

`note-collision-threshold.ly`

collisions
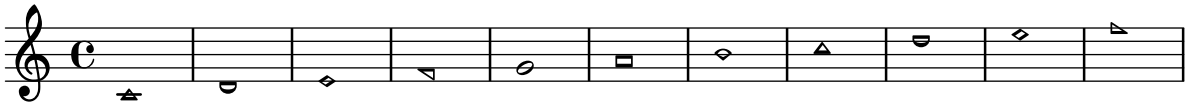


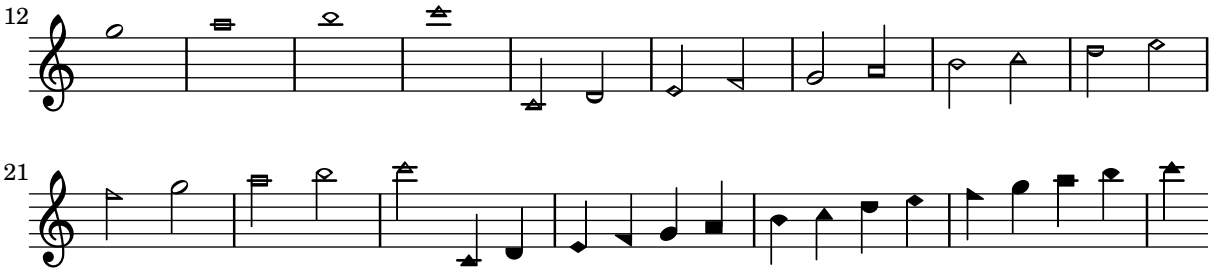
collisions prevented



Notes can be set in the Aiken (Christian Harmony) style.
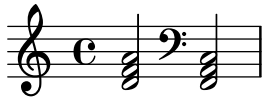
`note-head-aiken.ly`
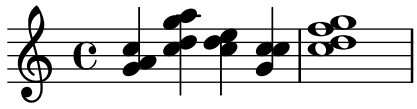




Note heads are placed on the correct side of the stem; this placement changed is not changed by magic values of layout-set-staff-size. (Fix of issue 5303.)

`note-head-chord-layout-set-staff-size.ly`



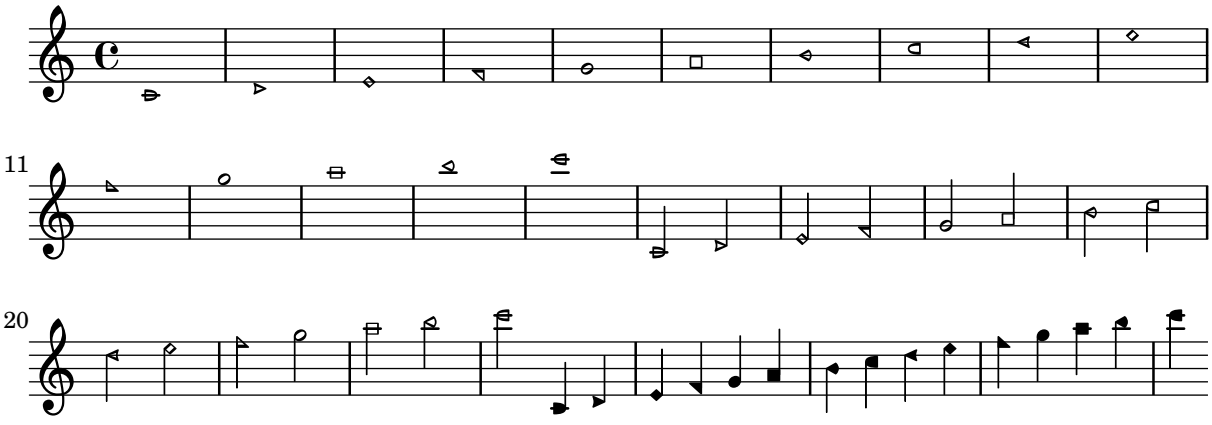
Note heads are flipped on the stem to prevent collisions. It also works for whole heads that have invisible stems.

`note-head-chord.ly`
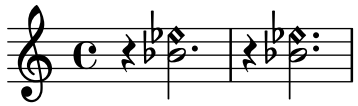


Notes can be set in the Funk (Harmonia Sacra) style.

`note-head-funk.ly`



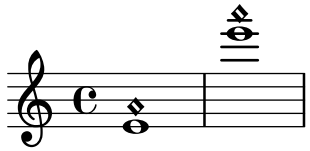
Dots on harmonic note heads can be shown by setting the property `harmonicDots`.

`note-head-harmonic-dotted.ly`



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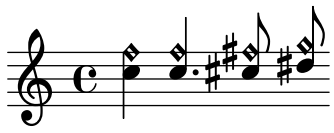
`note-head-harmonic-whole.ly`



The handling of stems for harmonic notes must be completely identical to normal note heads.
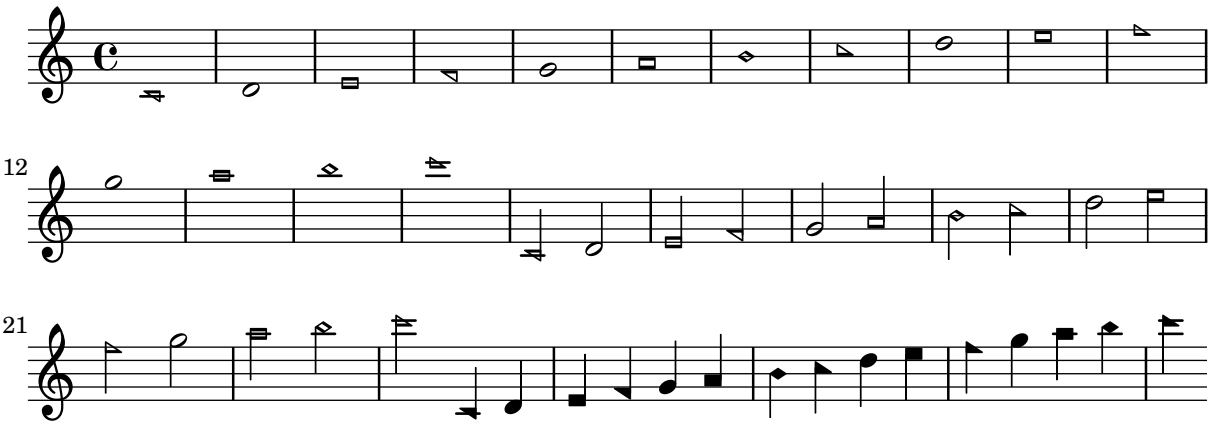
Harmonic heads do not get dots. If `harmonicAccidentals` is unset, they also don't get accidentals.

`note-head-harmonic.ly`
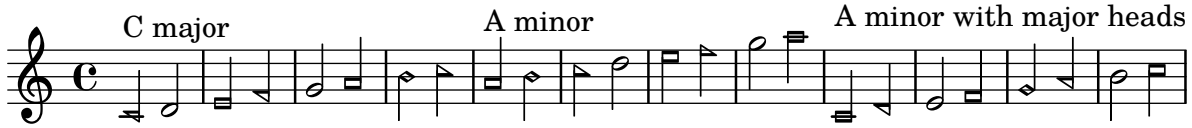


Notes can be set in the Sacred Harp style.

`note-head-sacred-harp.ly`



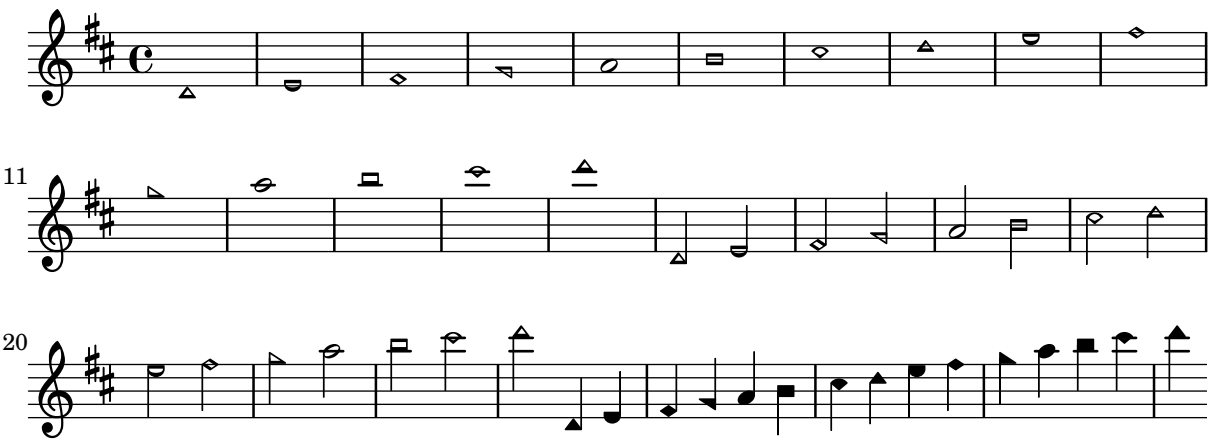
Shape notes can be set to work properly in minor keys.

`note-head-shape-minor.ly`
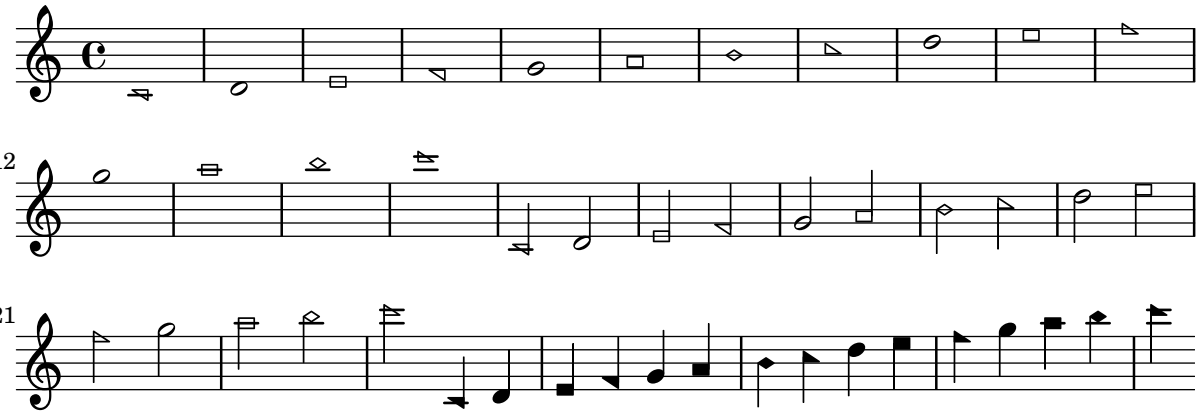


With `shapeNoteStyles`, the style of the note head is adjusted according to the step of the scale, as measured relative to the `tonic` property.

`note-head-solfa.ly`



Notes can be set in the Southern Harmony style.

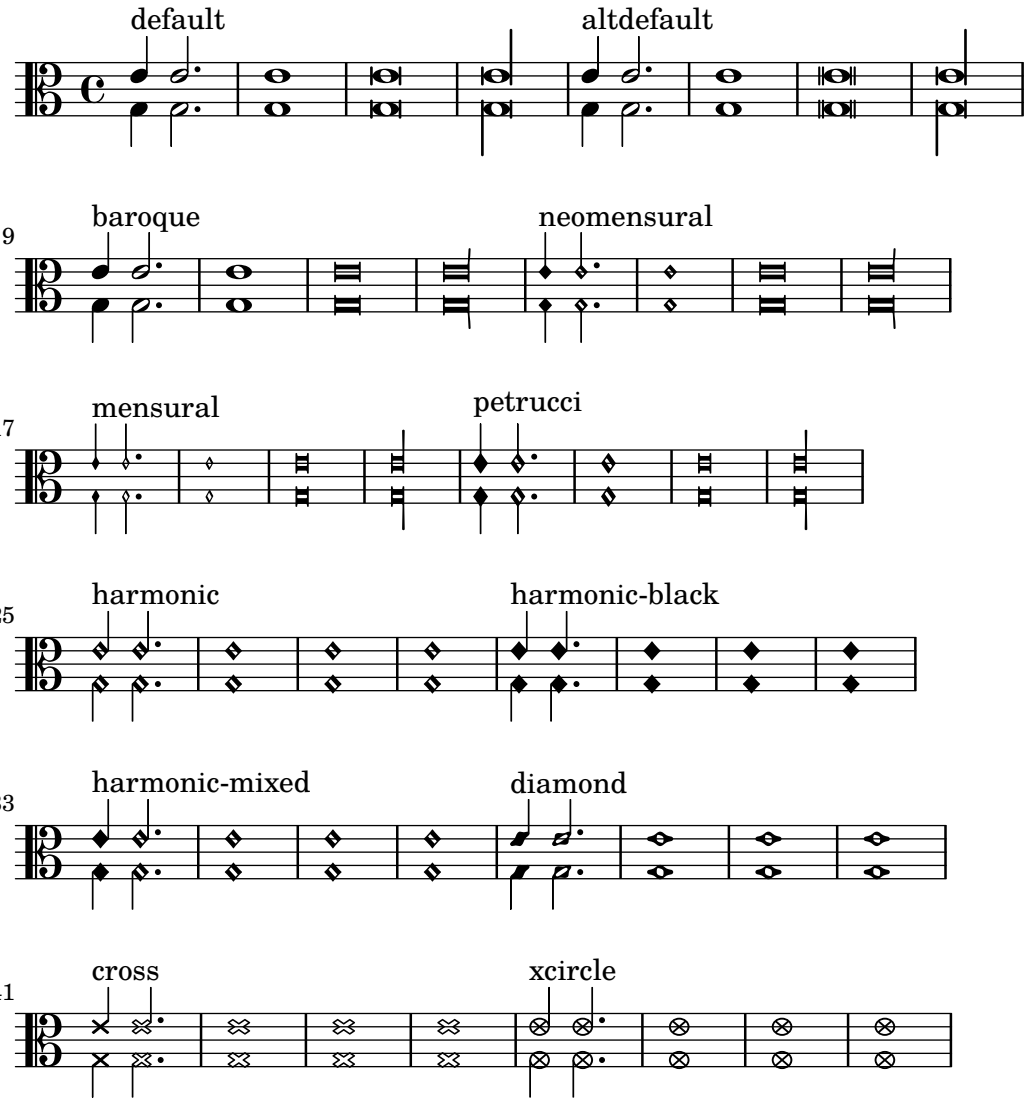
note-head-southern-harmony.ly

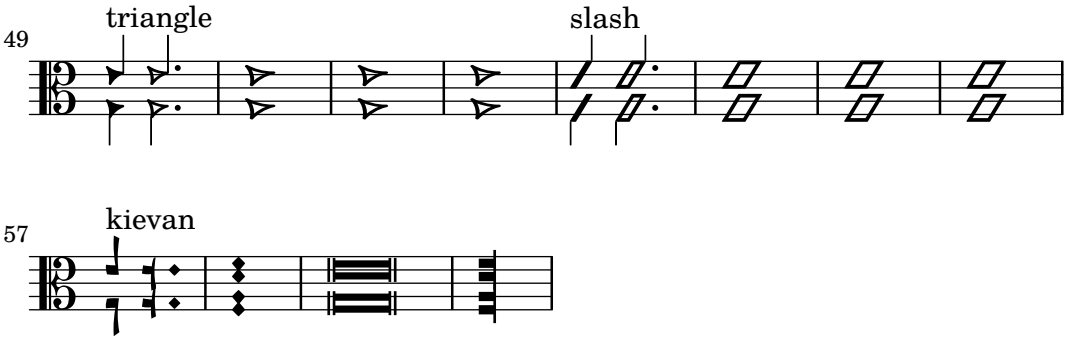


Note head shapes may be set from several choices. The stem endings should be adjusted according to the note head. If you want different note head styles on one stem, you must create a special context.

Harmonic notes have a different shape and different dimensions.

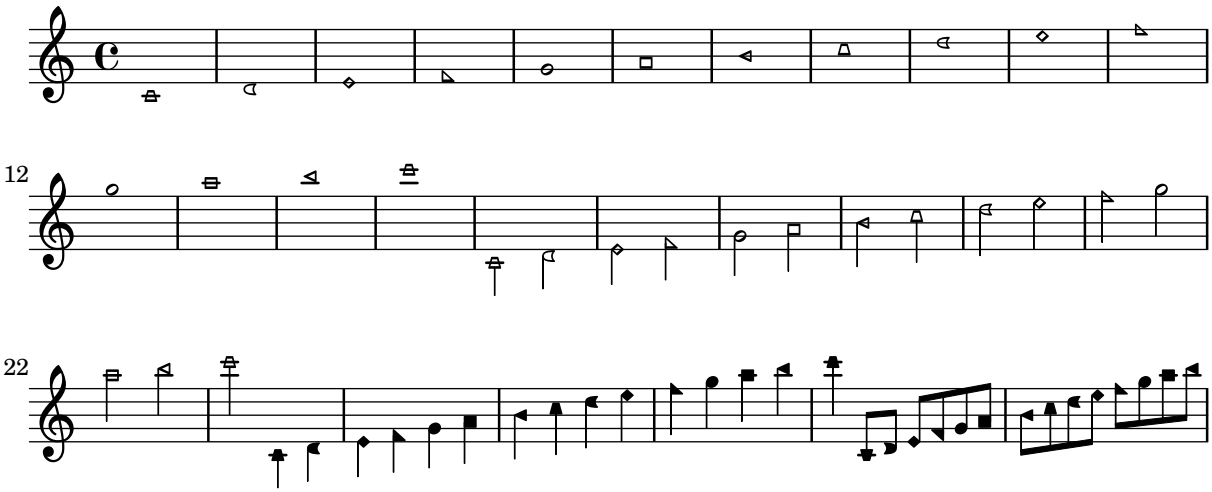
note-head-style.ly





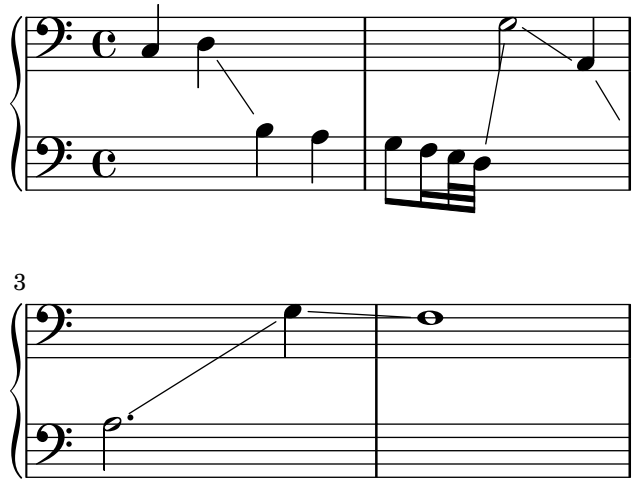
Notes can be set in the Walker (Christian Harmony) style.

`note-head-walker.ly`



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

`note-line.ly`



Note names may be printed in various languages, with or without accidentals and octave marks.

`note-name-context-custom.ly`

la sibb do# reb+fa+lab mid

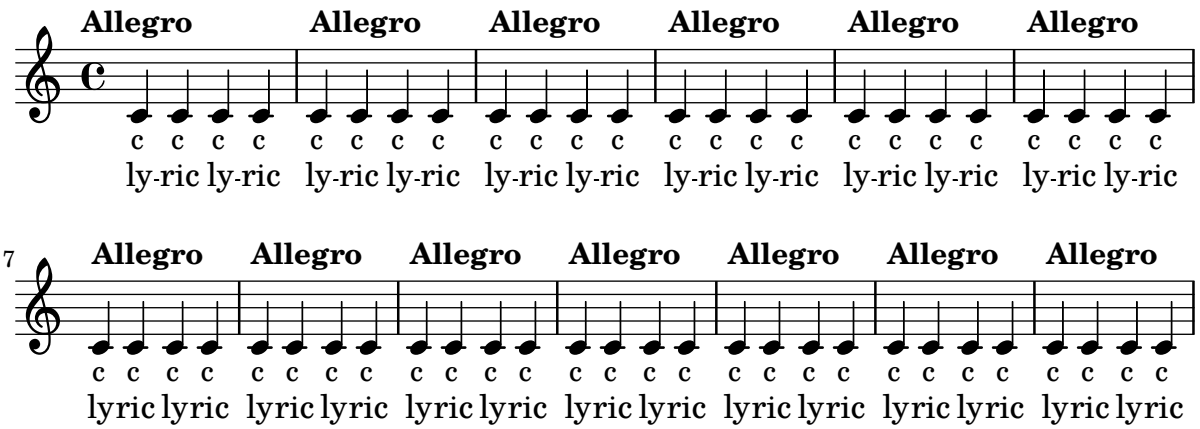
(ref.) la sibb do# reb+fa+lab mid

do' ré♯' mi' la$_5$

(ref.) do' ré♯' mi' la5
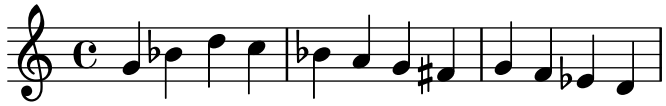
NoteNames context should be close to the related notes, and should not collide with the tempo markings.

`note-names-context.ly`



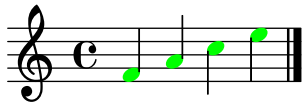
Various languages are supported for note names input. Selecting another language within a music expression is possible, and doesn't break point-and-click abilities.

`note-names.ly`



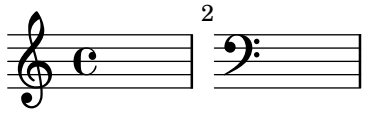
Noteheads do not extend above the upper staff line.

`notehead-height.ly`
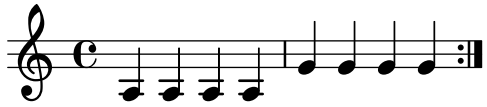


'`NullVoice`' responds to `\change Staff` as a '`Voice`' would. In consequence, in the first shown system it keeps a single treble-clef staff alive. In the second system, it is in a single bass-clef staff.

`nullvoice-change.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.

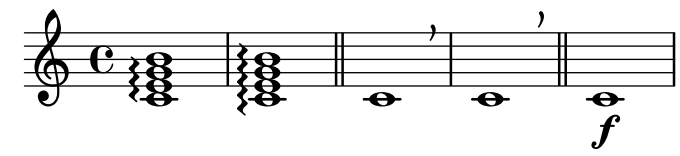
`number-staff-lines.ly`
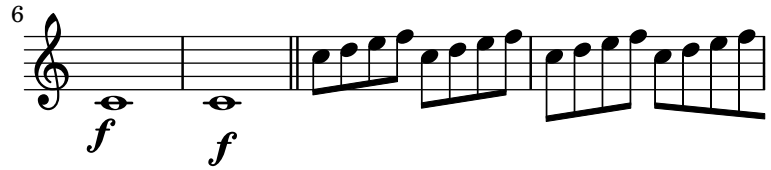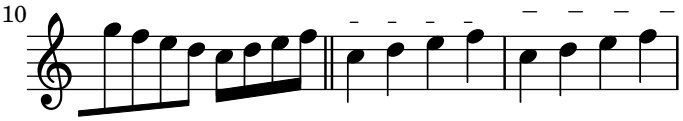


The `\offset` command may be used to displace various properties from the default settings contained in grob descriptions. Settings which may be offset are limited to those of type `number`,
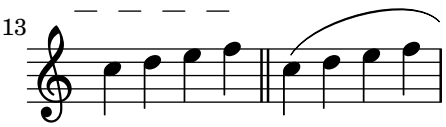
number-pair, or number-pair-list. Most of the following examples begin with the grob in its default appearance. The command is demonstrated as a tweak and as an override.

offsets.ly





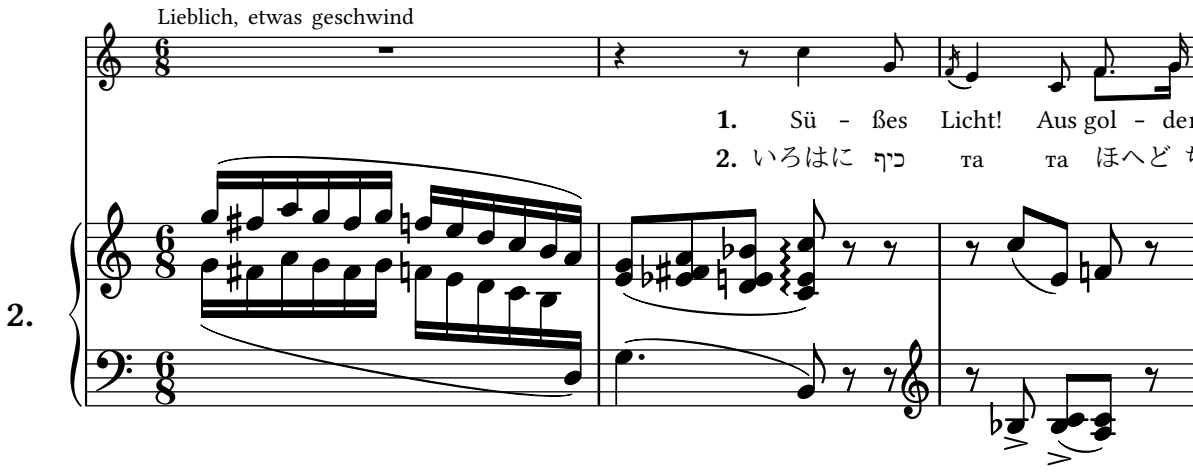






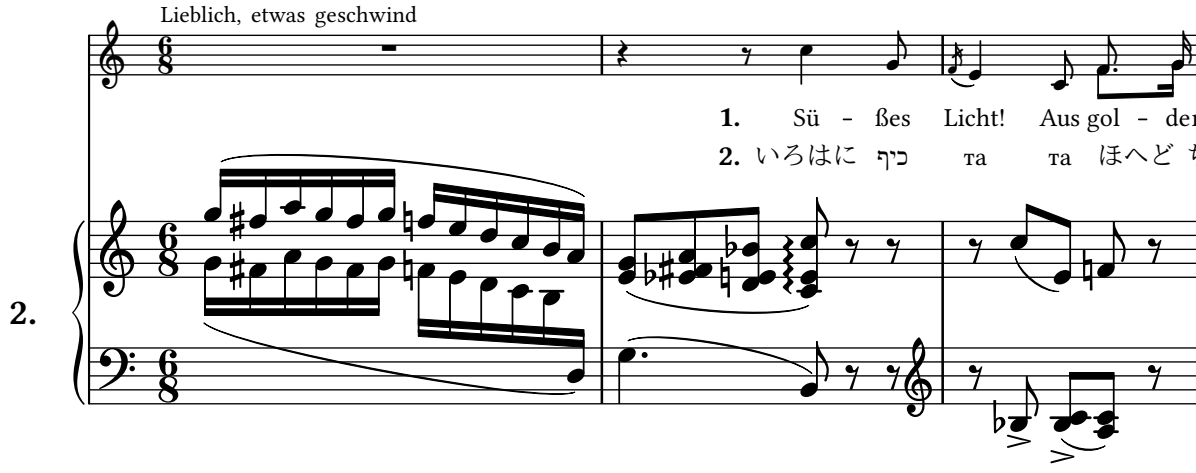
heavily mutilated Edition Peters Morgenlied by Schubert

one-line-auto-height-breaking.ly



heavily mutilated Edition Peters Morgenlied by Schubert
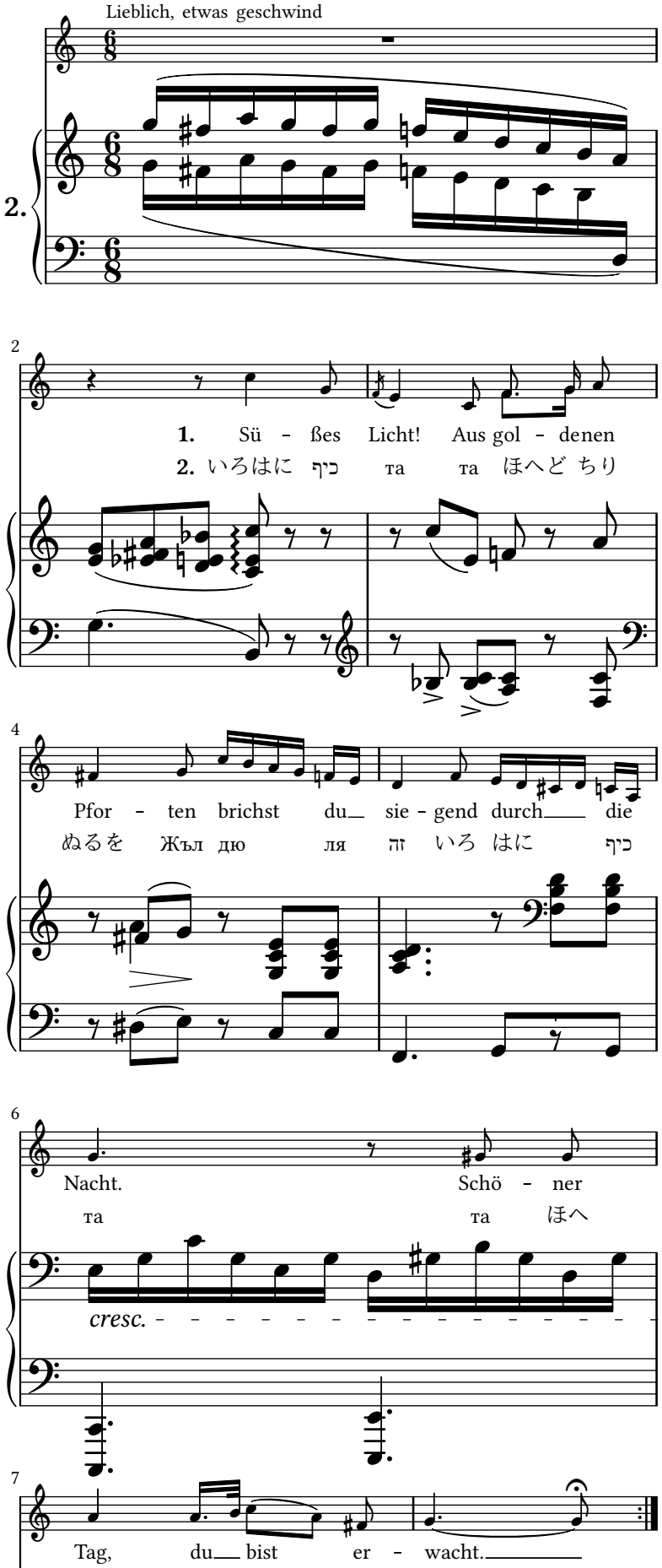
one-line-breaking.ly



heavily mutilated Edition Peters Morgenlied by Schubert
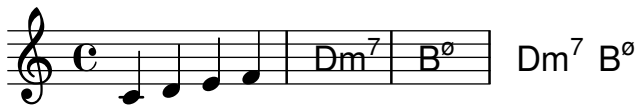
one-page-breaking.ly

LilyPond demo
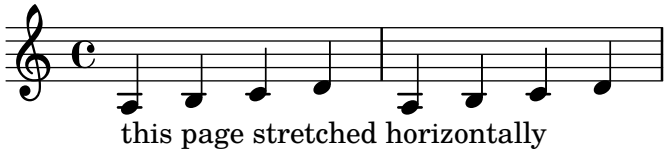


`OneStaff` contexts can be used for letting several contexts use the same vertical position. This example shows chords being placed in a staff and immediately following it.
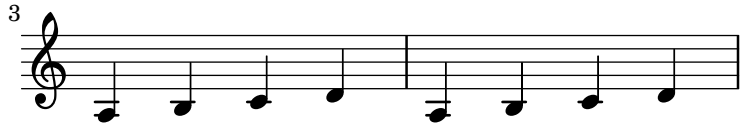
`one-staff.ly`
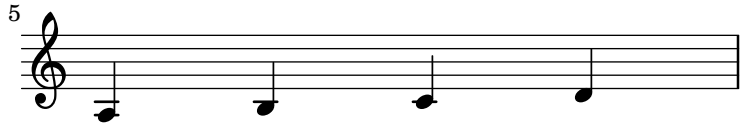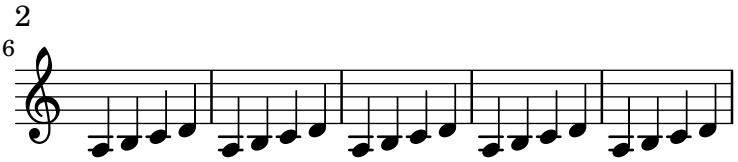


The optimal page breaker will make trade-offs between horizontal and vertical stretching so that the overall spacing will be more acceptable. The `page-spacing-weight` parameter controls the relative importance of vertical/horizontal spacing. Because `ragged-last-bottom` is on, there is no penalty for odd vertical spacing on the final page. As a result, only the first page should be horizontally stretched.

`optimal-page-breaking-hstretch.ly`









this page with natural spacing

Music engraving by LilyPond 2.21.1—www.lilypond.org

Print the option help text, for comparison against previous releases.

`option-help.ly`

Test backup of predicate-based optional music function arguments.

Unit expressions like `3\cm` can't be parsed as optional arguments in one go since they would require lookahead after `3`. The predicate is checked after `3`, and if it is suitable, Lilypond commits to parsing as a unit number, and checks the result again. For the predicate `integer?` and `3\cm`, you would actually get a syntax error (since the combination is no longer an integer) rather than Lilypond trying to see `3\cm` as two separate arguments.
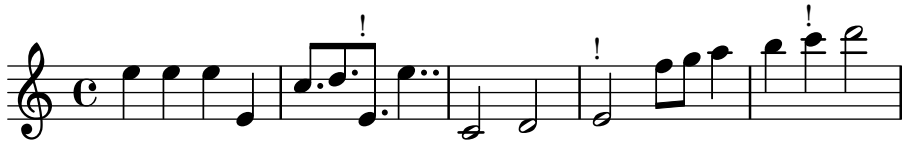
`optional-args-backup.ly`

Test predicate-based optional music function argument skipping.

`optional-args-predicate.ly`
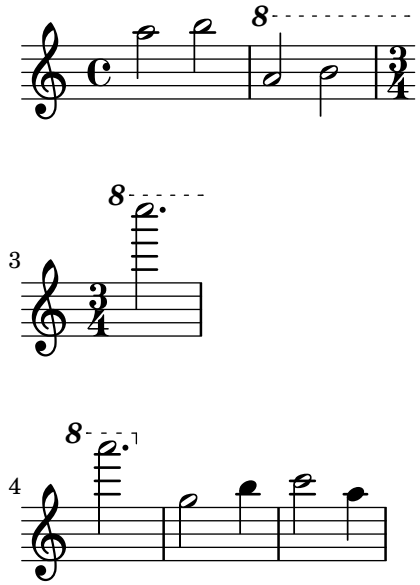
Test optional music function arguments. The output is nonsensical, but if you wrack your brain, you'll figure it out. Remember that optional arguments are matched left to right, and after the first non-match, the rest is skipped.

`optional-args.ly`
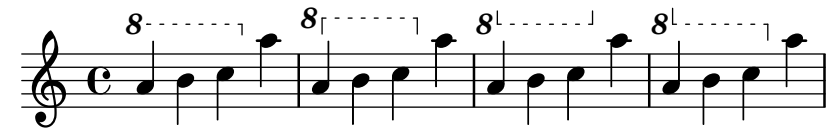


At line breaks, ottava brackets have no vertical line and their horizontal line does not stick out. The dashed line runs until the end of the line (regardless of prefatory matter).

`ottava-broken.ly`



Both edge heights of an ottava bracket can be specified.

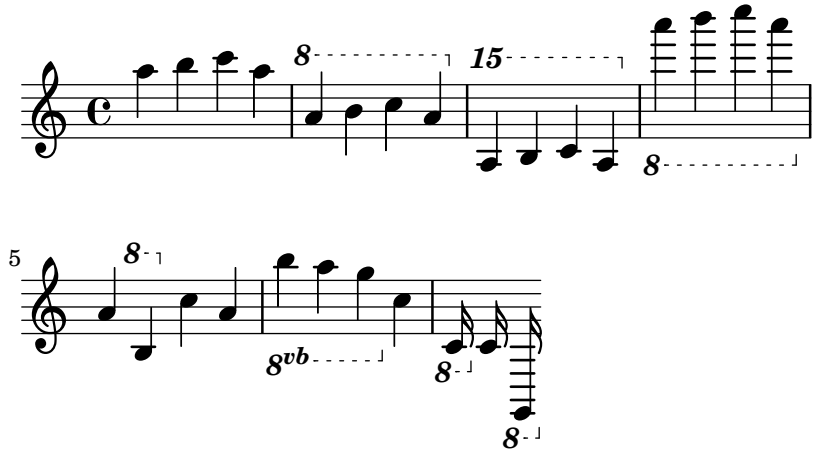
`ottava-edge.ly`



Ottava brackets are supported, through the use of the music function `\ottava`.

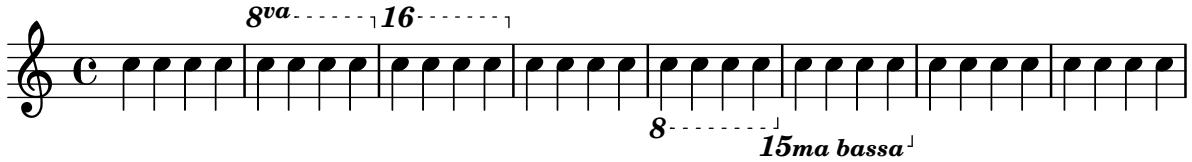
The spanner should go below a staff for 8va bassa, and the ottavation markup can be tuned with `Staff.ottavation`.

`ottava.ly`



Ottavation markups can be changed by the user. LilyPond warns about missing markups (in this example for +3 and -3 octaves).

ottavation-markups.ly



Shows the output-attributes property of a grob being set. This should have no effect in the Postscript backend. In the SVG backend these settings should produce this group tag: `<g id="123" class="foo" data-whatever="bar"> ... </g>`

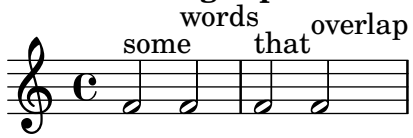
output-attributes.ly



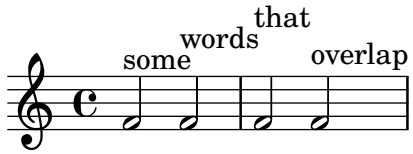
The `outside-staff-placement-directive` adjusts the order in which objects are placed outside the staff.

outside-staff-placement-directive.ly

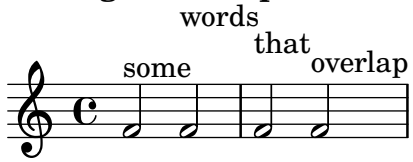
left-to-right-polite
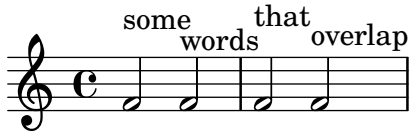


left-to-right-greedy



right-to-left-polite

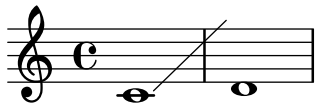


right-to-left-greedy



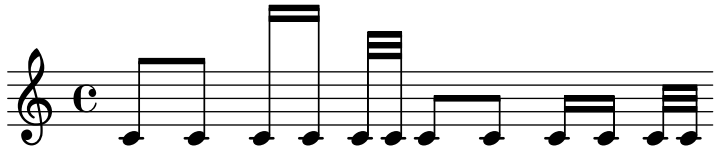
A sublist of grob property lists may be overridden within a callback. This test uses a custom stencil callback which changes the Y coordinate of the right bound of the glissando spanner.

override-nest-scheme.ly
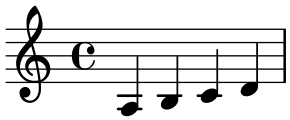


Sublist of grob property lists may be also tuned. In the next example, the `beamed-lengths` property of the `Stem` grob is tweaked.

`override-nest.ly`

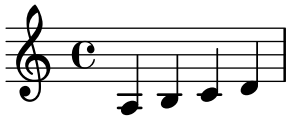


Page breaks work when they are placed at the end of a score, or between scores.

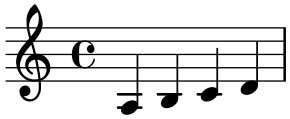
`page-break-between-scores.ly`



2
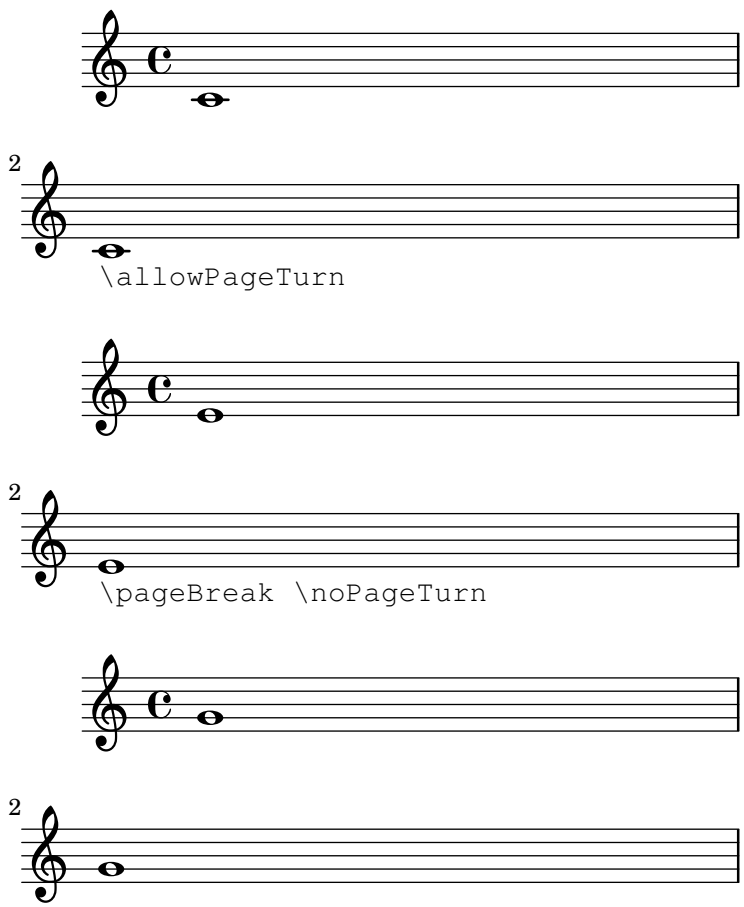


3



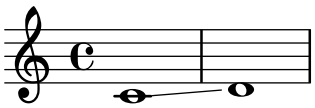
Page breaking and page turning commands (`\pageBreak`, `\noPageBreak`, etc), can be used at top level.

`page-break-turn-toplevel.ly`



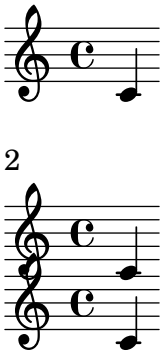
If a page break is forced where it is forbidden, a warning is printed.

`page-break-warn-forbidden.ly`
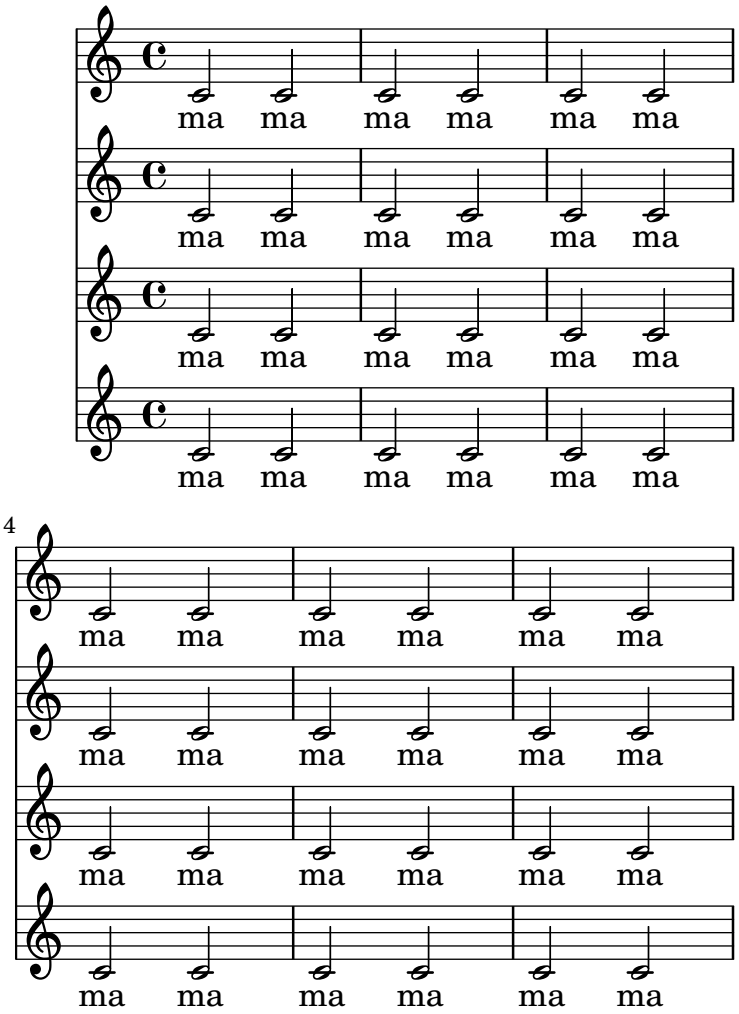


Page breaks are allowed by default at the end of the score, but the user can override them. There should be one line on the first page and two (colliding) lines on the second page.

`page-breaking-end-of-score.ly`



The page breaking algorithm can handle clefs combined with lyrics. That is, the Y-extent approximations are a little more accurate than just using bounding boxes. In particular, everything should fit on one page here.

`page-breaking-good-estimation.ly`
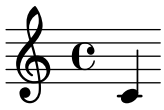


Music engraving by LilyPond 2.21.1—www.lilypond.org

Padding between markups is honored by the page breaker. This should take up two pages.

`page-breaking-markup-padding.ly`

2
01
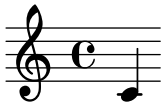


Music engraving by LilyPond 2.21.1—www.lilypond.org

Padding between a markup and a system is honored by the page breaker. This should take up two pages.

`page-breaking-markup-padding2.ly`

00
01

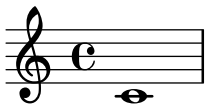


Music engraving by LilyPond 2.21.1—www.lilypond.org

Padding between a score and a markup is honored by the page breaker. This should take up two pages.

`page-breaking-markup-padding3.ly`

00
01

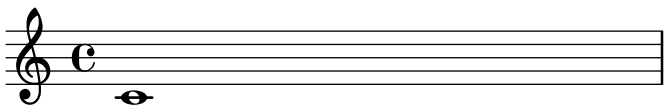


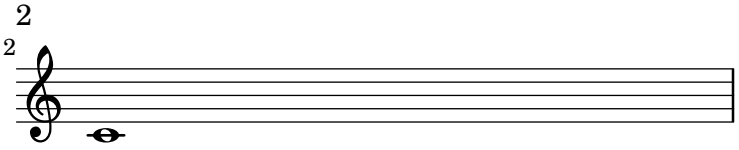
Music engraving by LilyPond 2.21.1—www.lilypond.org

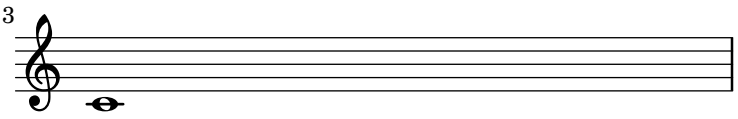
The max-systems-per-page variable prevents more than a given number of systems from being on a page. Titles are not counted as systems. \noPageBreak can override max-systems-per-page in unusual situations.

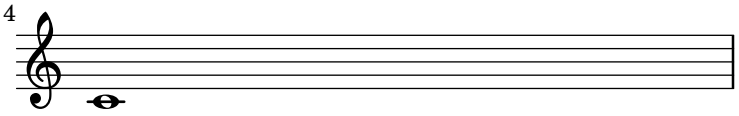
`page-breaking-max-systems-per-page.ly`

Title




2
2

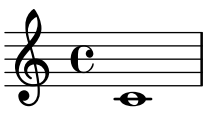

3

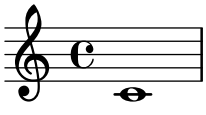


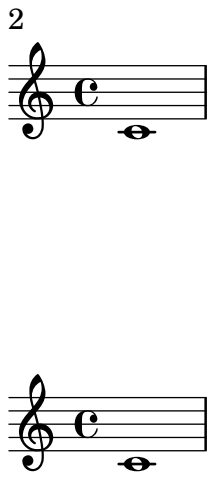
Music engraving by LilyPond 2.21.1—www.lilypond.org

minimum-distance is correctly accounted for in page breaking.

`page-breaking-min-distance.ly`







Music engraving by LilyPond 2.21.1—www.lilypond.org

minimum-distance within a system is correctly accounted for in page breaking.

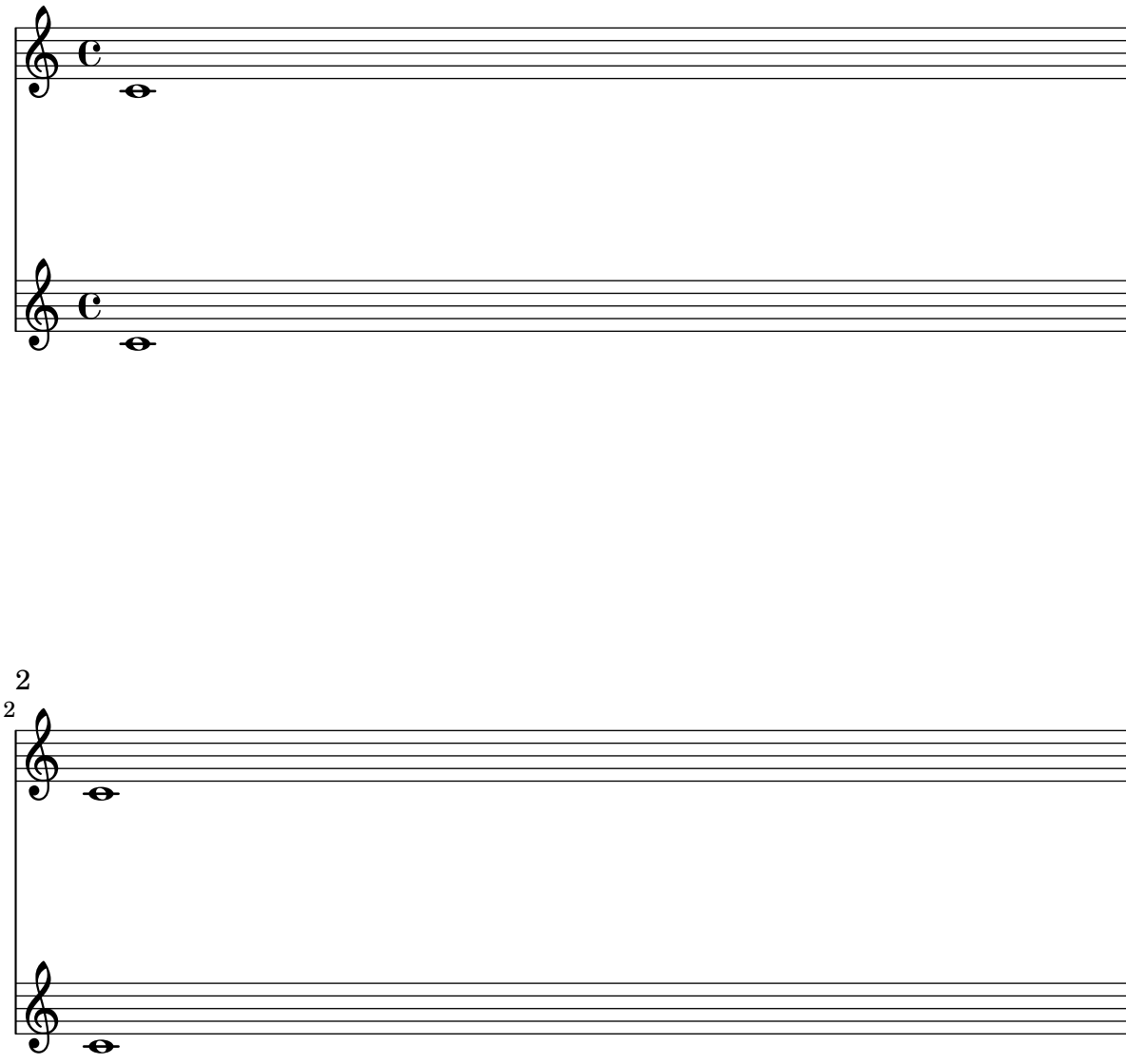
`page-breaking-min-distance2.ly`



Music engraving by LilyPond 2.21.1—www.lilypond.org

minimum-distance within a system is correctly accounted for in page breaking.

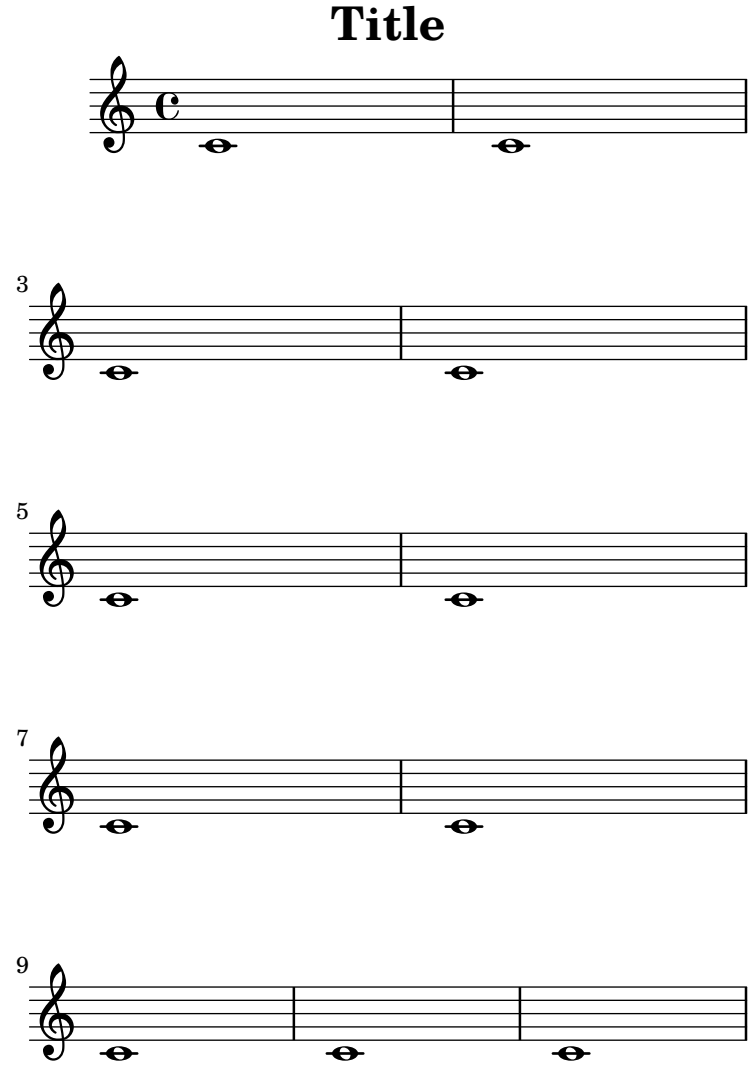
page-breaking-min-distance3.ly
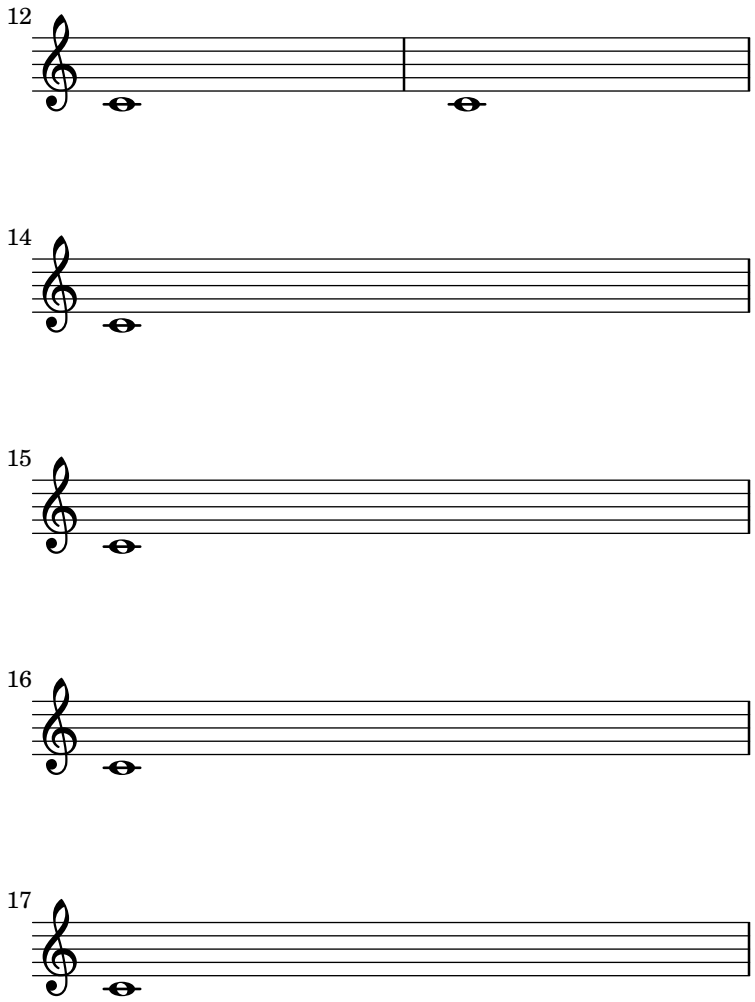


Music engraving by LilyPond 2.21.1—www.lilypond.org

The min-systems-per-page variable forces each page to have a minimum number of systems. Titles do not count as systems here.

page-breaking-min-systems-per-page1.ly

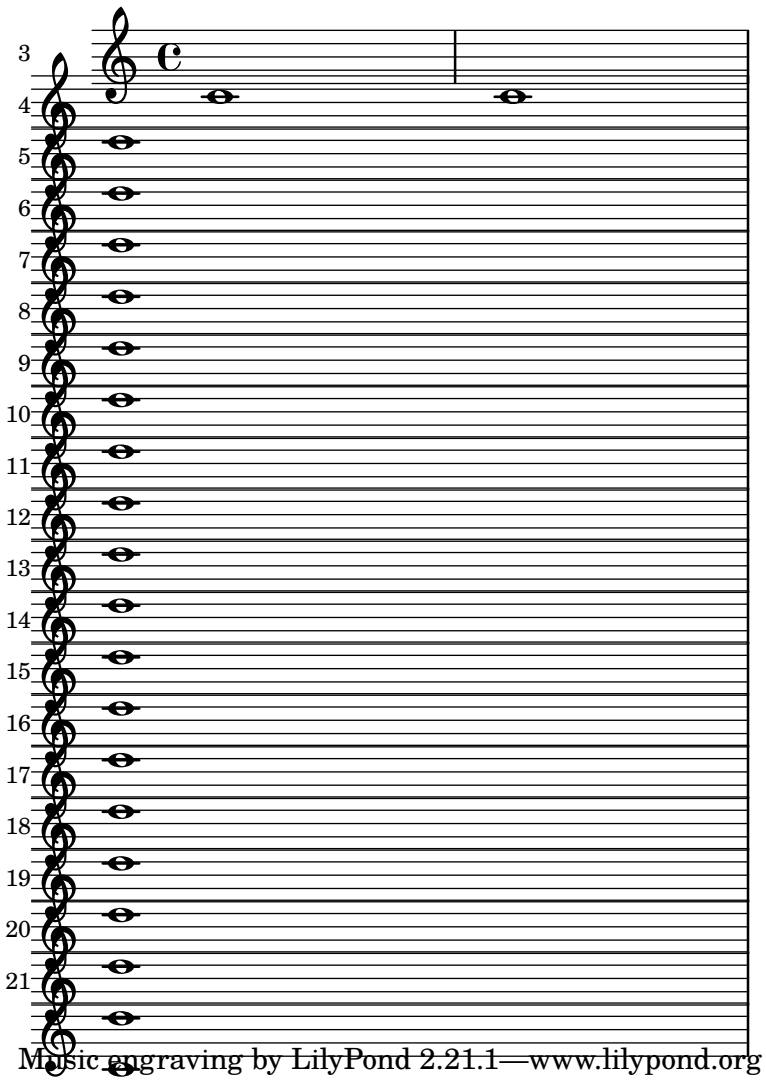




Music engraving by LilyPond 2.21.1—www.lilypond.org

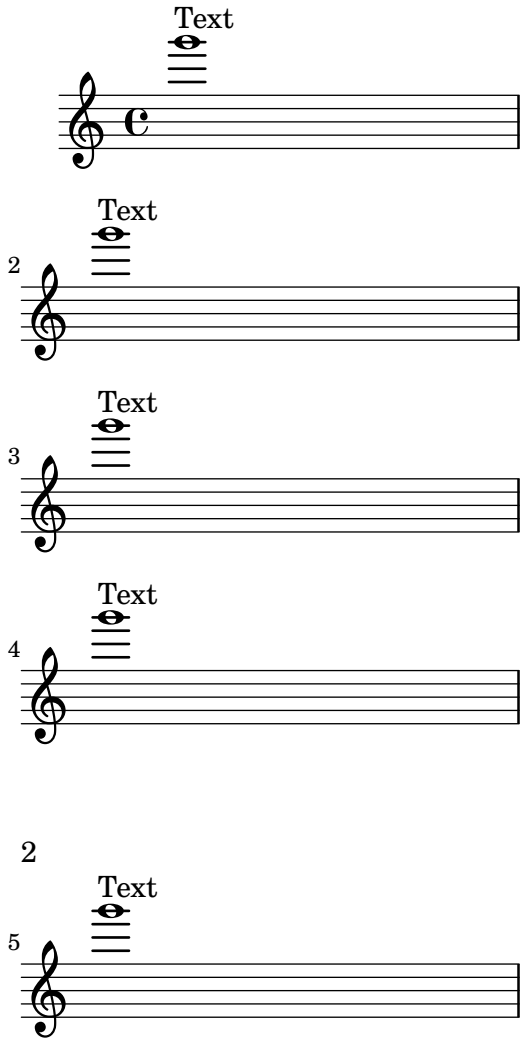
The min-systems-per-page variable takes precedence over the desire not to overfill a page. In this case, systems will overlap because they are forced to be on the page.

page-breaking-min-systems-per-page2.ly



The height-estimation routine takes into account the fact that the TextScript needs to be moved up to avoid the note. This should be spaced on two pages.

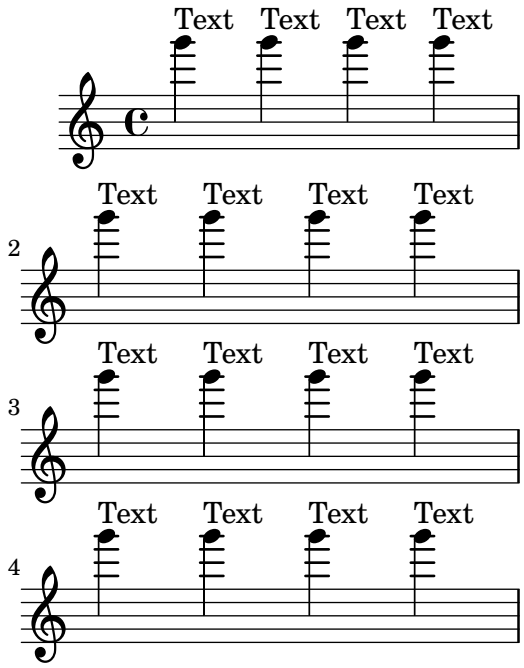
page-breaking-outside-staff-estimation.ly



Music engraving by LilyPond 2.21.1—www.lilypond.org

The height-estimation routine doesn't get confused by multiple outside-staff grobs in the same measure.

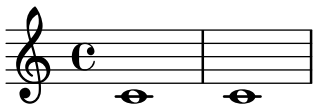
`page-breaking-outside-staff-estimation2.ly`



Music engraving by LilyPond 2.21.1—www.lilypond.org

The number of pages in a score can be forced by setting `page-count` in the (book-level) paper block.

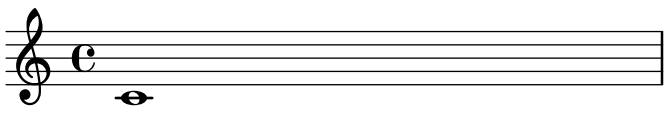
`page-breaking-page-count1.ly`



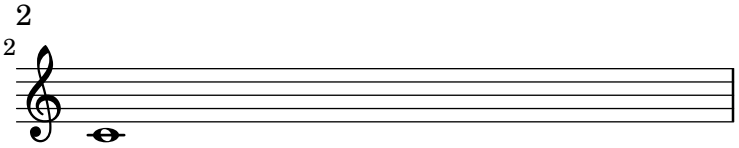
Music engraving by LilyPond 2.21.1—www.lilypond.org

The number of pages in a score can be forced by setting `page-count` in the (book-level) paper block. If there are too few systems for the number of pages, we append blank pages.

`page-breaking-page-count2.ly`
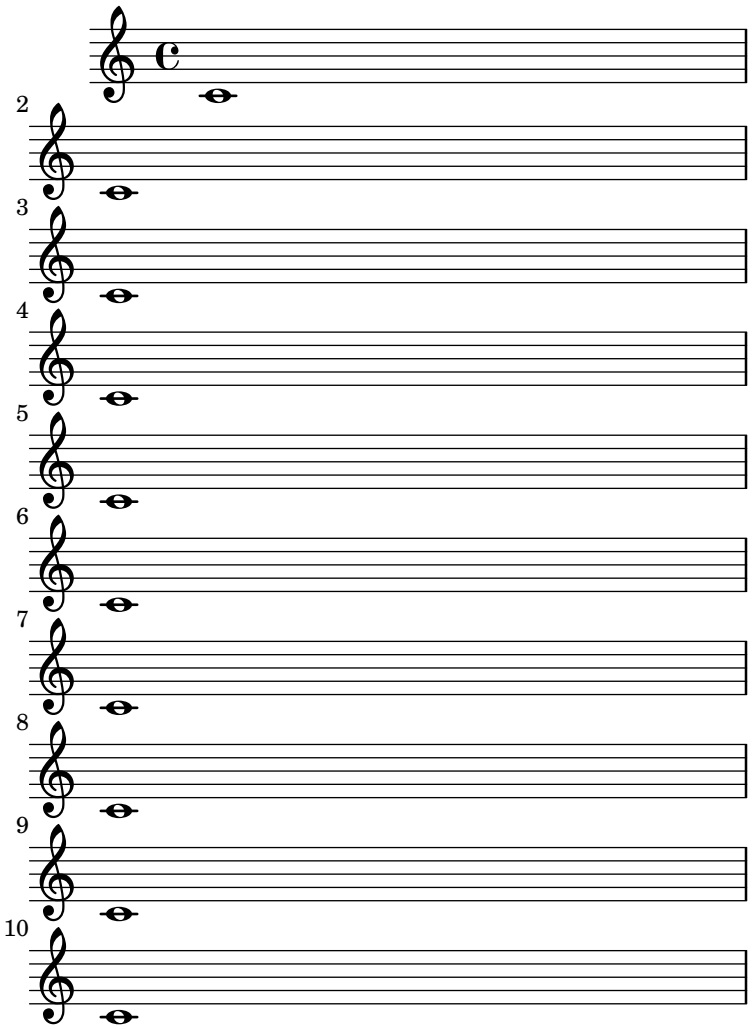


2



Music engraving by LilyPond 2.21.1—www.lilypond.org

The number of pages in a score can be forced by setting `page-count` in the (book-level) paper block. Even if there are too many systems for that number of pages, we will squeeze them in.

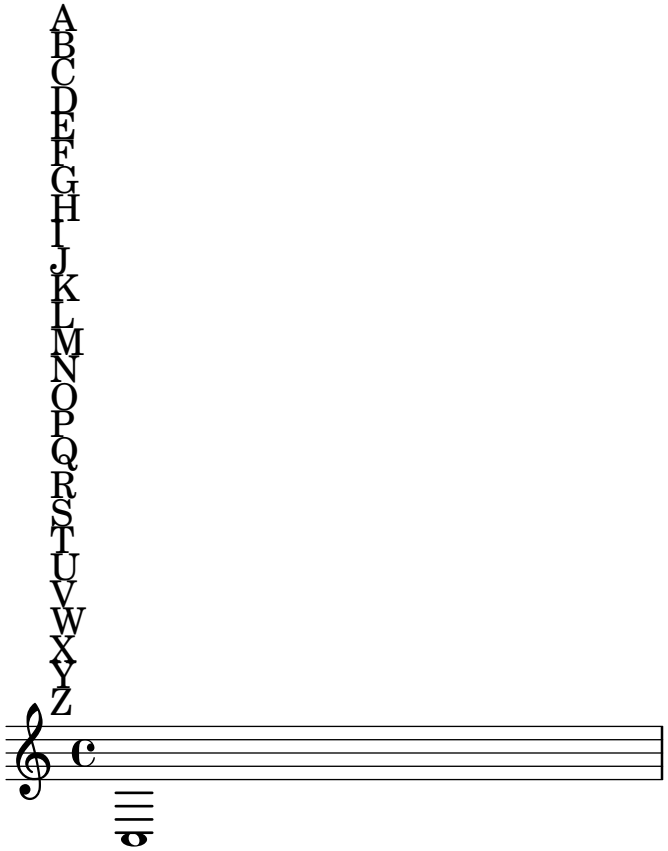
page-breaking-page-count3.ly

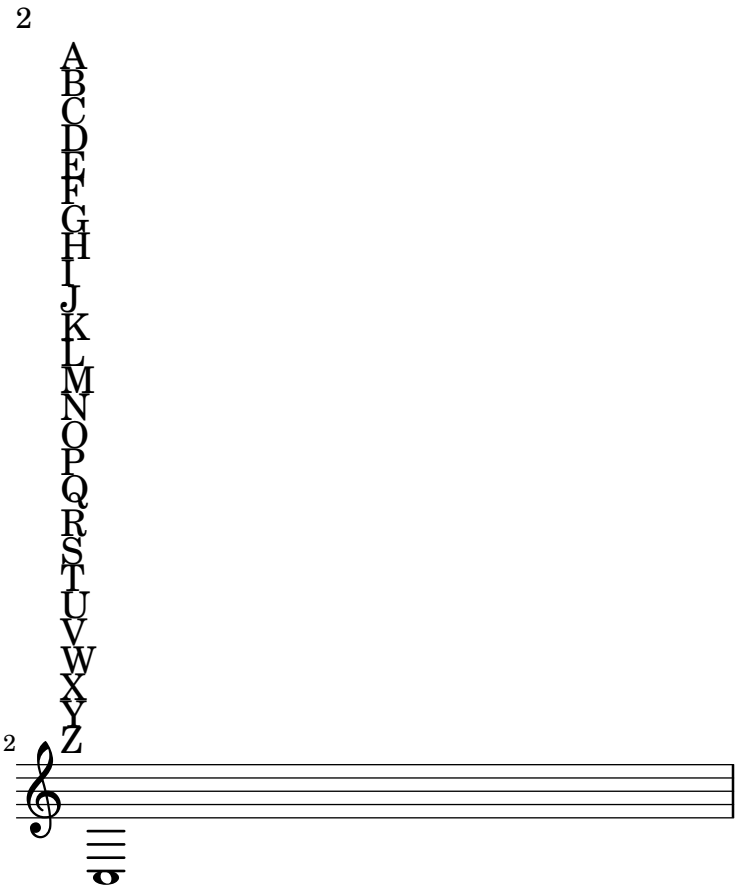


Music engraving by LilyPond 2.21.1—www.lilypond.org

The height of RehearsalMarks is taken into account during page breaking.

page-breaking-rehearsal-mark.ly
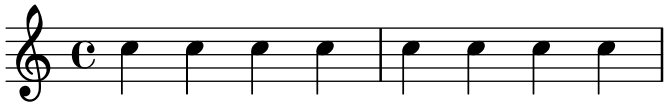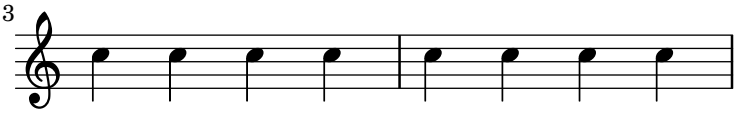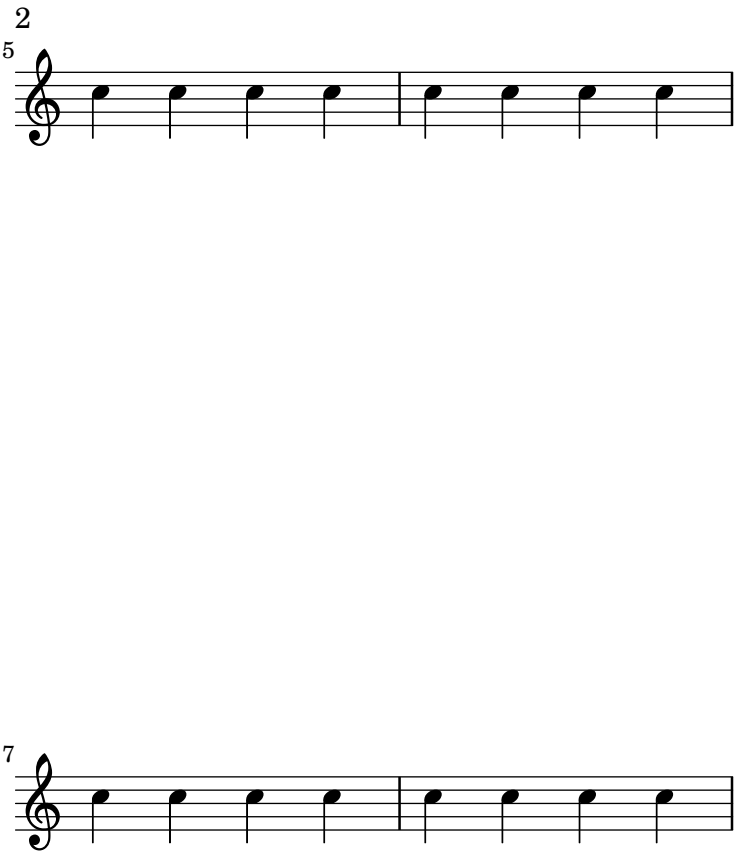




Music engraving by LilyPond 2.21.1—www.lilypond.org

system-count and \pageBreak are compatible.

page-breaking-system-count-forced-break.ly



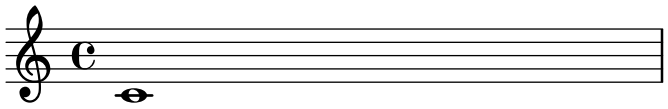


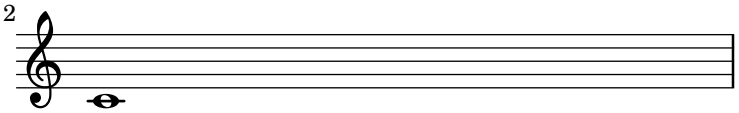


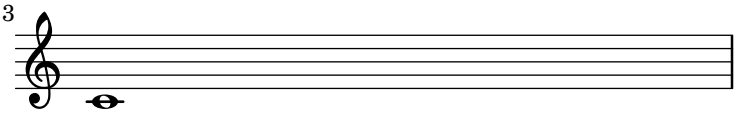
Music engraving by LilyPond 2.21.1—www.lilypond.org

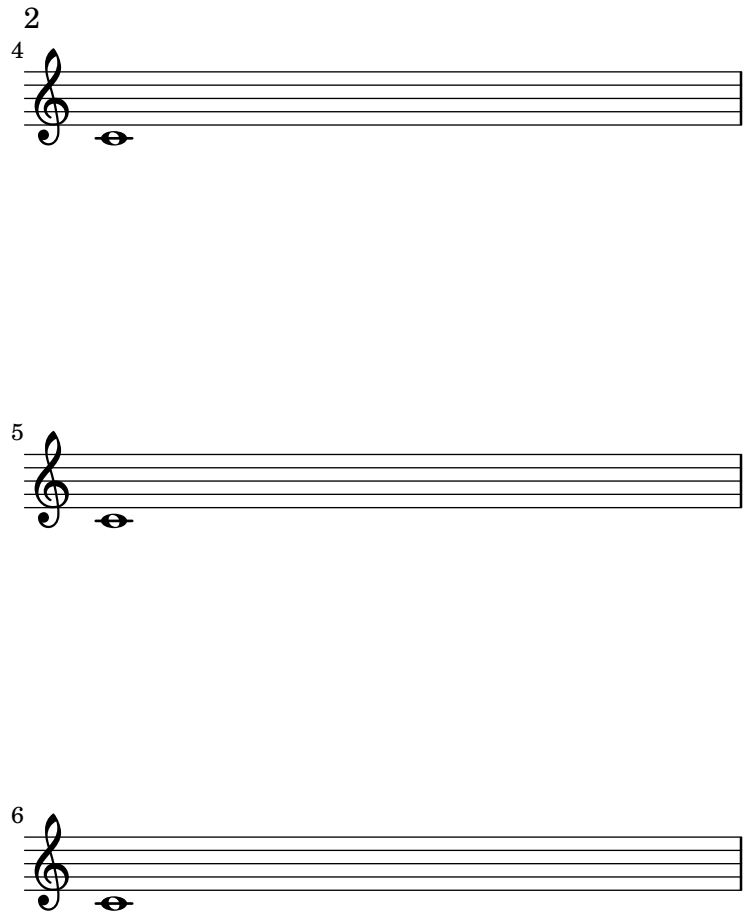
The systems-per-page variable forces a certain number of systems per page. Titles are not counted as systems.

Title









Music engraving by LilyPond 2.21.1—www.lilypond.org

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

page-breaks.ly

Title

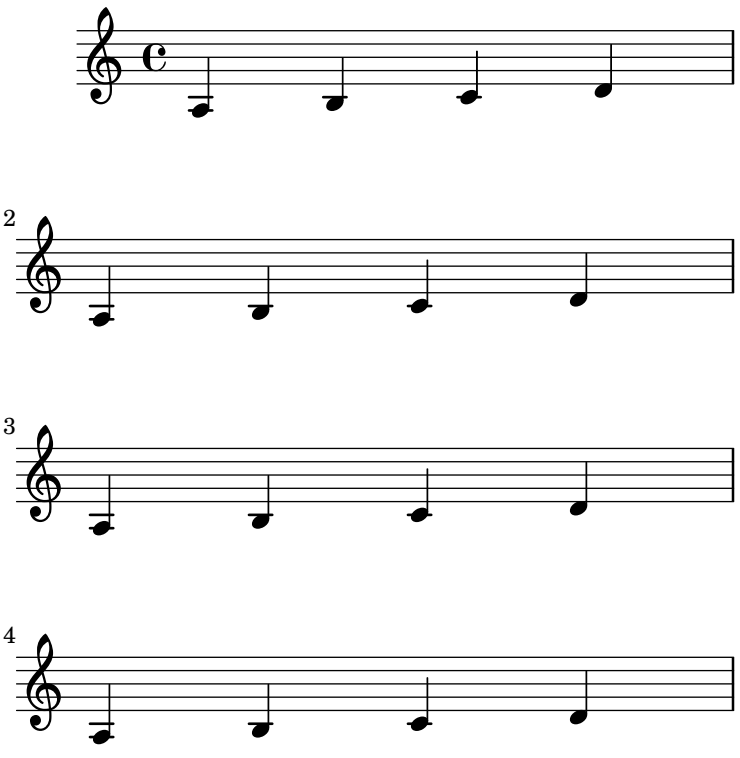
(and (the) subtitle)

Sub sub title
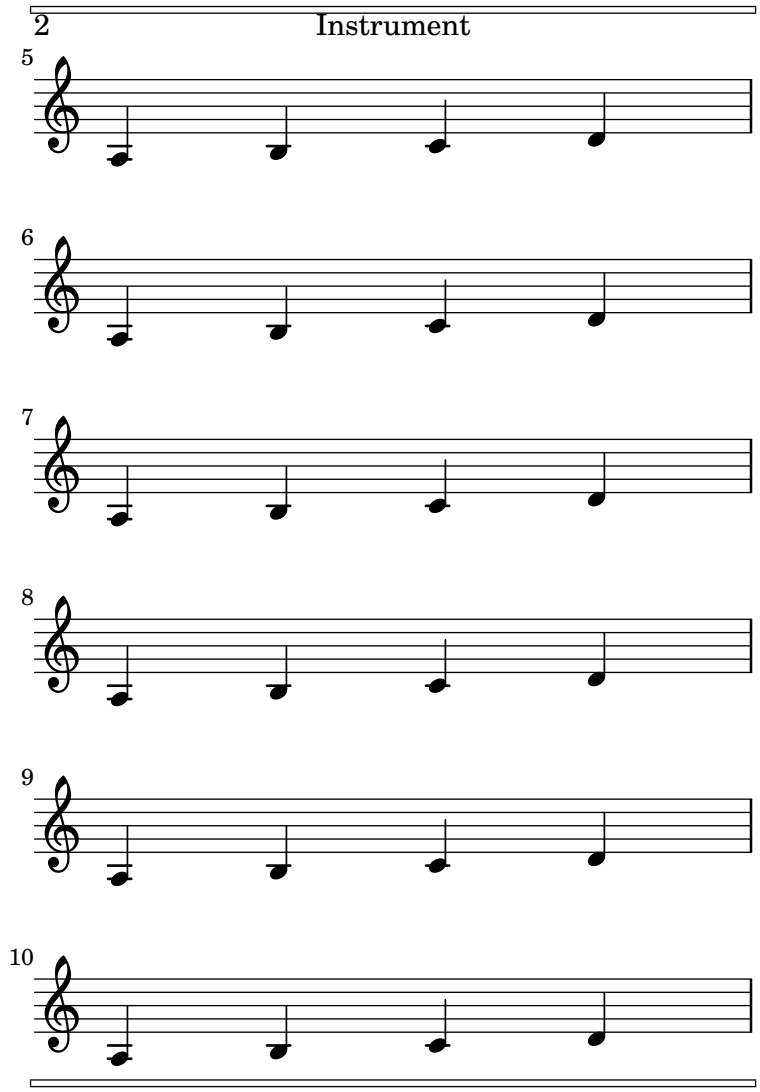
Poet **Instrument** Composer
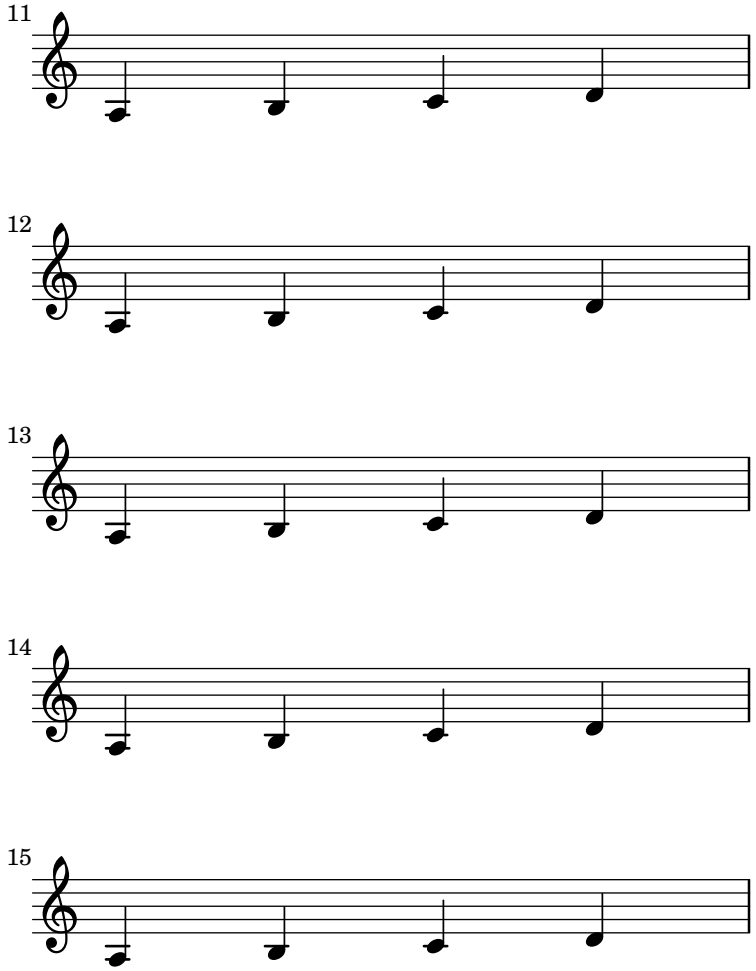
Meter (huh?) Arranger

Piece opus 0
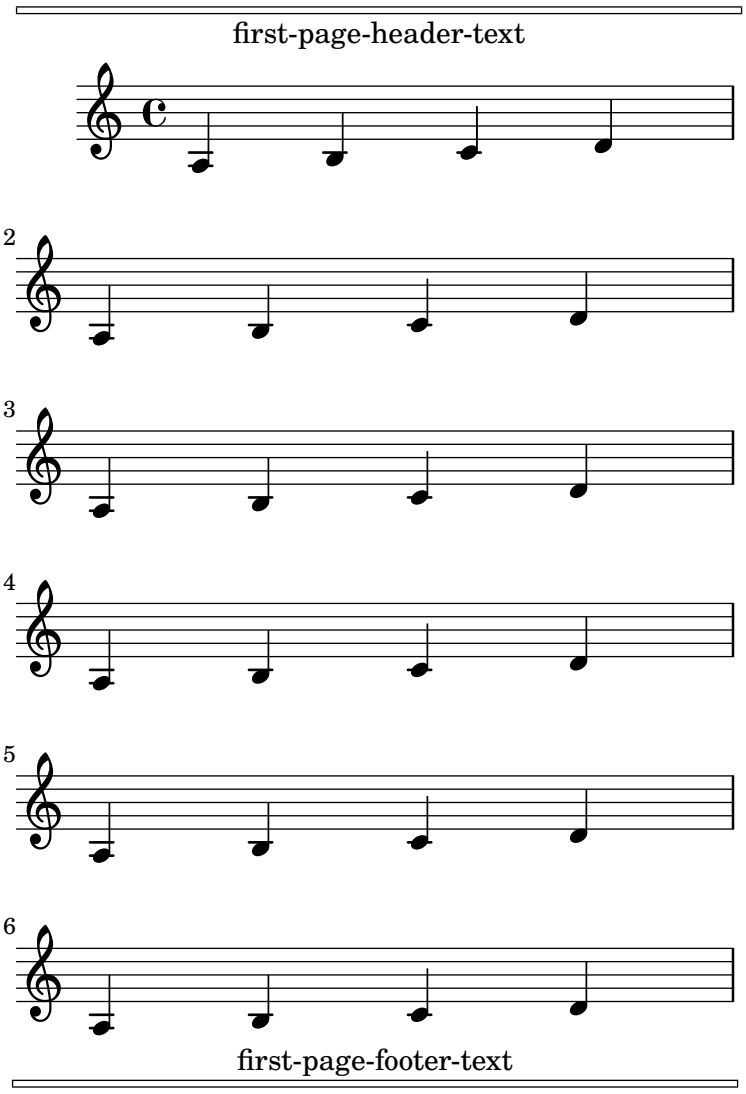


Copyright by /me

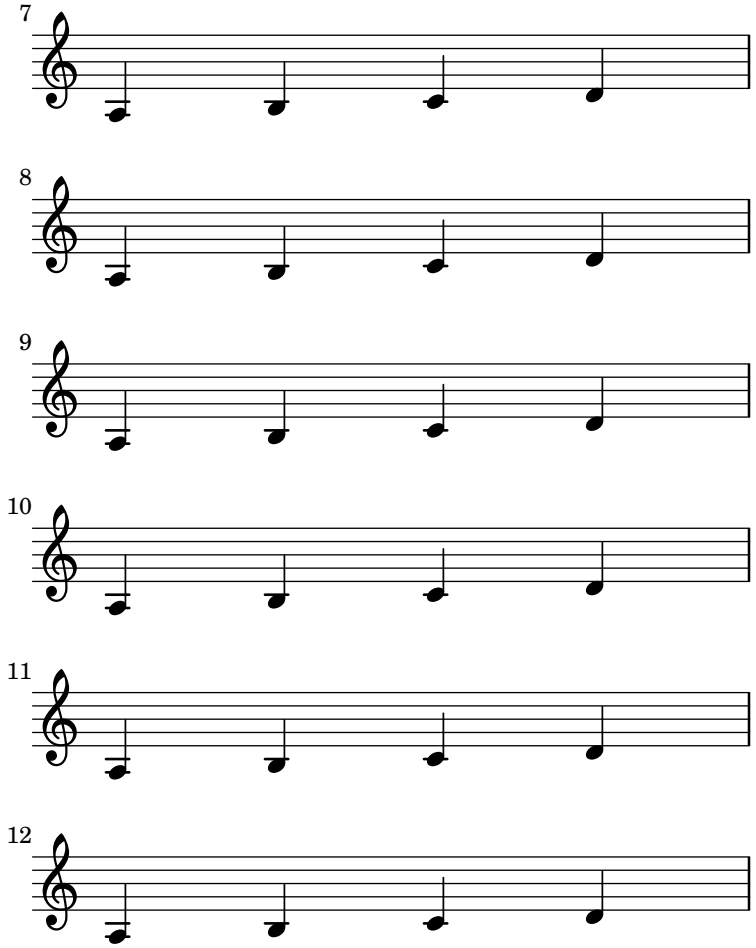

2
Instrument
5
6
7
8
9
10

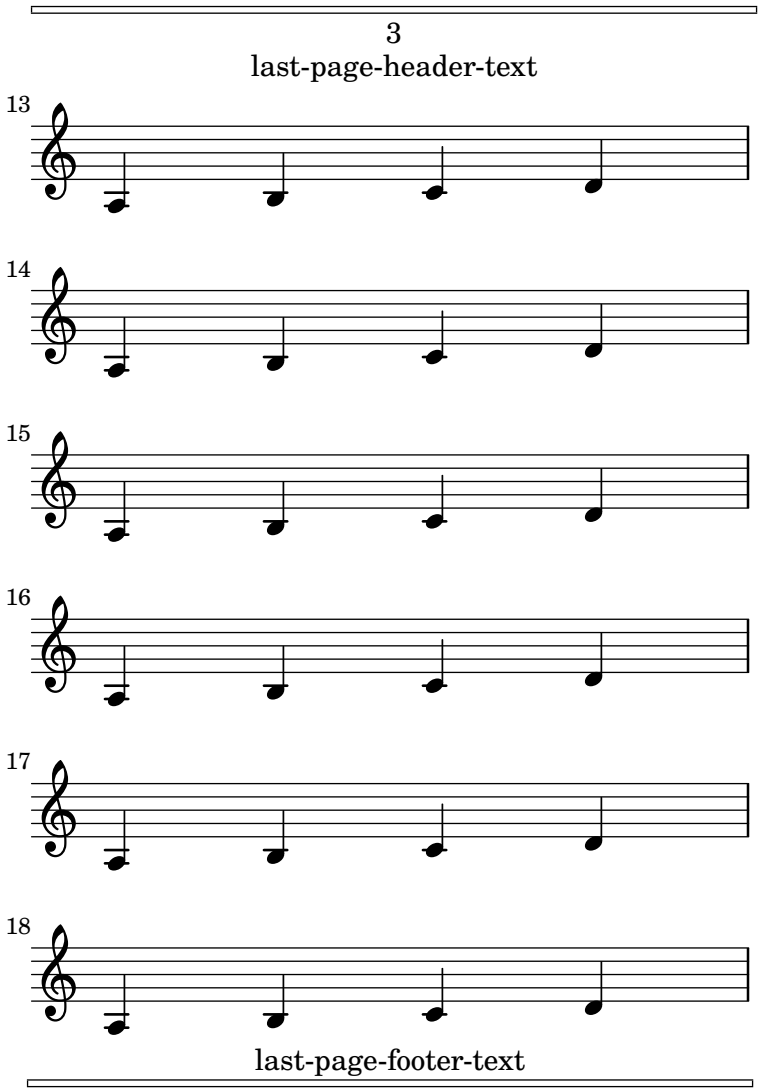


Page-headers and -footers. All headers and footers should be printed on their specified page.

page-headers-and-footers.ly




7
8
9
10
11
12

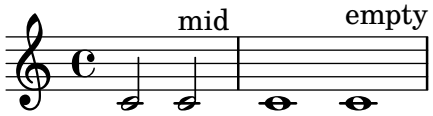


Page labels on loose columns are not ignored: this includes both mid-line unbreakable columns which only contain labels and columns with empty bar lines (and no other break-aligned grobs).

page-label-loose-column.ly

Table of Contents

Mid-line 1
Empty bar line 1



Music engraving by LilyPond 2.21.1—www.lilypond.org

Page labels may be placed inside music or at top-level, and referred to in markups.

page-label.ly

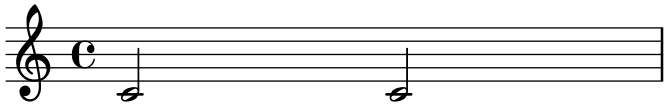
Title Page

2
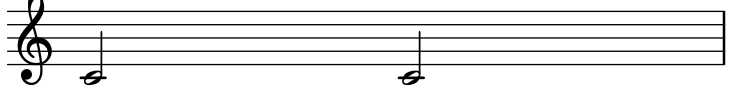
Table of contents
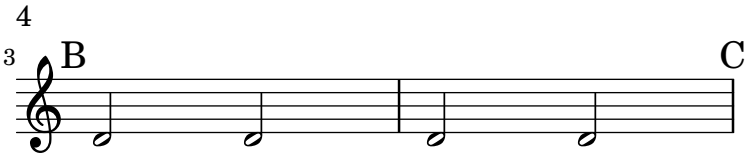
Table of contents 2
First Score 3
Mark A 3
Mark B 4
Mark C 4
Unknown label ?

First score



2 A (page 3)

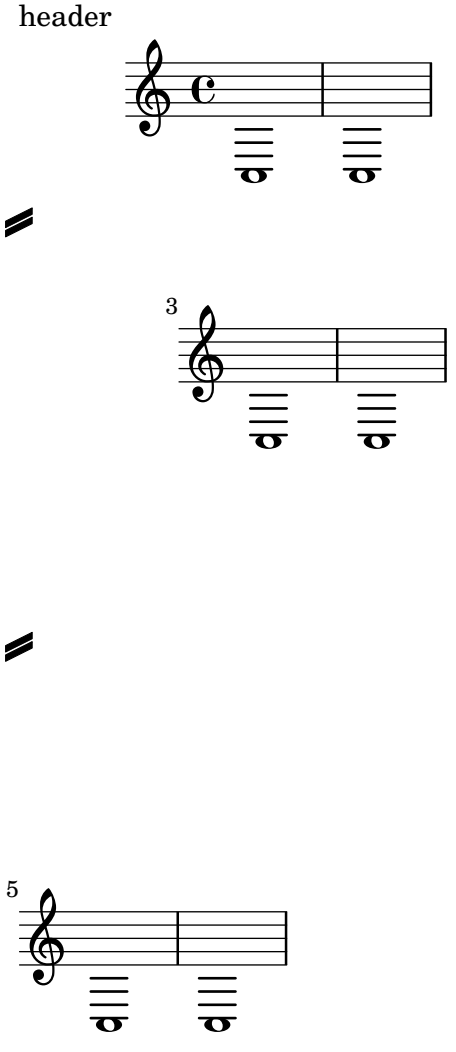




Music engraving by LilyPond 2.21.1—www.lilypond.org

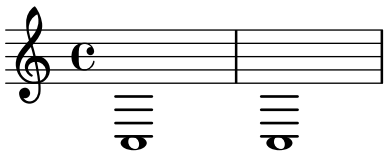
By setting `extra-offset` within the `line-break-system-details` of `NonMusicalPaperColumn`, systems may be moved in relation either to their default position on the printable area of the page or the absolute position specified by `X-offset` or `Y-offset` within `line-break-system-details`.

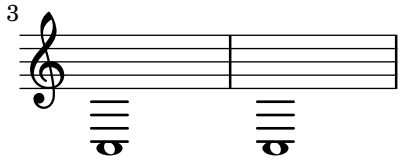
`page-layout-extra-offset.ly`



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-manual-position.ly



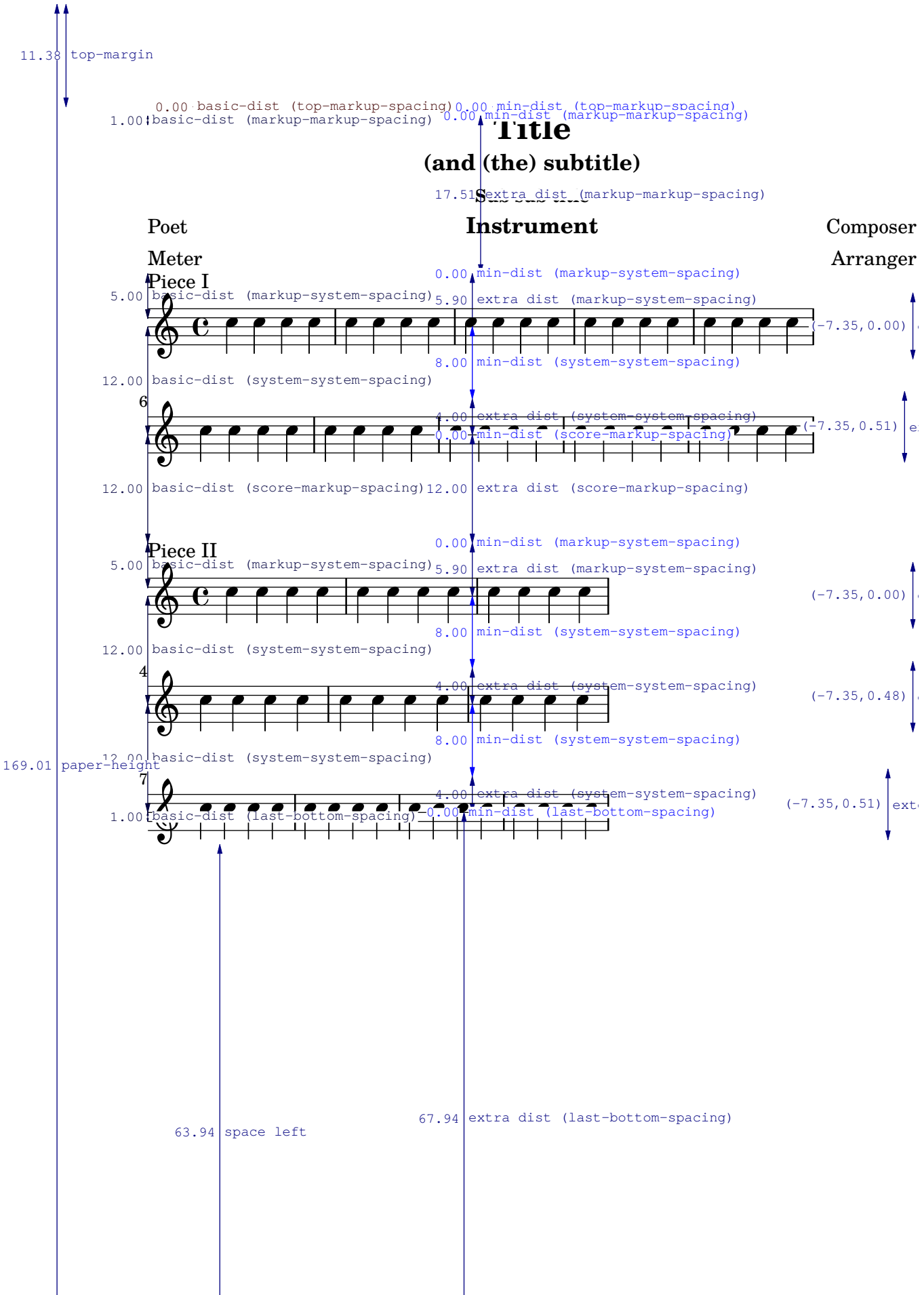


this is the tagline

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-layout.ly



Links to labels should not break if the label doesn't exist.

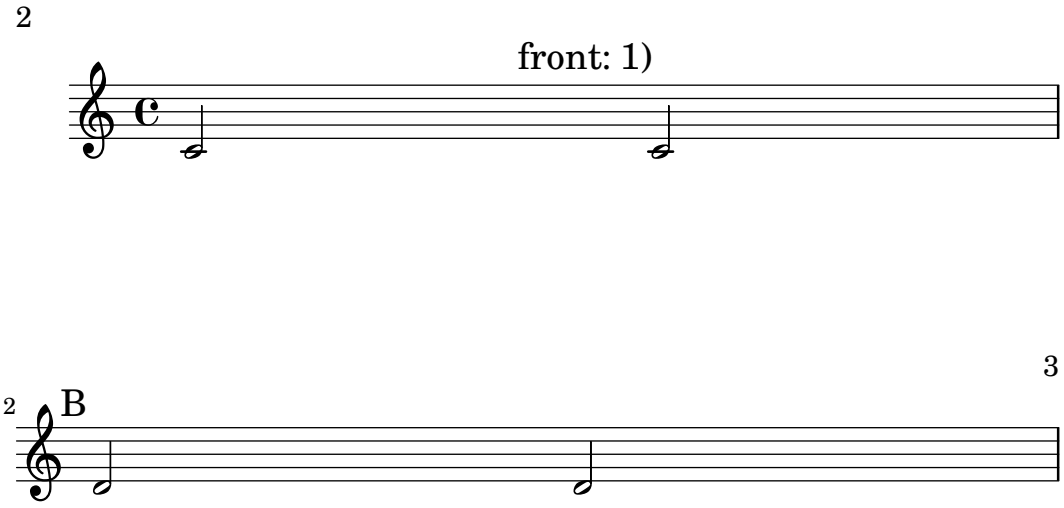
page-links-nolabel.ly

Link to non-existing label

Links to labels and explicit page number (PDF backend only).

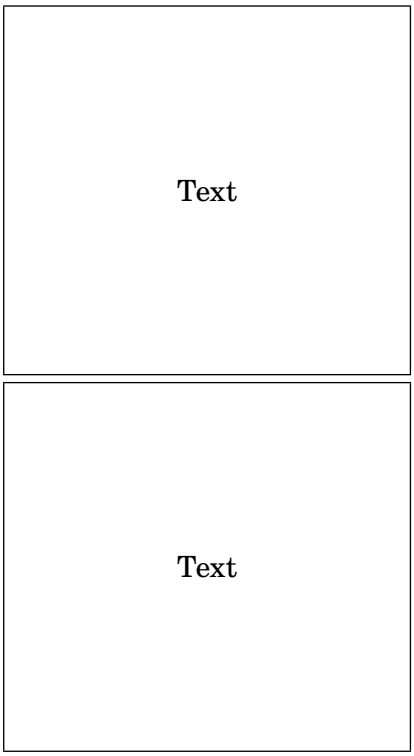
`page-links.ly`

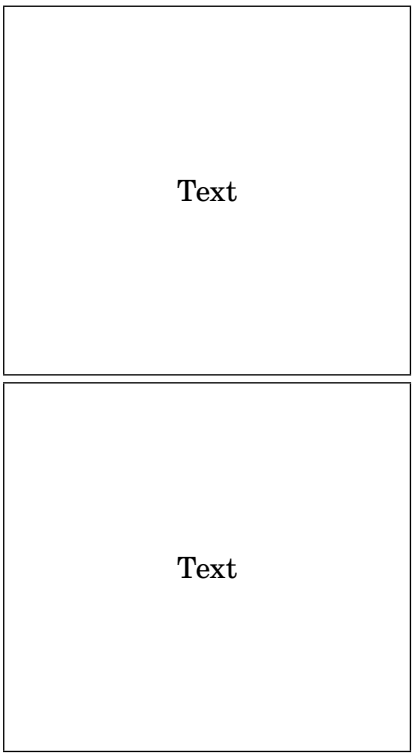
Link to page 2 with label #'second.
Explicit link to page 3
Link to mark B

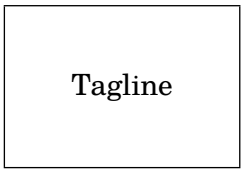


Minimal page breaker: special case when the last system is moved to an other page when there is not enough space because of the tagline.

page-minimal-page-breaking-last-page.ly

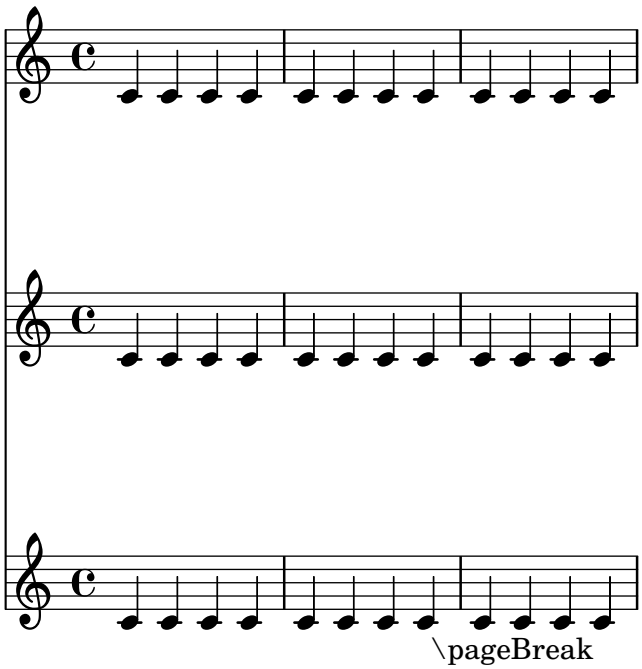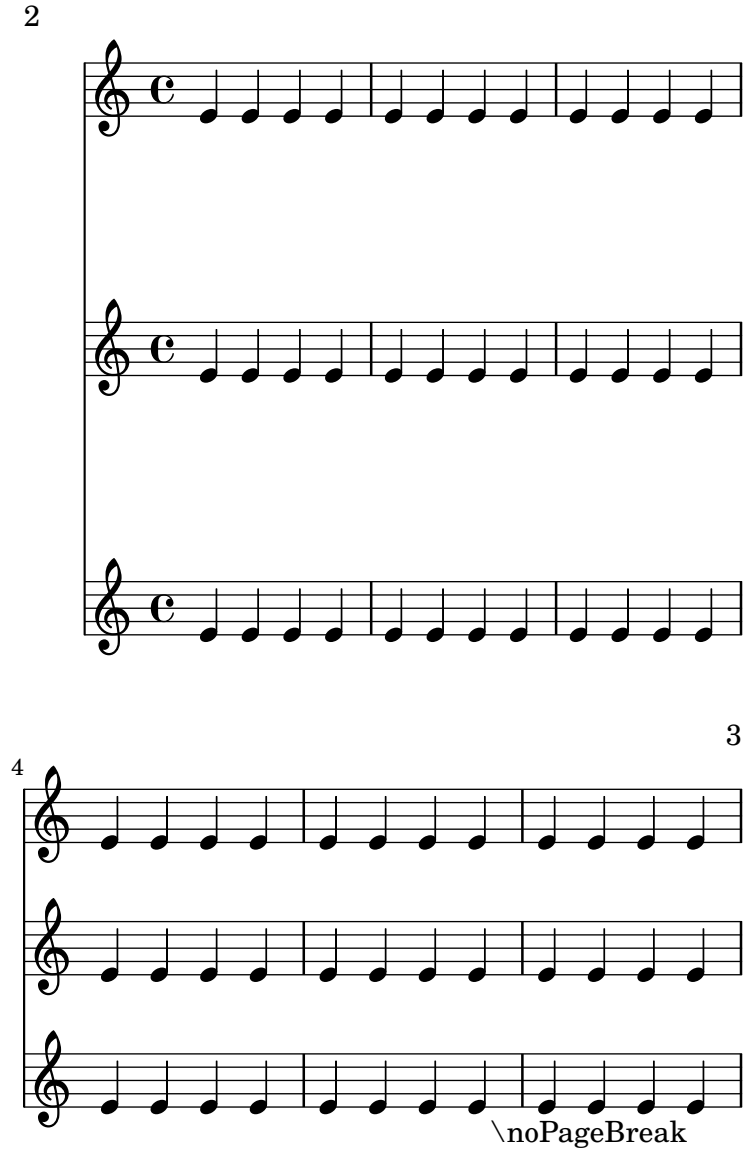






The minimal page breaker stacks as many lines on pages, only accounting for manual page break commands.

`page-minimal-page-breaking.ly`

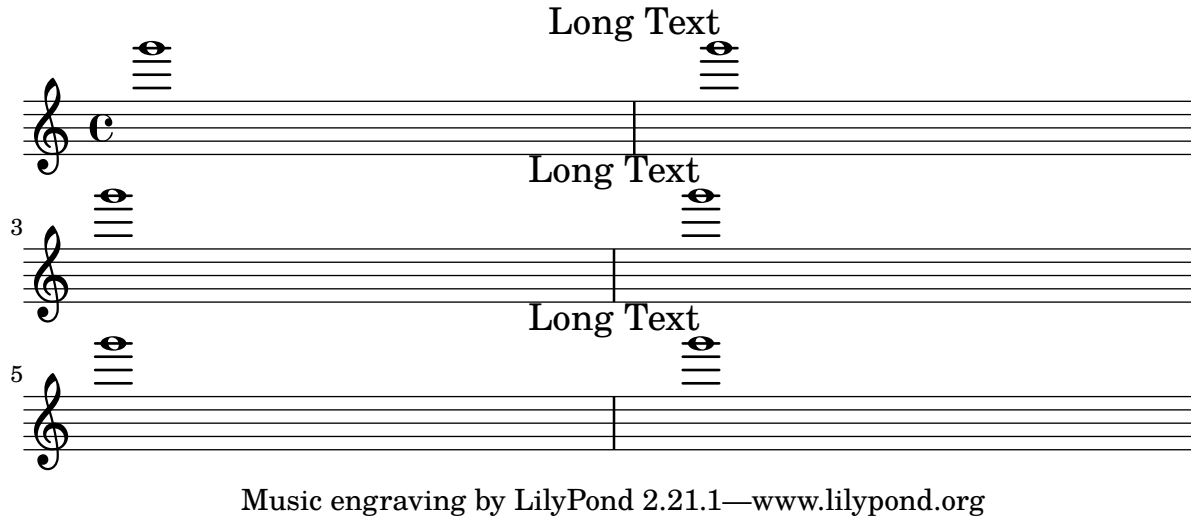




Music engraving by LilyPond 2.21.1—www.lilypond.org
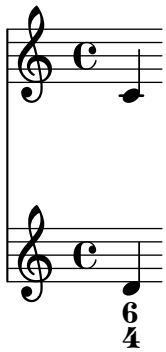
Layouts that overflow a page will be compressed in order to fit on the page, even if it causes collisions. In this example, the tagline should not collide with the bottom staff.

`page-overflow-compression.ly`



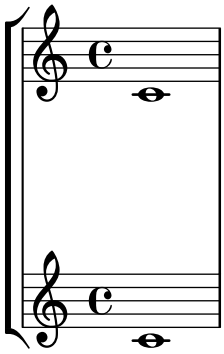
alignment-distances applies to the toplevel VerticalAlignment but not to BassFigureAlignment. The 4 in the bass figure line should be directly below the 6.

`page-spacing-bass-figures.ly`



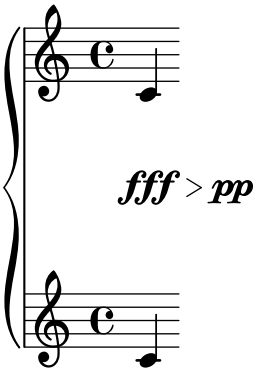
The spring at the bottom of a page is fairly flexible (much more so than the one at the top), so it does not drag the staff to the bottom of the page. However, it is sufficiently stiff to cause stretching.

page-spacing-bottom-spring.ly



Dynamic centering still works with alignment-distances.

page-spacing-dynamics.ly



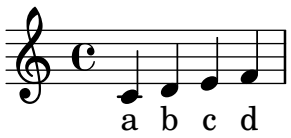
Adjacent lines of markup are placed as closely together as possible.

A
B
C
D
E

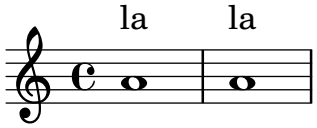
Music engraving by LilyPond 2.21.1—www.lilypond.org

Having markup after a non-staff line doesn't confuse the page layout engine.

page-spacing-nonstaff-lines-and-markup.ly



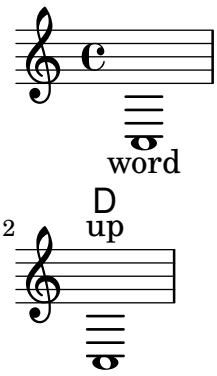
next song



Music engraving by LilyPond 2.21.1—www.lilypond.org

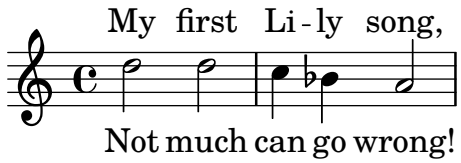
The vertical spacing engine is not confused by a non-staff line below a system followed by a loose line above the next system. Systems are spaced far enough that loose lines are not interleaved, even if gaps would allow interleaving.

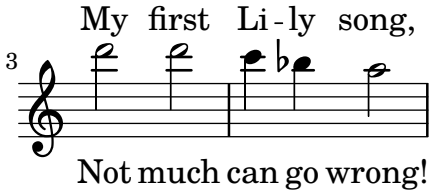
page-spacing-nonstaff-lines-between-systems.ly



Non-staff lines between two systems don't confuse the layout engine. In particular, they don't interfere with `system-system-spacing`, which controls the flexible spacing between the two closest staves of consecutive systems.
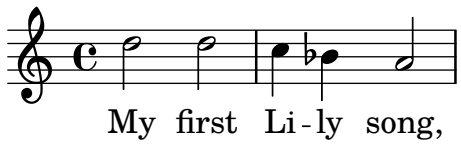
`page-spacing-nonstaff-lines-between.ly`





A non-staff line (such as `Lyrics`) at the bottom of a system gets spaced appropriately.

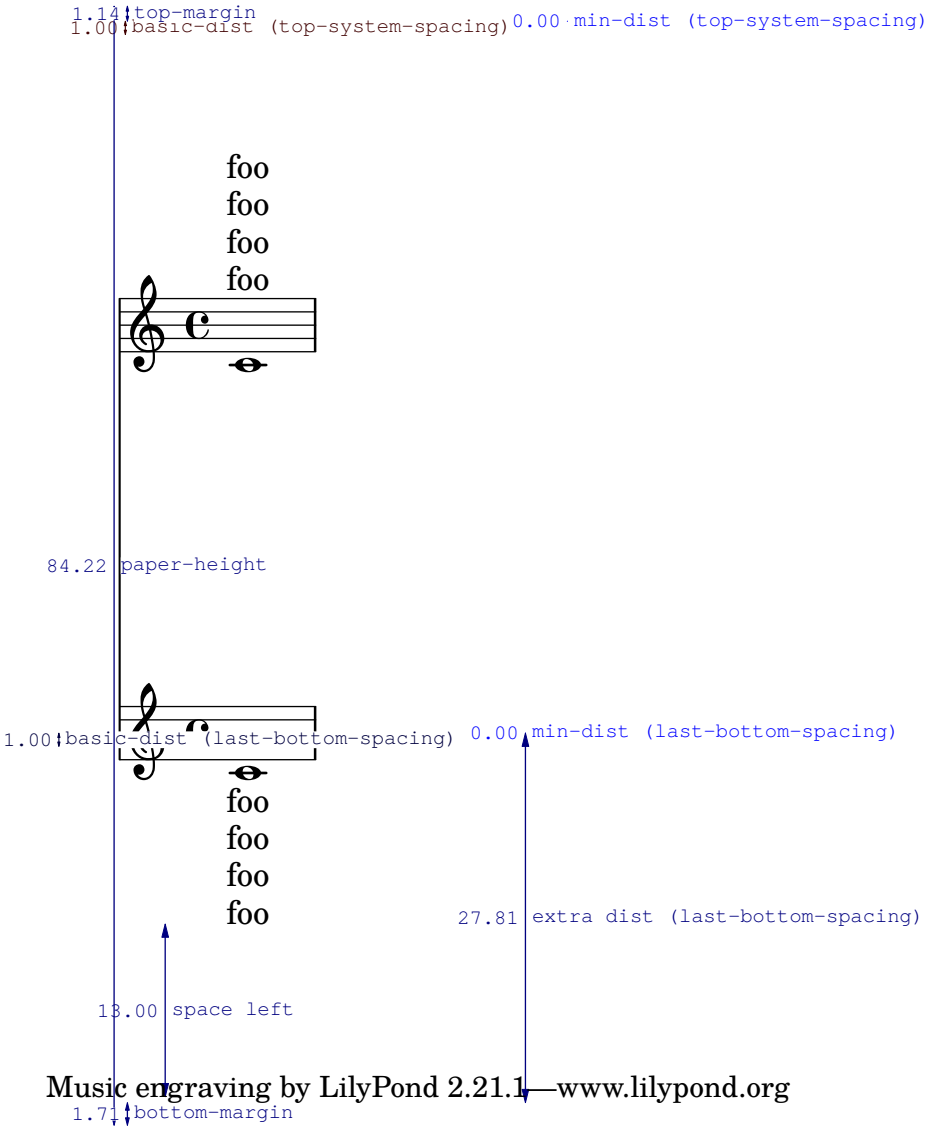
`page-spacing-nonstaff-lines-bottom.ly`



Not much can go wrong!

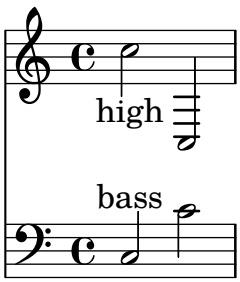
Padding from the header and footer is measured to the first non-staff line, whether or not it is spaceable.

`page-spacing-nonstaff-lines-header-padding.ly`



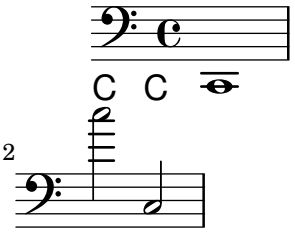
Spacing rules between Staves coexist with rules affecting non-staff lines. Here, the `padding` separating items on different staves is larger than the `padding` for associated lyrics.

`page-spacing-nonstaff-lines-independent.ly`



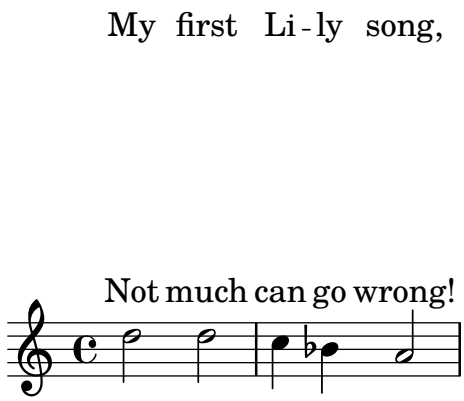
Relative indentation between systems is taken into account in allowing space for loose lines between systems.

page-spacing-nonstaff-lines-skylines.ly



A non-staff line (such as `Lyrics`) at the top of a system is spaced appropriately.

`page-spacing-nonstaff-lines-top.ly`



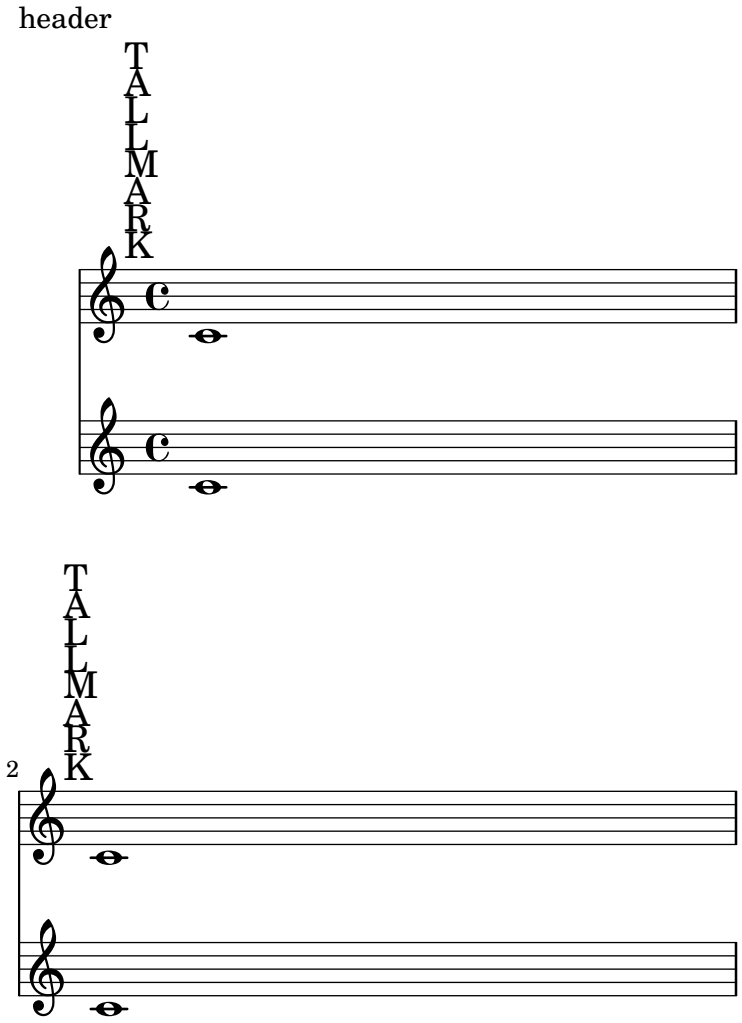
Non-staff lines (such as `Lyrics`) can specify their `padding` or `minimum-distance` to the staff for which they don't have affinity.

`page-spacing-nonstaff-lines-unrelated.ly`



The space taken up by rehearsal marks is correctly accounted for, even though they live in the Score context.

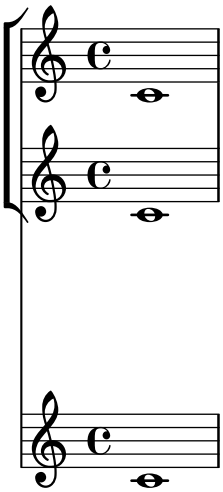
page-spacing-rehearsal-mark.ly

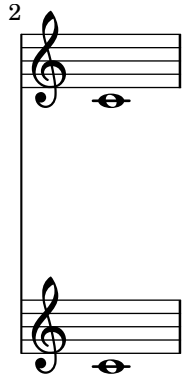


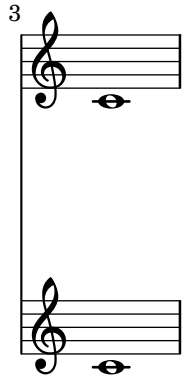
Music engraving by LilyPond 2.21.1—www.lilypond.org

StaffGrouper interacts correctly with \RemoveEmptyStaffContext. In both systems, there should be a large space between the staff groups.

page-spacing-staff-group-hara-kiri.ly

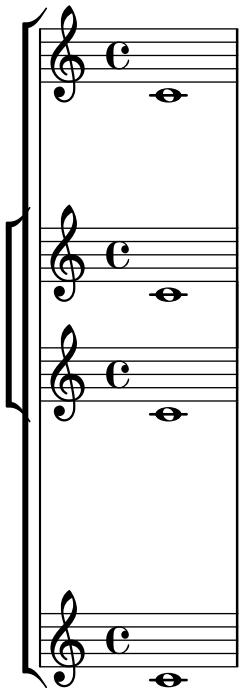






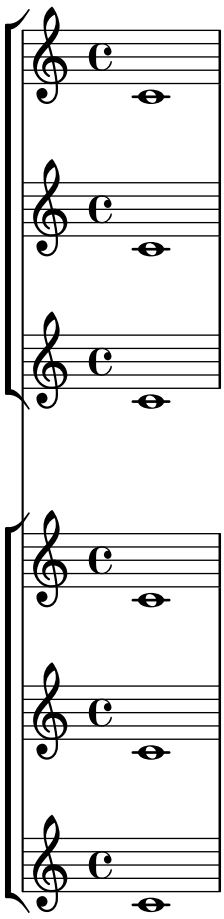
StaffGroups can be nested, in which case the inner StaffGroup wins.

`page-spacing-staff-group-nested.ly`



By default, the staves within a StaffGroup are spaced more closely than staves not in a StaffGroup.

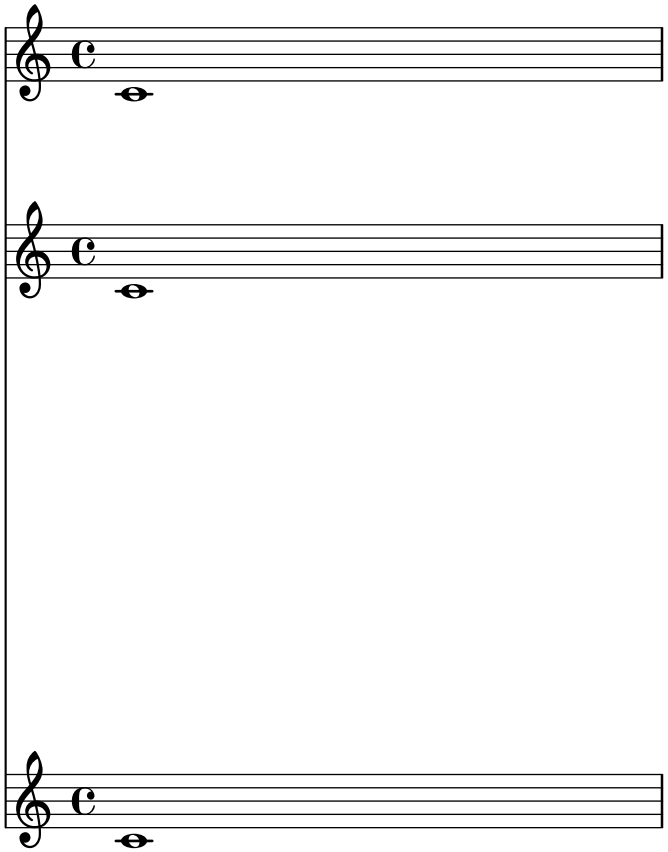
page-spacing-staff-group.ly

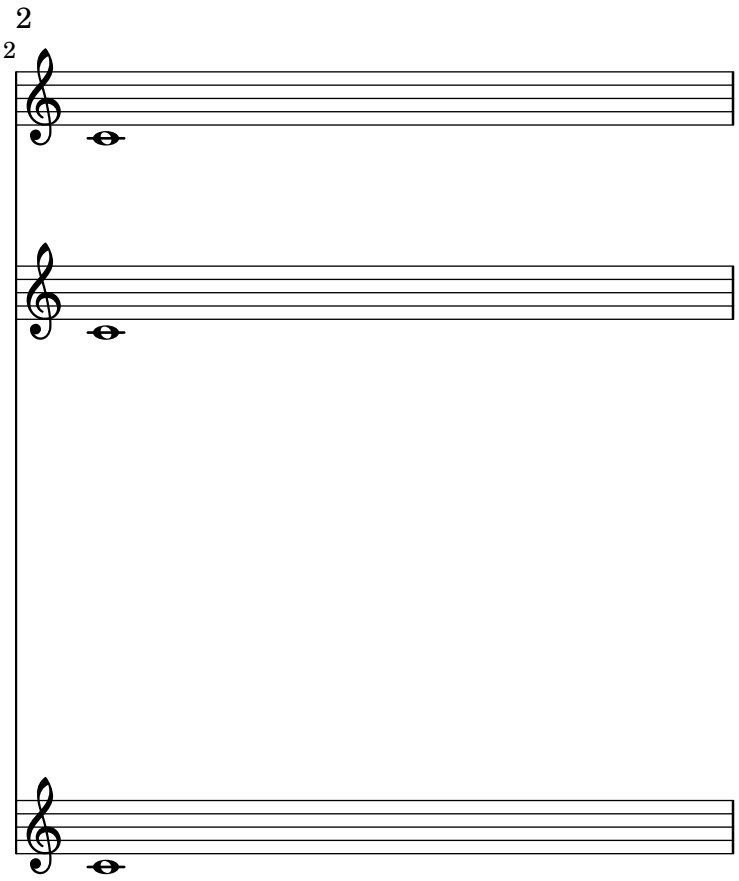


Music engraving by LilyPond 2.21.1—www.lilypond.org

The stretchability property affects the amount that staves will move under extreme stretching, but it does not affect the default distance between staves.

page-spacing-stretchability.ly
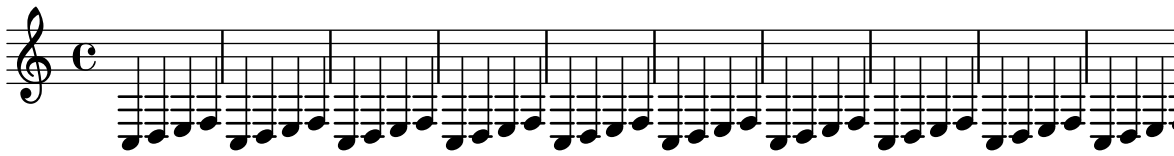




Music engraving by LilyPond 2.21.1—www.lilypond.org

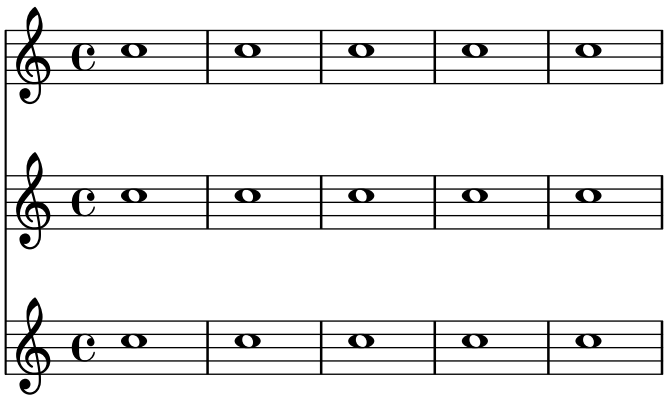
Page breaking doesn't crash when the line-breaking is invalid.

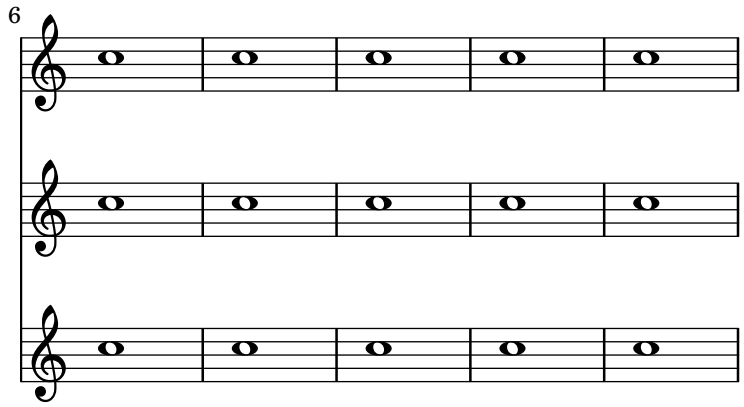
page-spacing-system-count-overfull.ly



Page layout and stretching work with system-count enabled.

`page-spacing-system-count.ly`

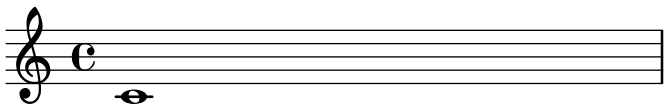


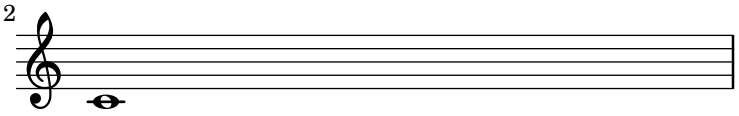


Music engraving by LilyPond 2.21.1—www.lilypond.org

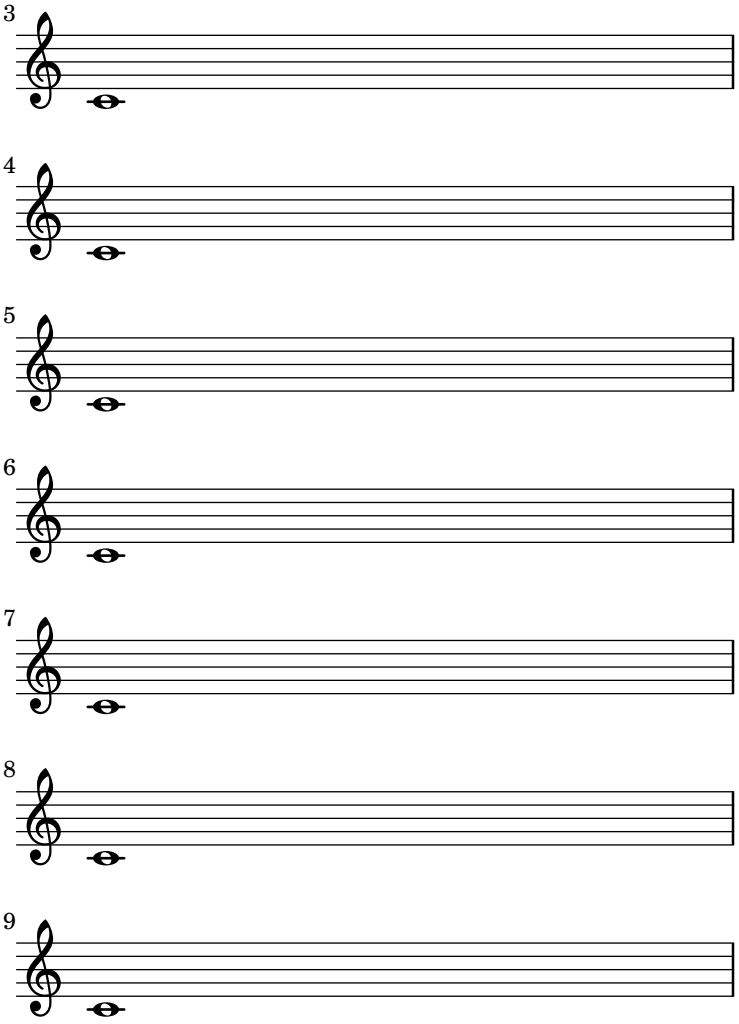
Both the page breaking and the page layout take account of the heights of the header and footer.

page-spacing-tall-headfoot.ly

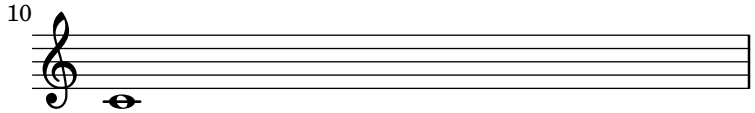




2
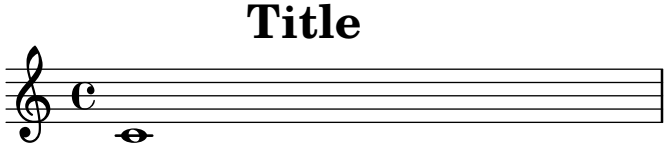


t
a
l
l
h
e
a
d
e
r

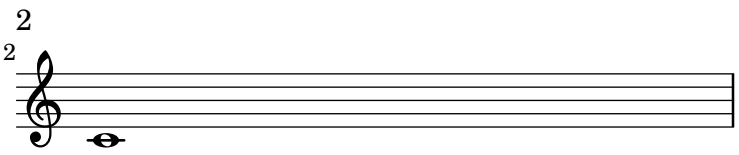


t
a
l
l
f
o
o
t
e
r

`top-markup-spacing` controls the spacing from the top of the printable area (i.e. the bottom of the top margin) to a title or markup, when it is the first item on a page.

`page-spacing-top-markup-spacing.ly`
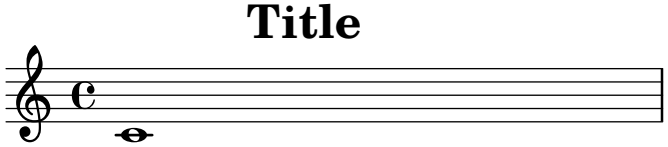




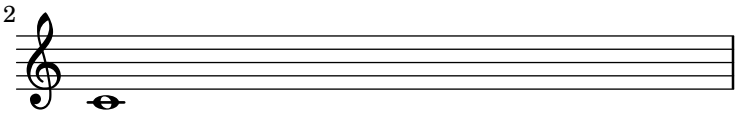
Music engraving by LilyPond 2.21.1—www.lilypond.org

top-system-spacing controls the spacing to the first non-title staff on every page.

`page-spacing-top-system-spacing.ly`



2



Music engraving by LilyPond 2.21.1—www.lilypond.org

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.

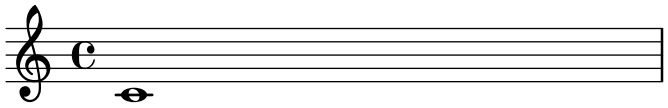
By setting `annotate-spacing`, we can see the effect of each property.

page-spacing.ly

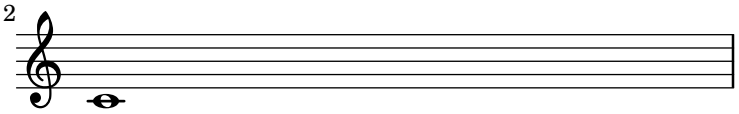


By setting `page-top-space,` the Y position of the first system can be forced to be uniform.

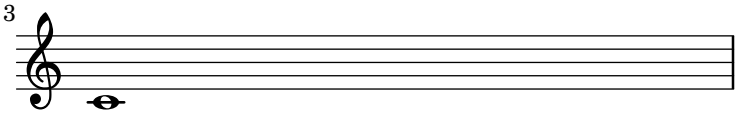
`page-top-space.ly`



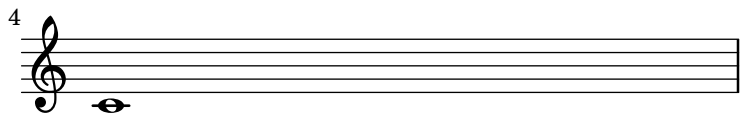
2



3
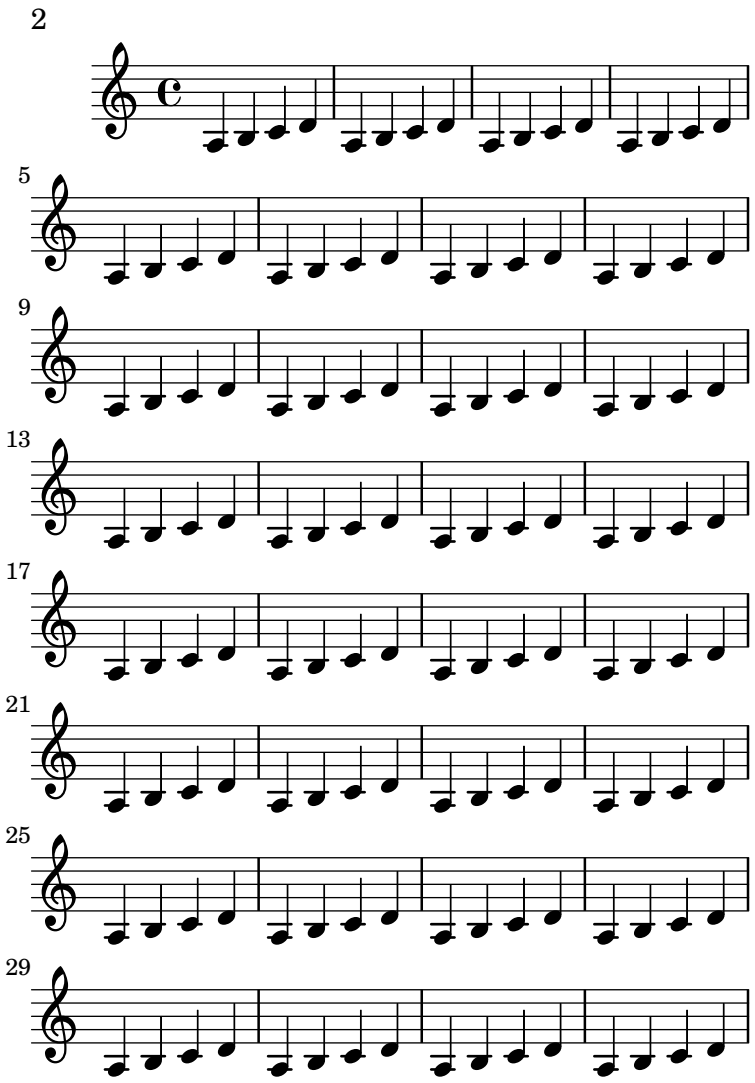


bla
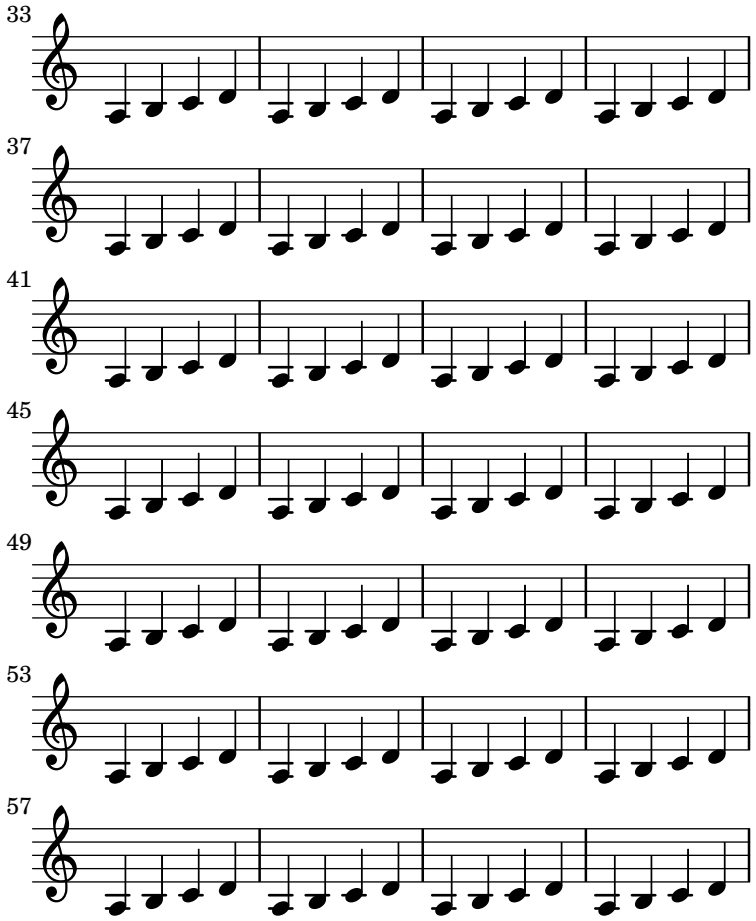


Music engraving by LilyPond 2.21.1—www.lilypond.org

By default, we start with page 1, which is on the right hand side of a double page. In this example, auto-first-page-number is set to ##t and the music won't fit on a single page, so we should automatically set the first page number to 2 in order to avoid a bad page turn.
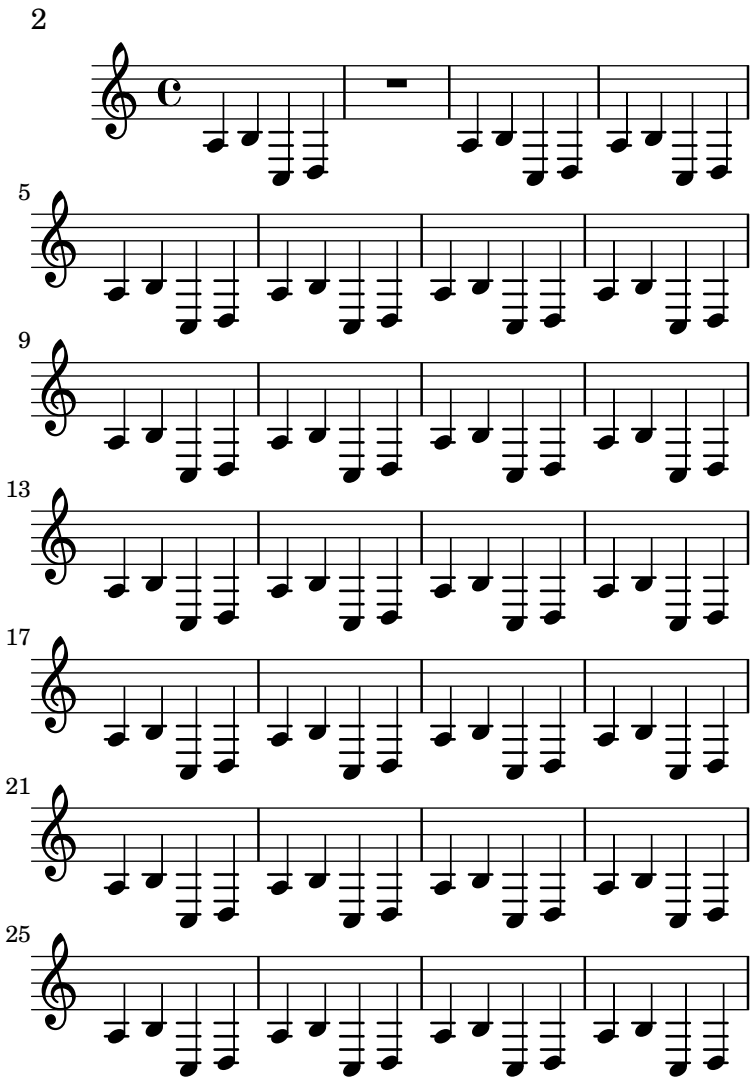
page-turn-page-breaking-auto-first-page.ly
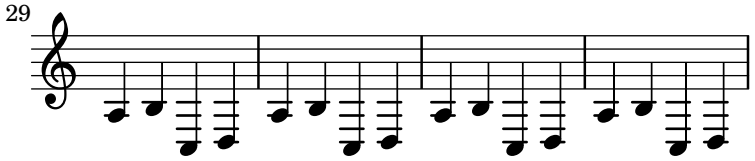




Music engraving by LilyPond 2.21.1—www.lilypond.org

By default, we start with page 1, which is on the right hand side of a double page. In this example, auto-first-page-number is set to ##t. Although the first measure could go on a page by itself, this would require stretching the first page badly, so we should automatically set the first page number to 2 in order to avoid a bad page turn.

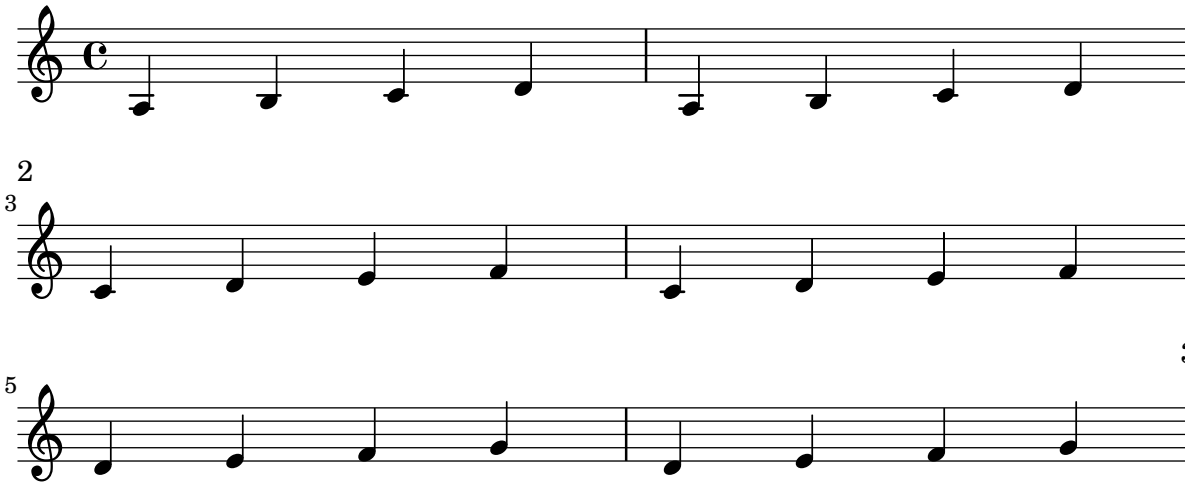

2
5
9
13
17
21
25



Music engraving by LilyPond 2.21.1—www.lilypond.org

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.

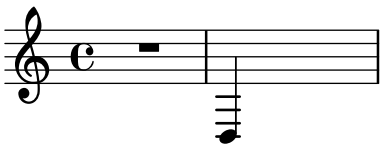
`page-turn-page-breaking-badturns.ly`



Music engraving by LilyPond 2.21.1—www.lilypond.org
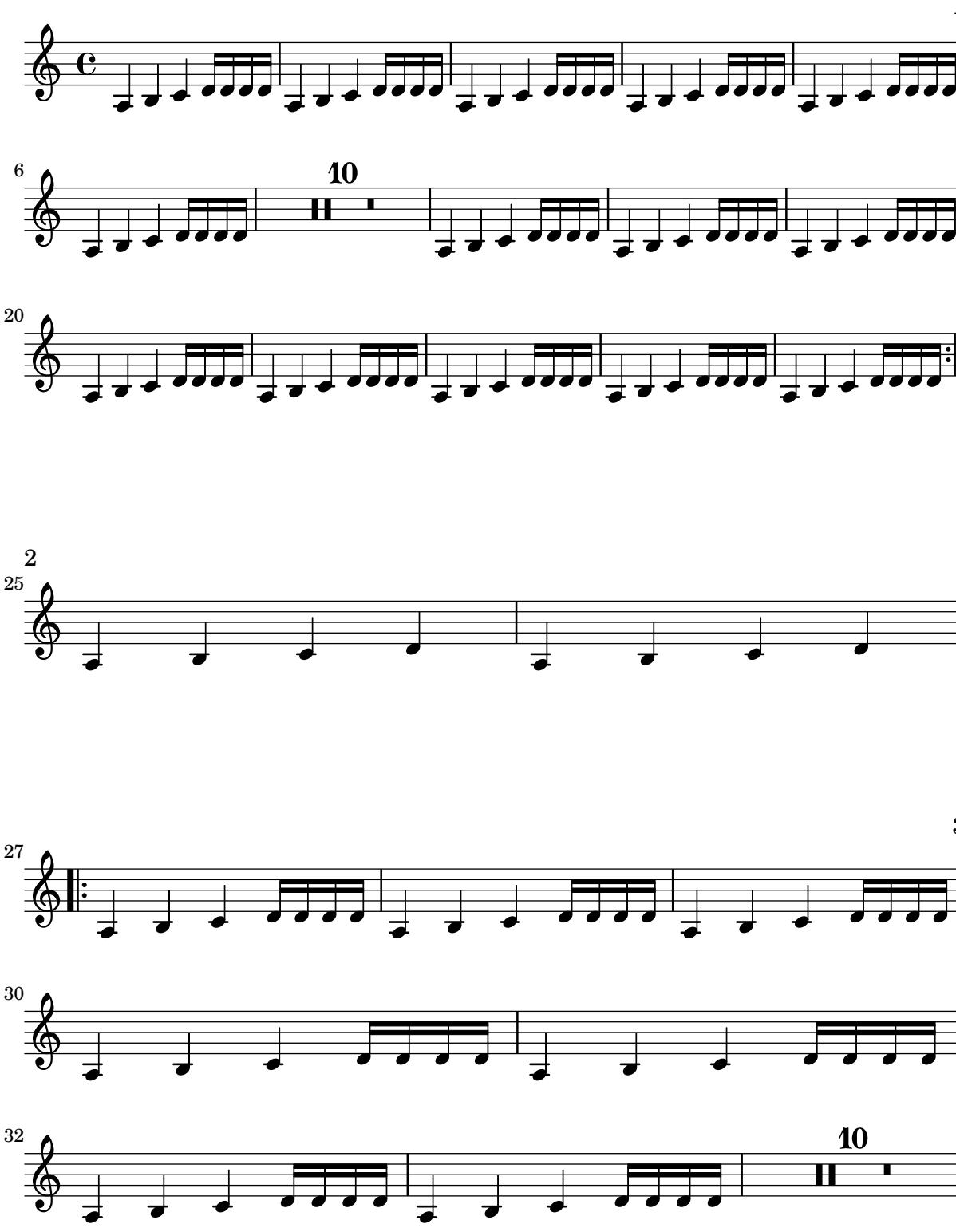
Allowing the first command column to be breakable caused a crash in Page_turn_page_breaking.
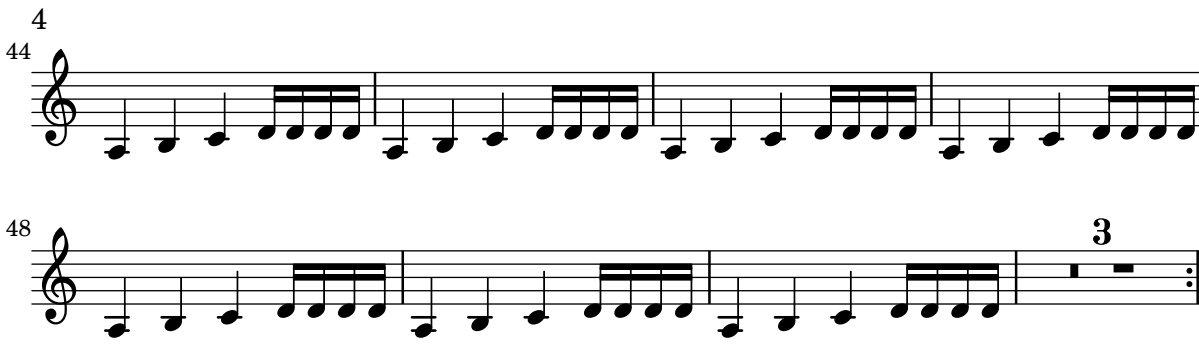
page-turn-page-breaking-first-column.ly



The page-turn engraver will not count potential page turns if they occur in the middle of a repeat unless there is a long gap at the beginning or at the end of the repeat.

`page-turn-page-breaking-repeats.ly`
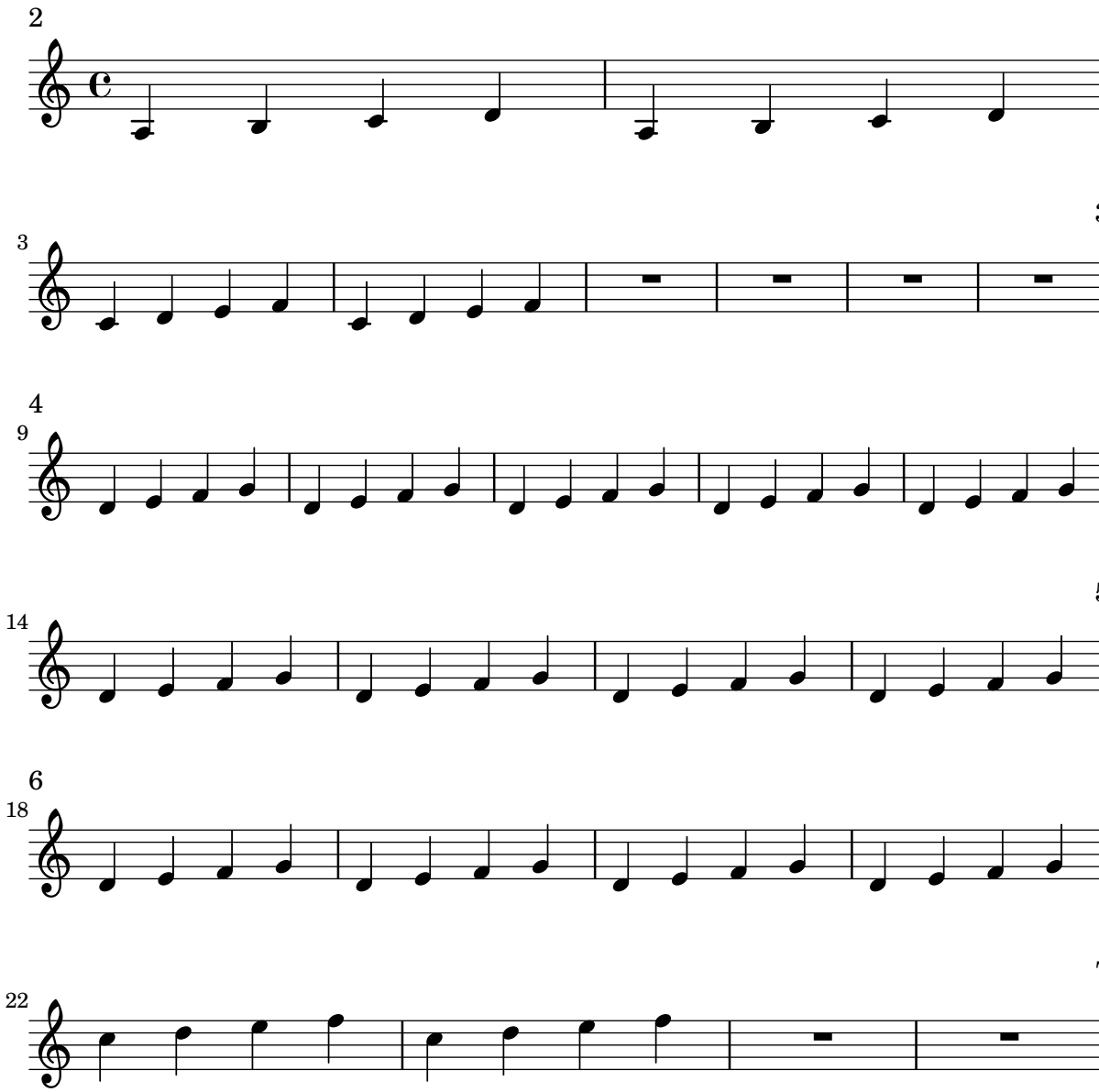


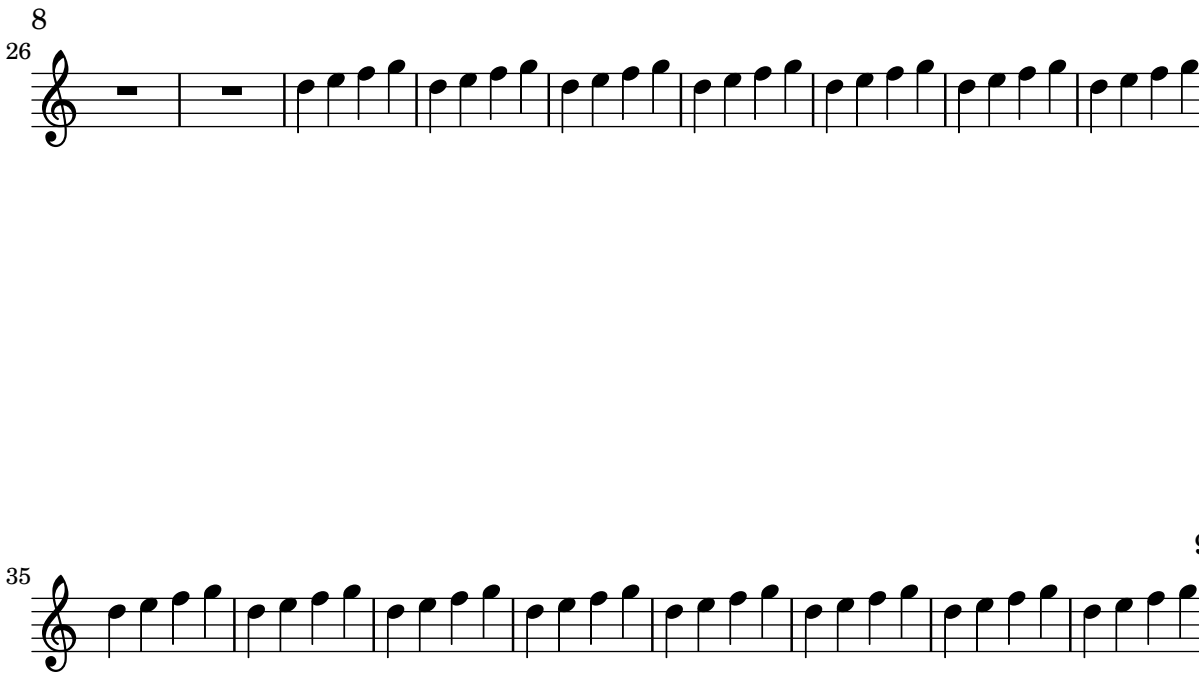


Music engraving by LilyPond 2.21.1—www.lilypond.org

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.

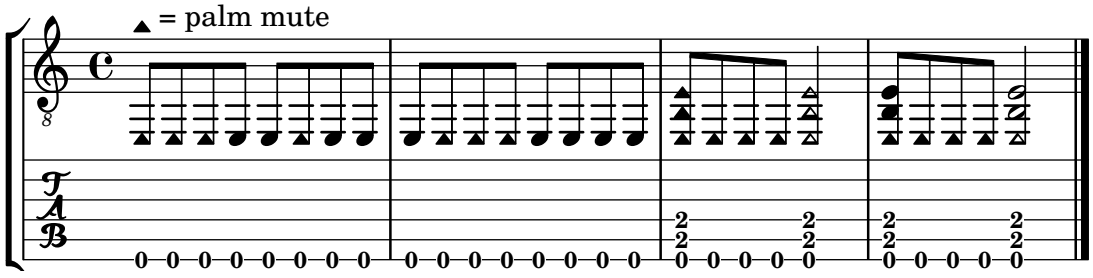
`page-turn-page-breaking.ly`





The palm mute technique for stringed instruments is supported by triangle-shaped note heads.
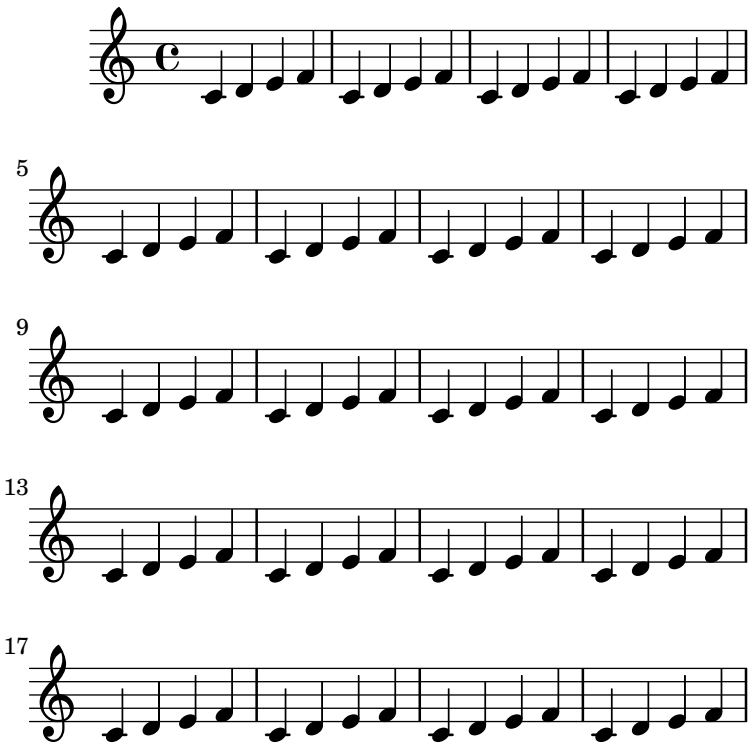
`palm-mute.ly`



Default values for margins, indents, and offsets are accessible in paper-defaults-init.ly and apply to the default paper size returned by (ly:get-option 'paper-size). For other paper sizes, they are scaled linearly.

paper-default-margins-a6.ly

For other paper sizes, margins are scaled accordingly.
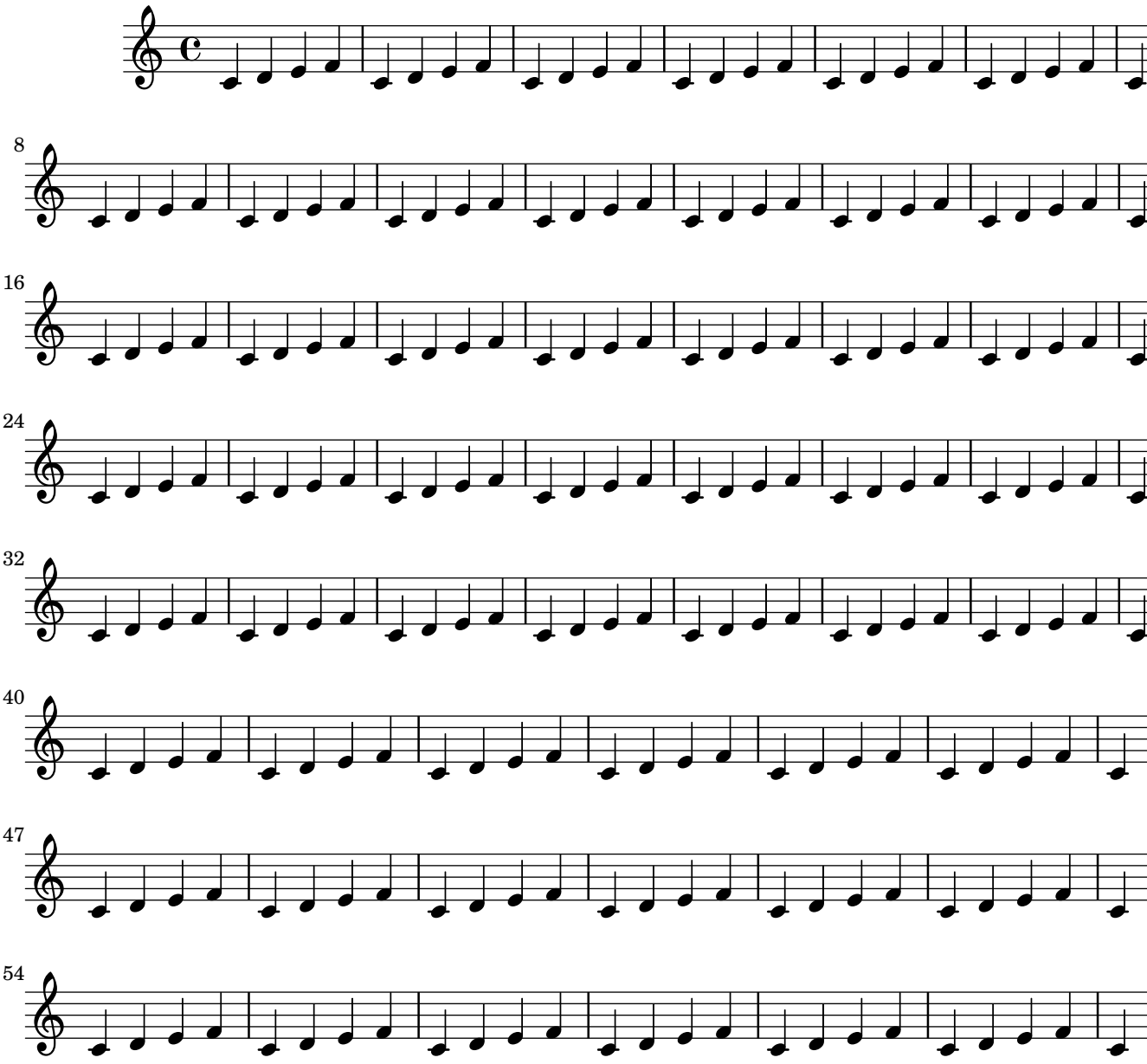


Music engraving by LilyPond 2.21.1—www.lilypond.org

Default values for margins, indents, and offsets are accessible in paper-defaults-init.ly and apply to the default paper size returned by (ly:get-option 'paper-size). For other paper sizes, they are scaled linearly.
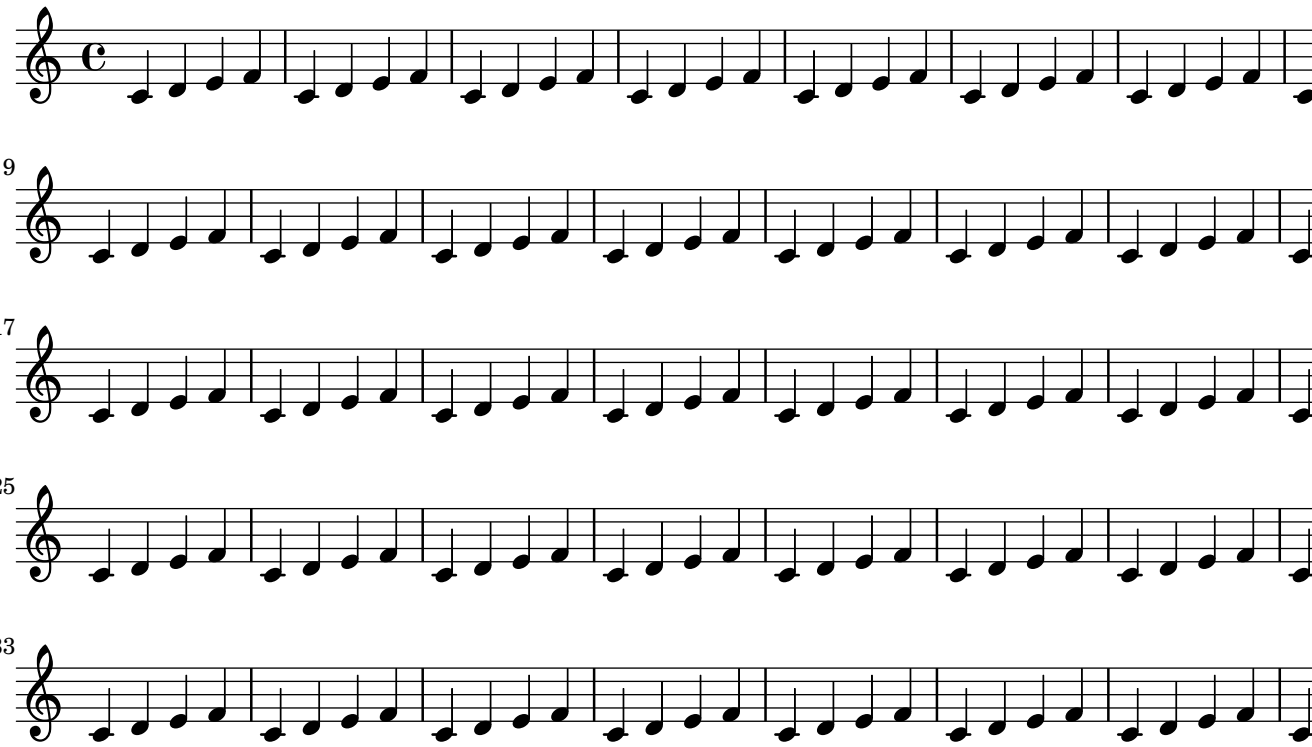
paper-default-margins-def.ly

If the paper size remains default, the margin values from paper-defaults-init.ly remain unchange
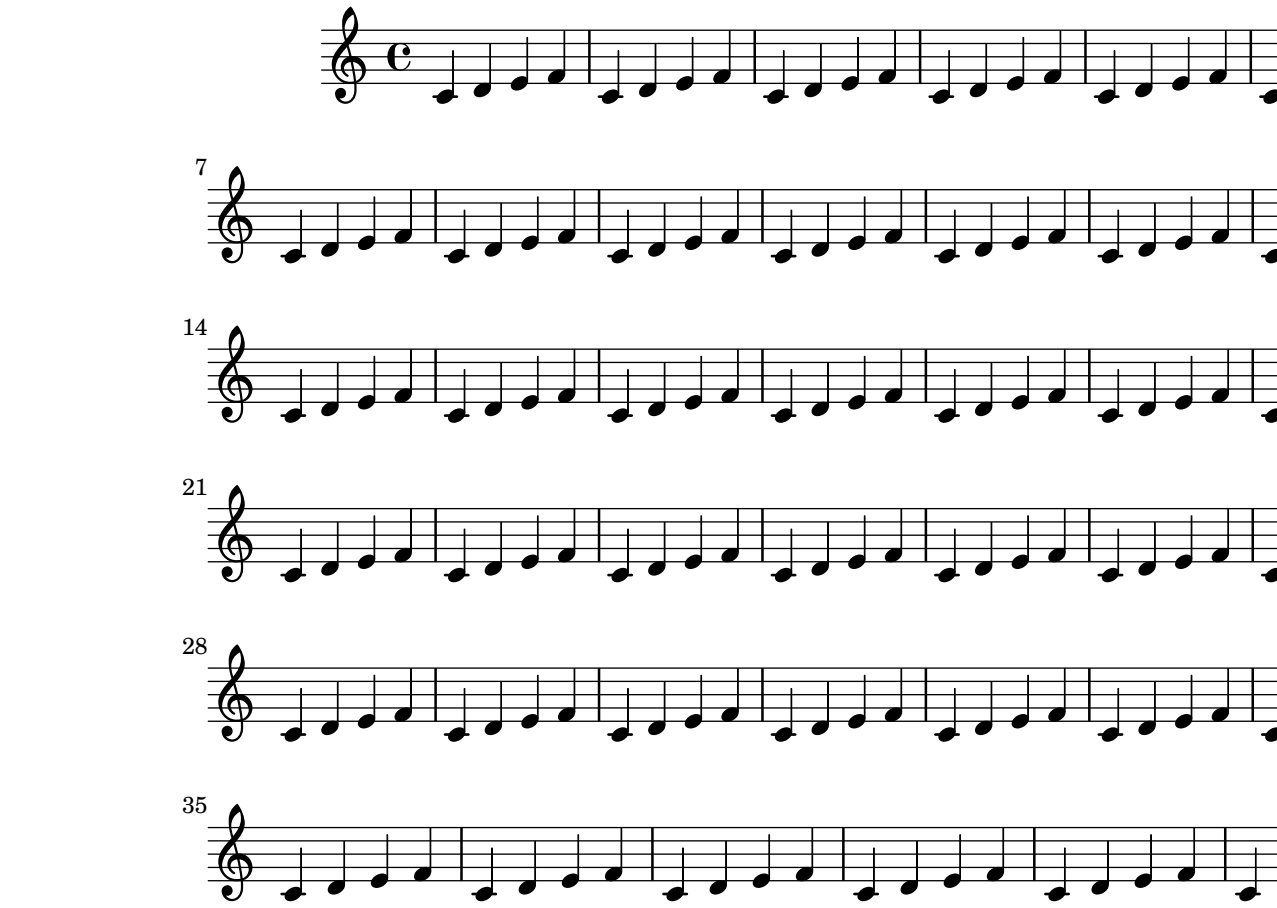


Margin values must fit the line-width, that means: paper-width = line-width + left-margin + right-margin. In case they do not, default margins are set and a warning is printed.

paper-margins-consistency.ly
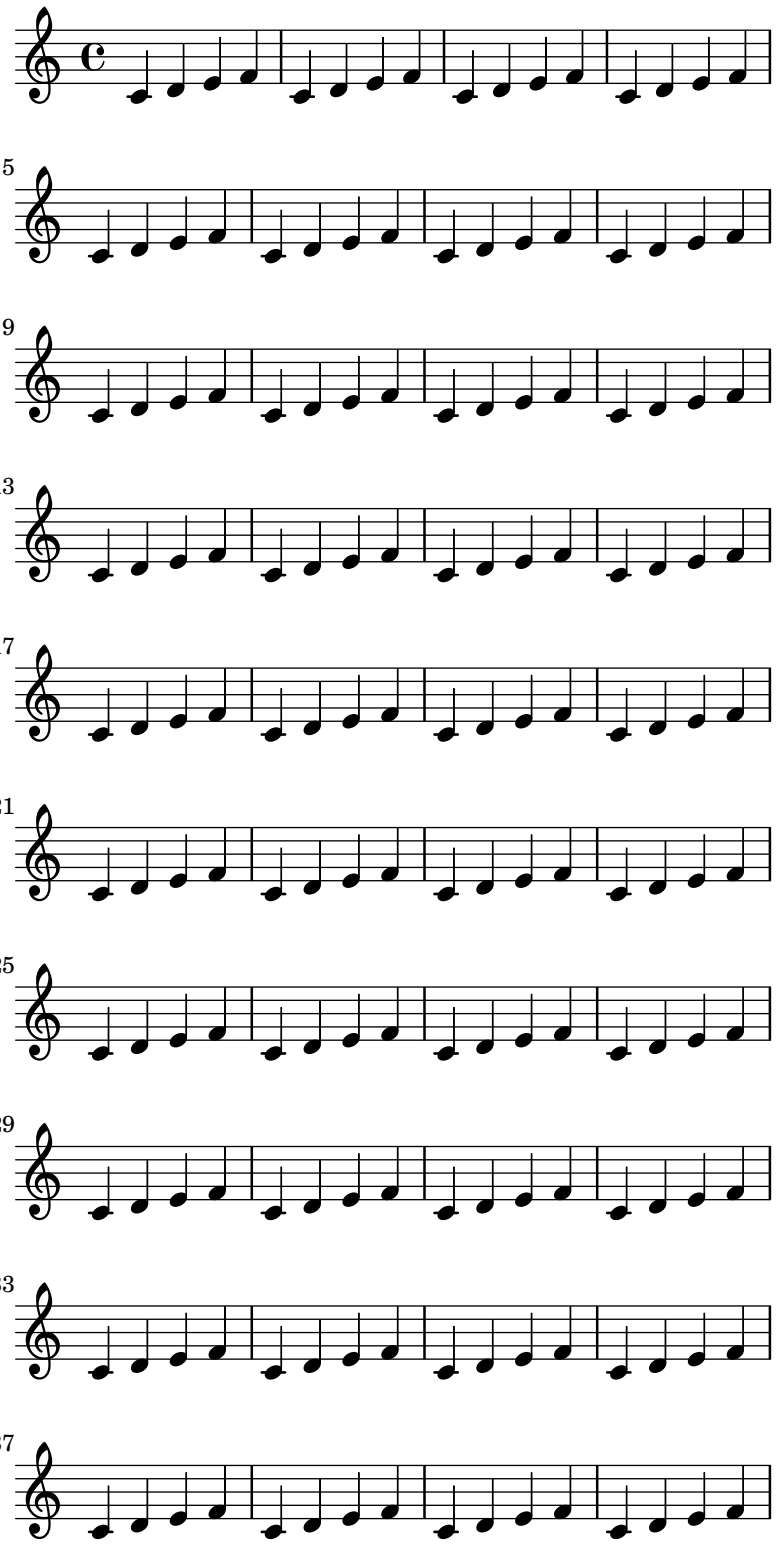


Here only left-margin is given, right-margin will remain default.

paper-margins-left-margin.ly
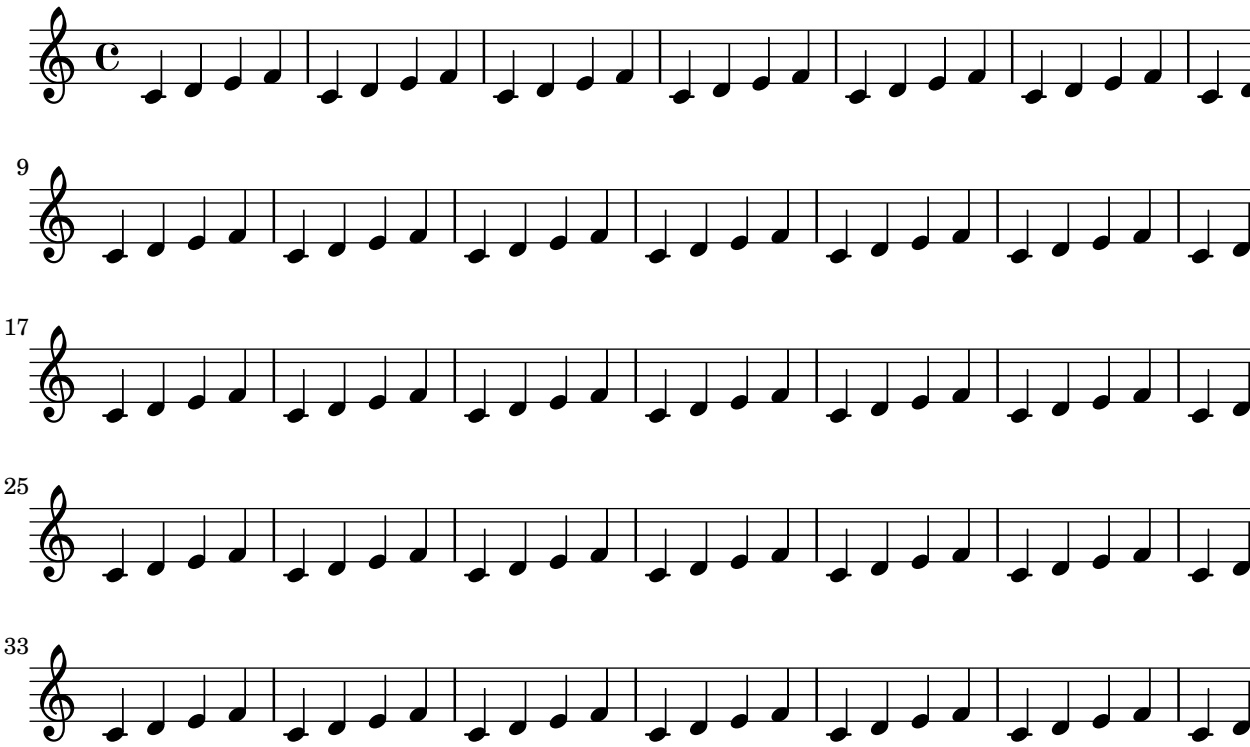


If only line-width is given, systems are horizontally centered.

paper-margins-line-width.ly
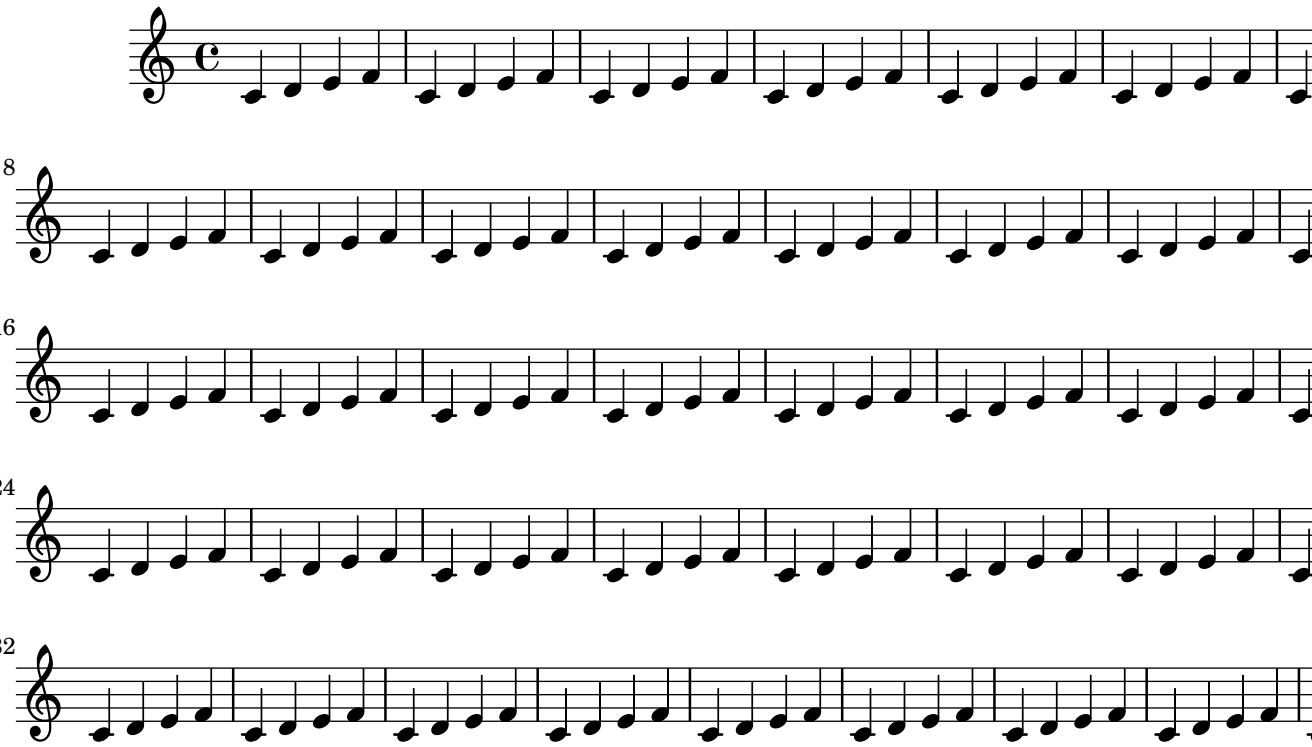


All checks can be avoided by setting check-consistency to ##f in \paper.

paper-margins-no-checks.ly
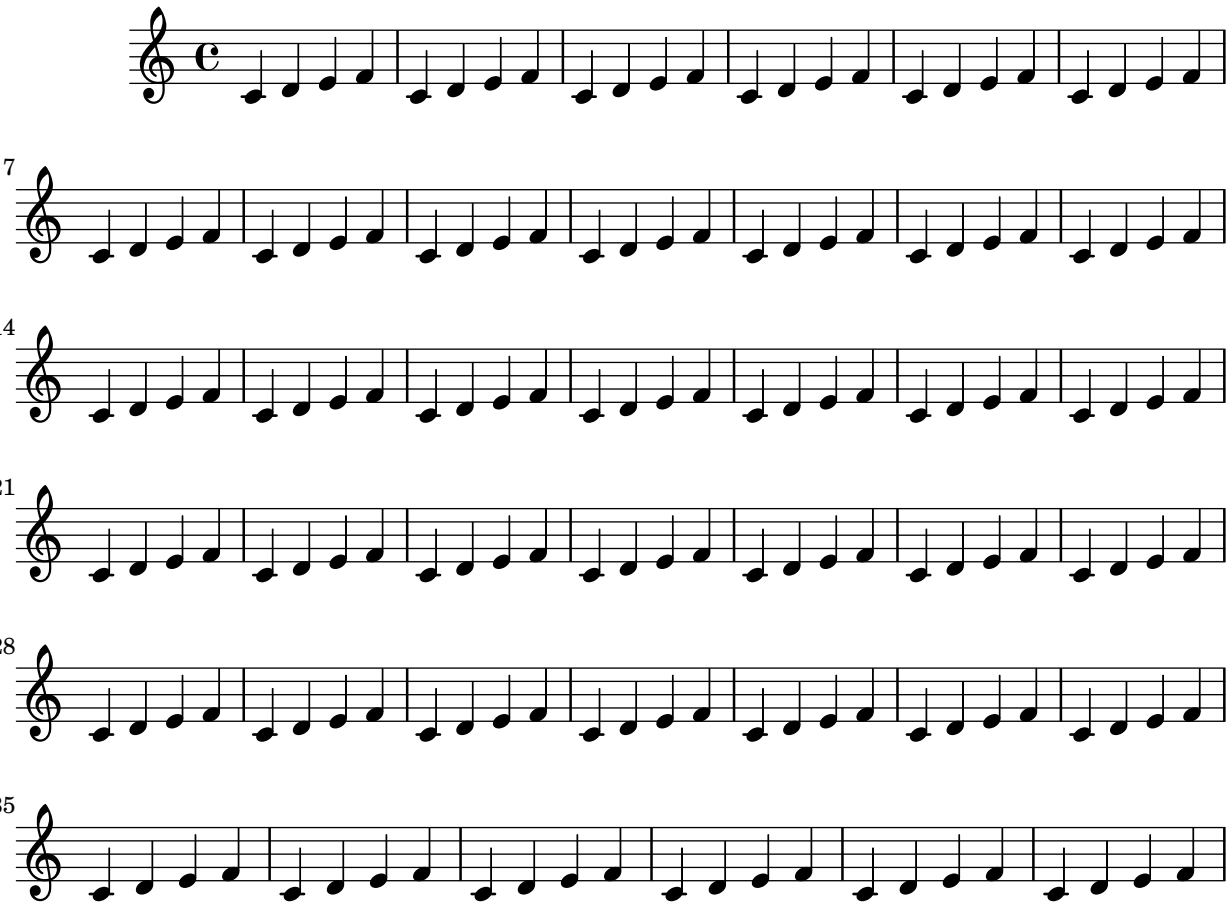


Normally, margin settings must not cause systems to run off the page.

paper-margins-overrun.ly
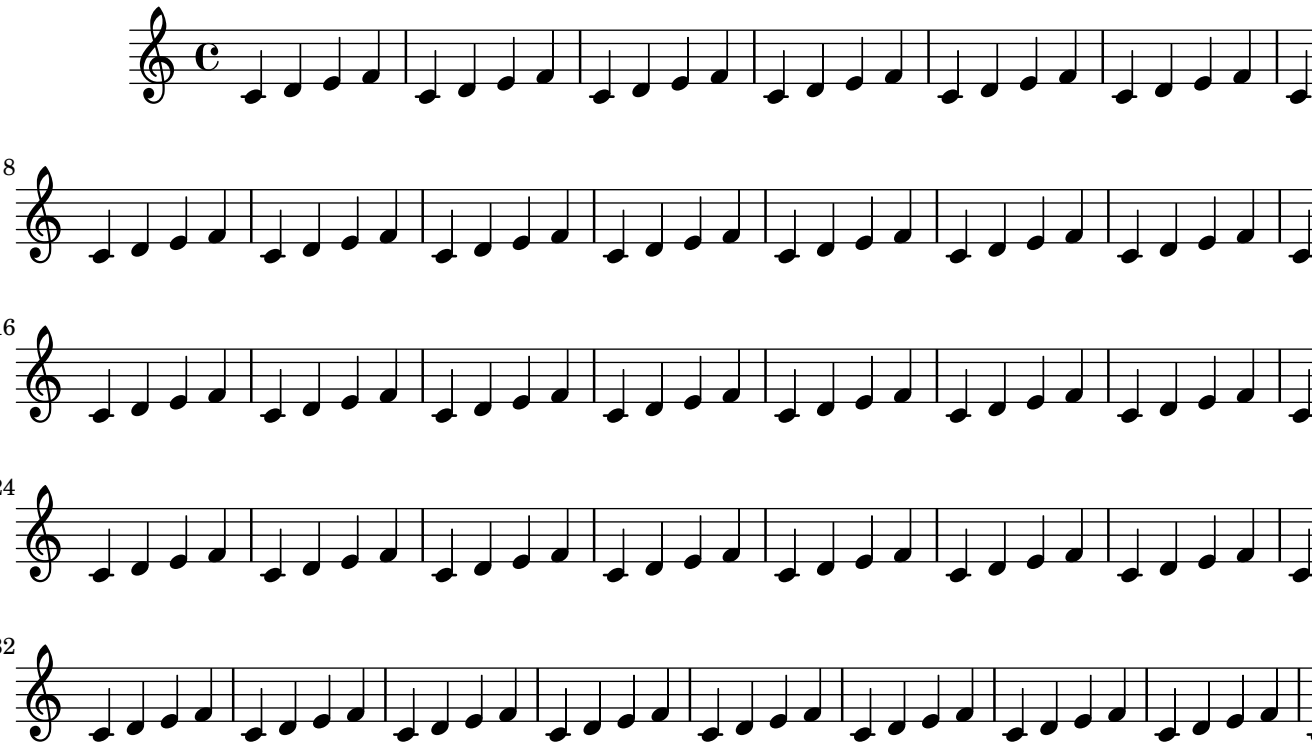


Here only right-margin is given, left-margin will remain default.

paper-margins-right-margin.ly



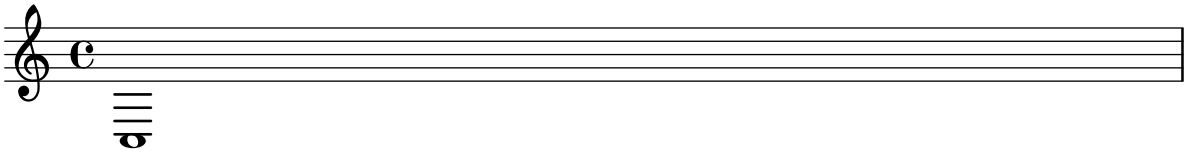
Paper margin settings do not have to be complete. Missing values are added automatically. If no paper settings are specified, default values are used.

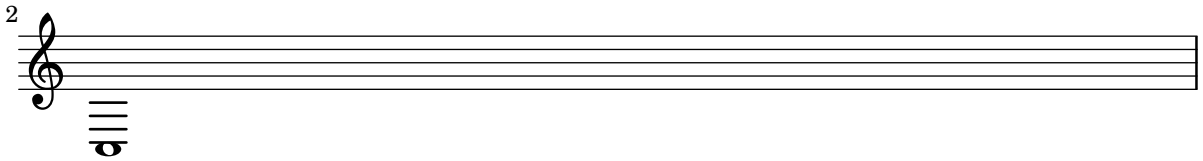
paper-margins.ly



Nested properties can be set in the paper block.

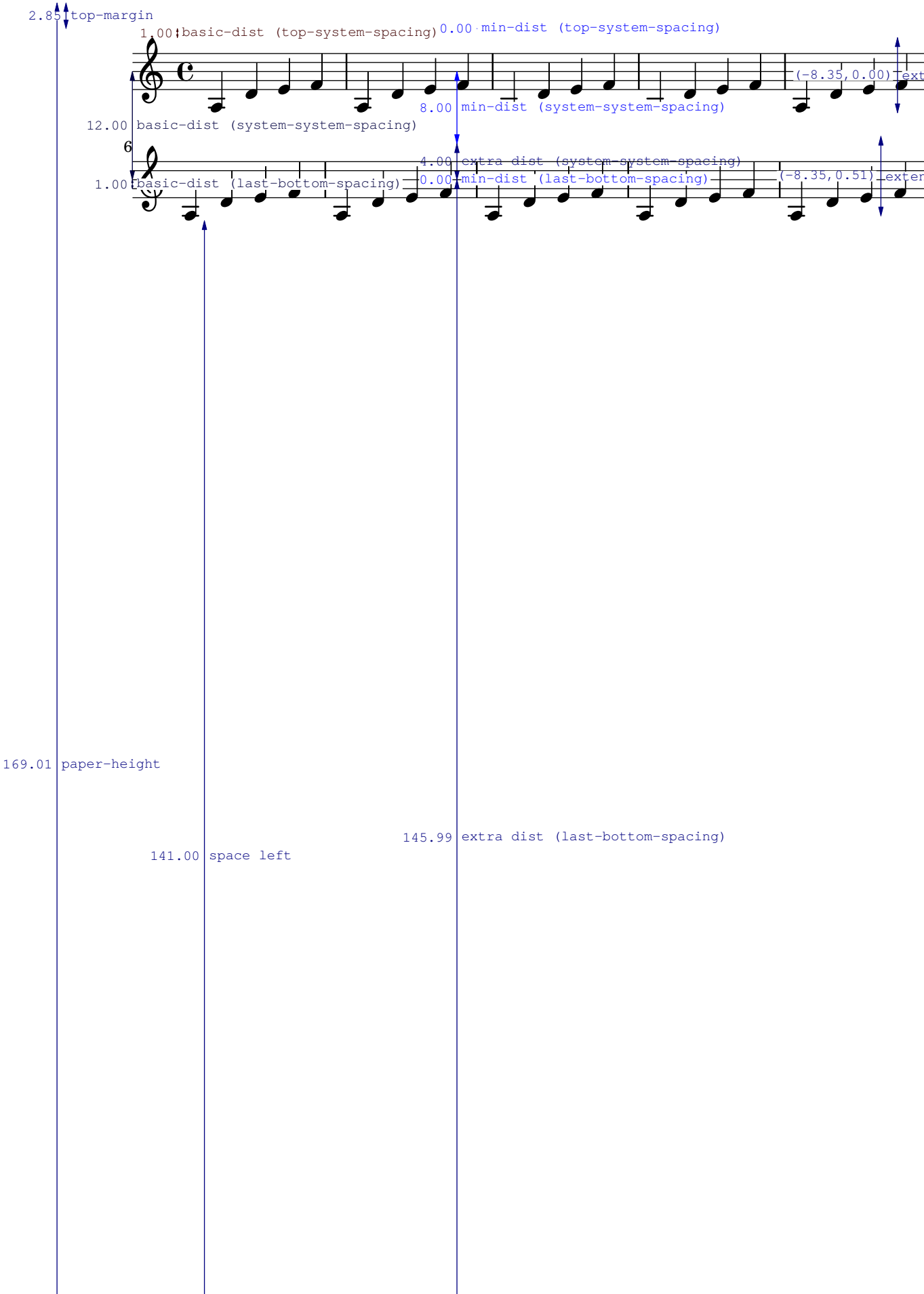
`paper-nested-override.ly`
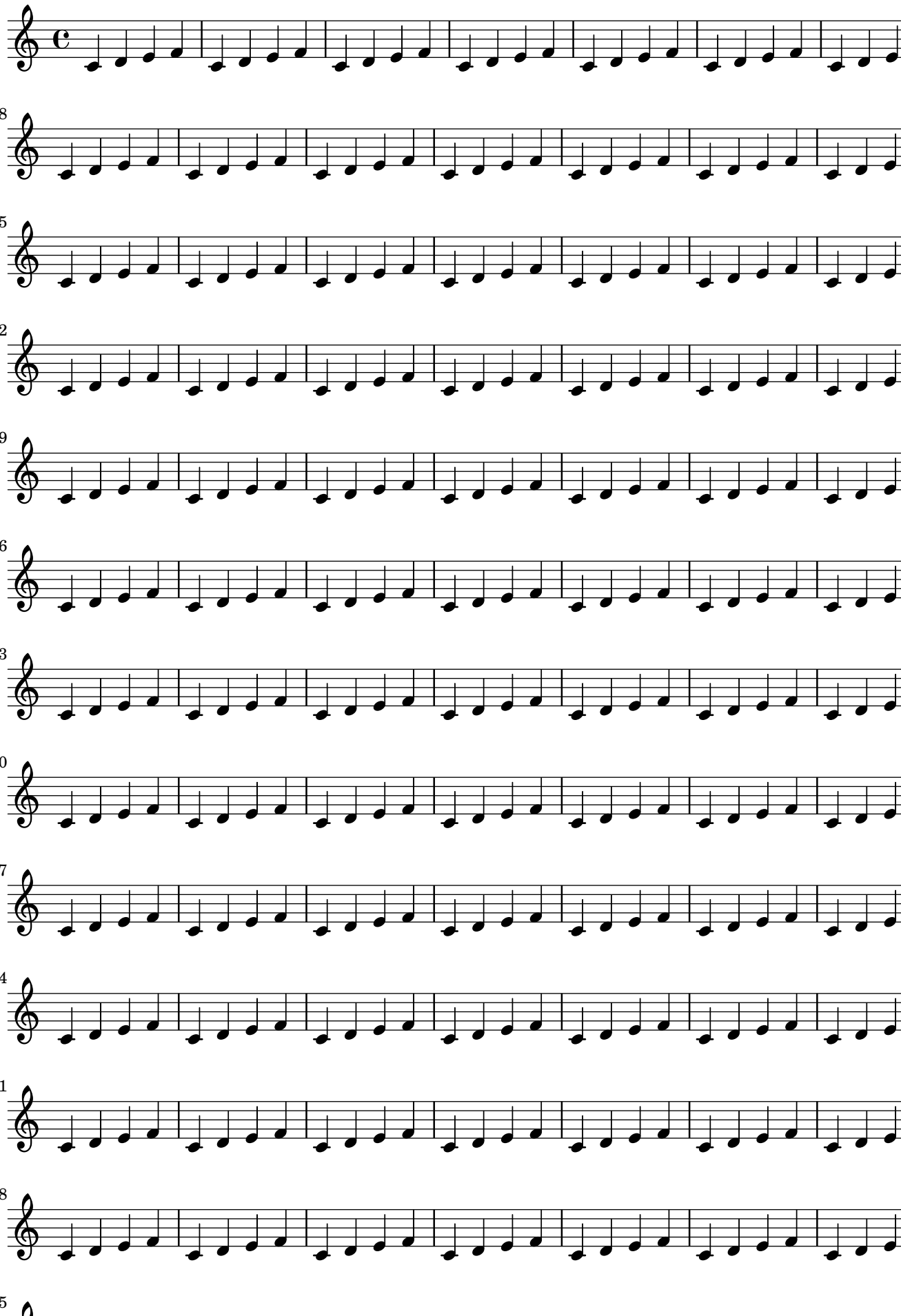




Setting individual nested paper properties does not remove existing settings or break spacing annotation.
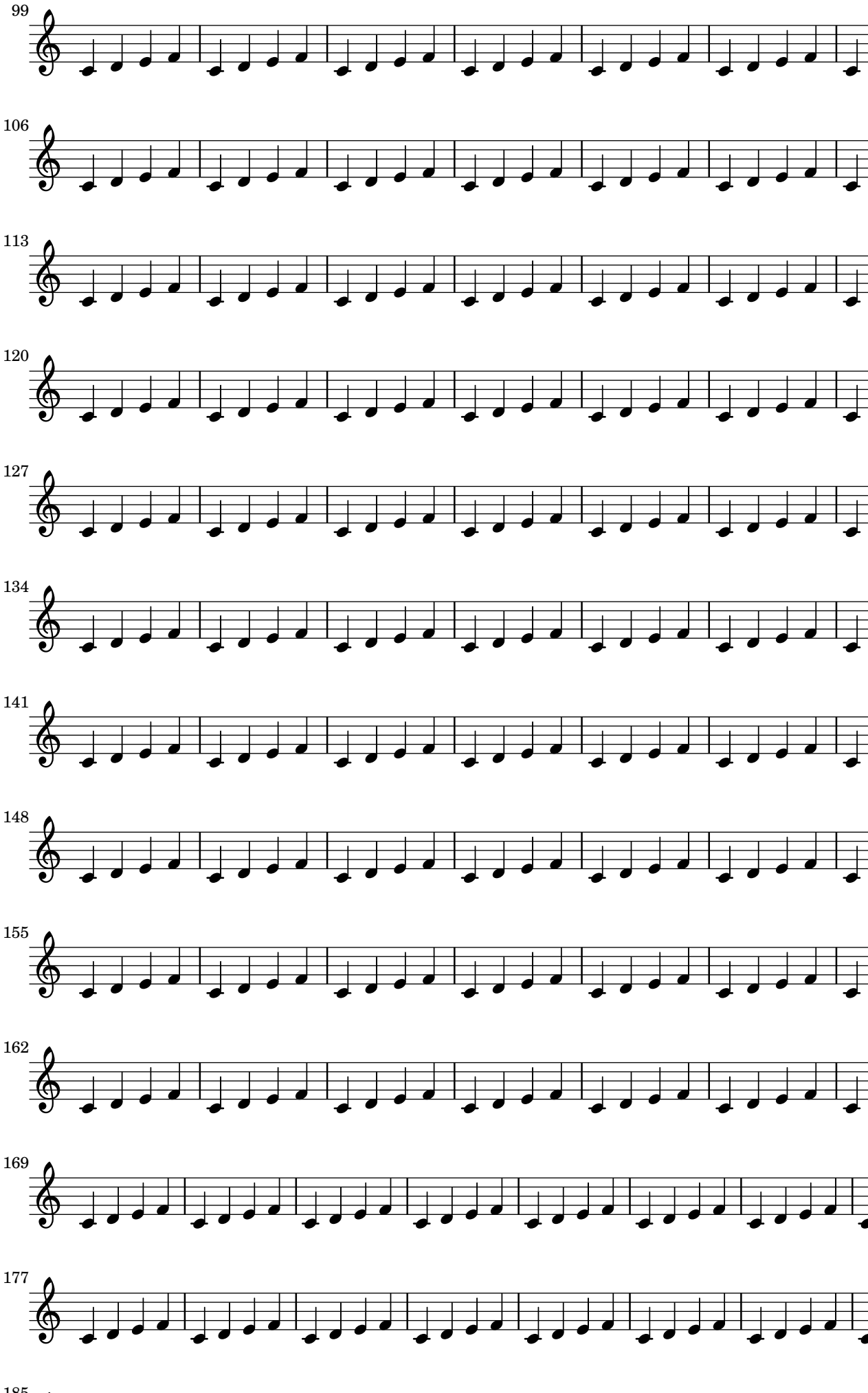
paper-nested-override2.ly
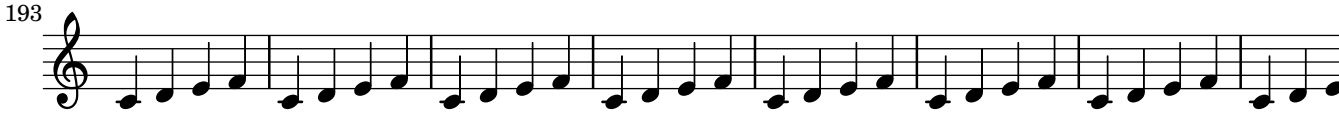


In two-sided mode, a binding offset can be specified, which is added to the inner margin automatically.

paper-twosided-bcorr.ly
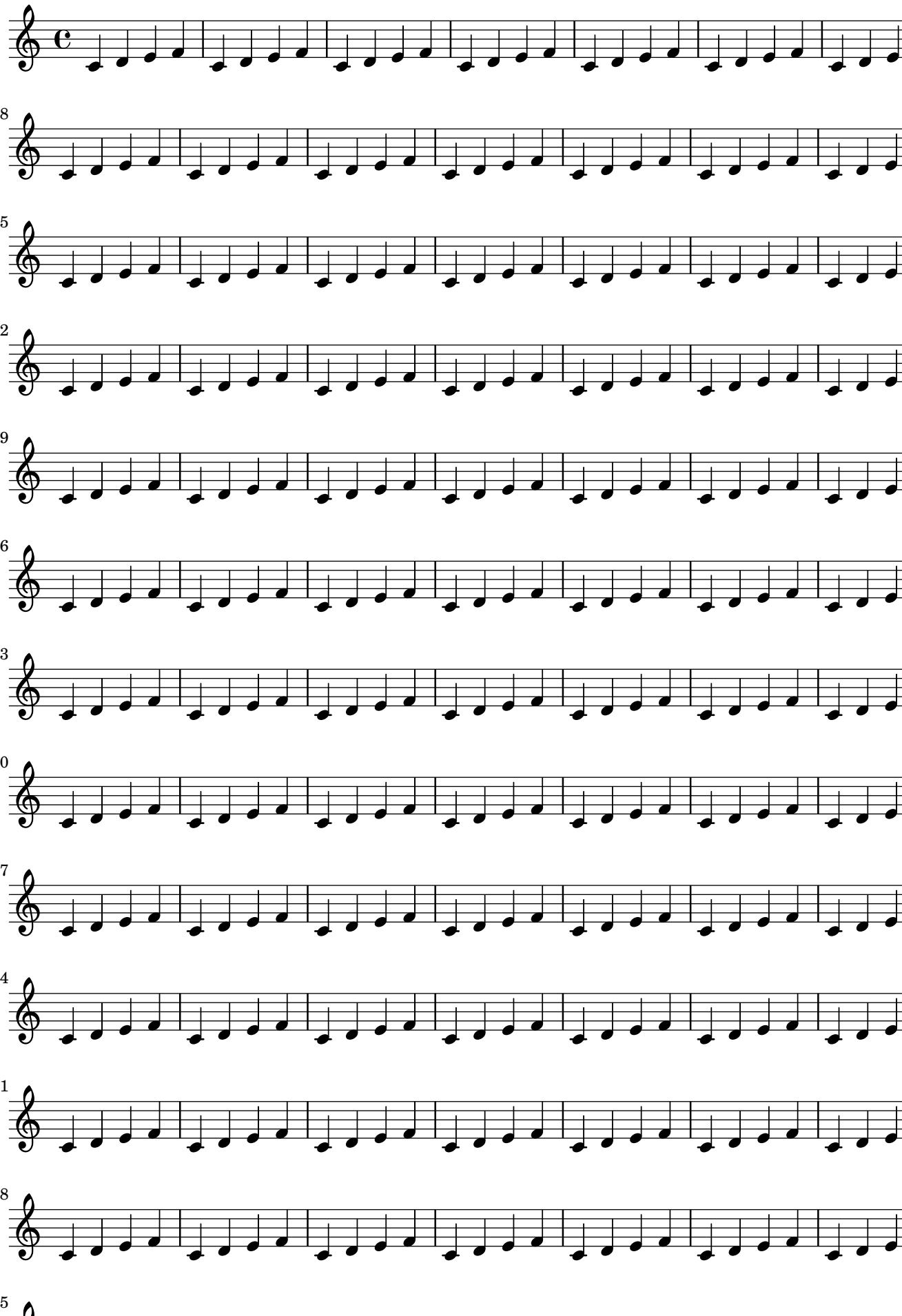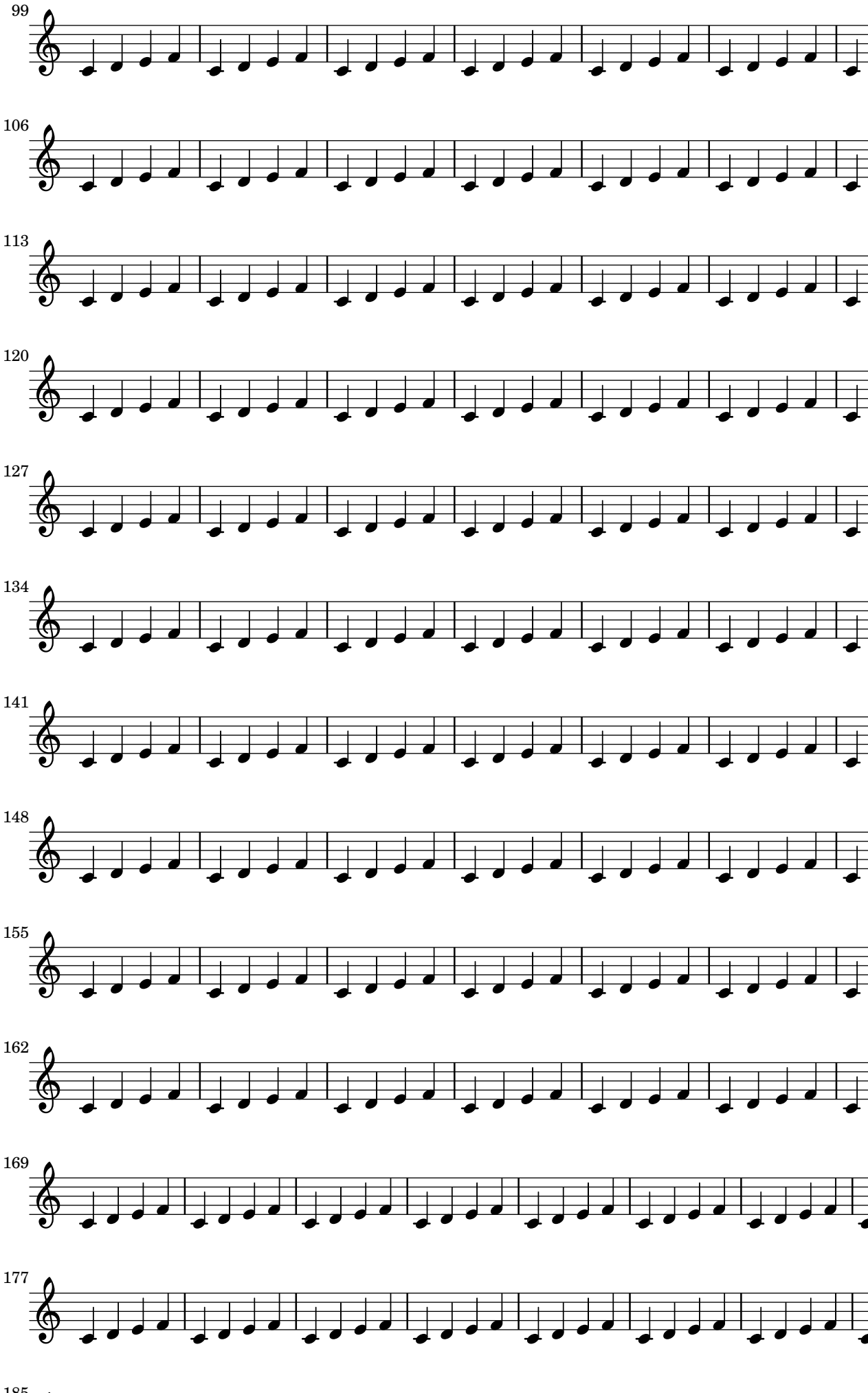



99
106
113
120
127
134
141
148
155
162
169
177
185
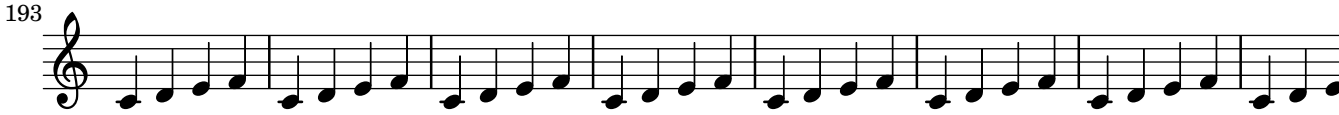

193

Two-sided mode allows you to use different margins for odd and even pages.

paper-twosided.ly
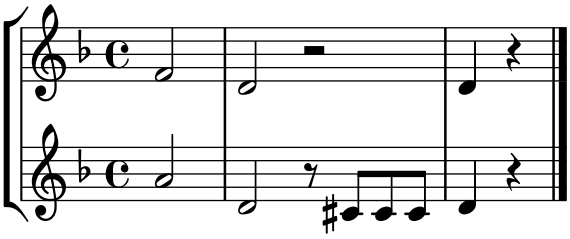



99
106
113
120
127
134
141
148
155
162
169
177
185


193

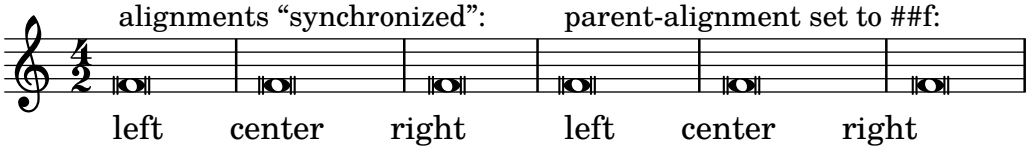
`\parallelMusic` does not complain about incomplete bars at its end.

`parallelmusic-partial.ly`
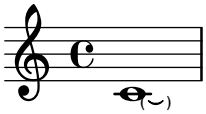


When `parent-alignment-X` property is unset, the value of `self-alignment-X` will be used as the factor for parent alignment. This happens e.g. for LyricTexts.

`parent-alignment-synchronized-with-self-alignment.ly`



`\laissezVibrer` can be parenthesized without programming errors.

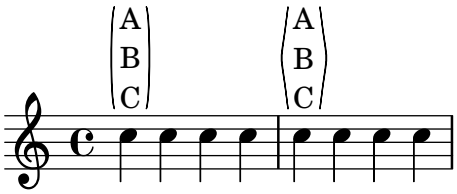
`parenthesize-laissezvibrer.ly`



The parenthesize markup will place parentheses around any stencil.
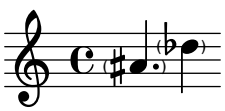
The angularity of the parentheses can be adjusted.

`parenthesize-markup.ly`



Parentheses around notes also include accidentals and dots; they are centered on the vertical center of the combined enclosed items.

`parenthesize-notes-accidentals.ly`

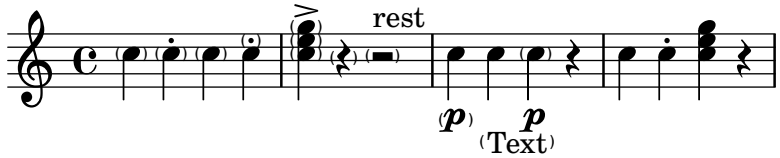


The parenthesize function should also work on single notes (not inside chords), rests and on whole chords (each note of the chord is parenthesized). Also, parenthesizing articulations, dynamics and text markup is possible. On all other music expressions, parenthesize does not have an effect.

Measure 1: Three parenthesized notes (staccato not parenthesized), one note with staccato in parentheses; Measure 2: Chord and two rests in parentheses (accent and markup not); Measure
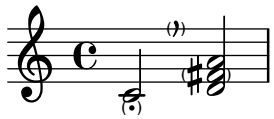
3: note (no parentheses) with \p in parentheses, with text in parentheses, and note in parentheses with p not in parentheses, rest (no parentheses); Measure 4: shows that \parenthesize does not apply to other expressions like SequentialMusic

`parenthesize-singlenotes-chords-rests.ly`
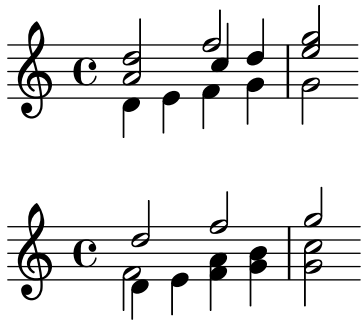


The parenthesize function is a special tweak that encloses objects in parentheses. The associated grob is `Score.ParenthesesItem`.

`parenthesize.ly`



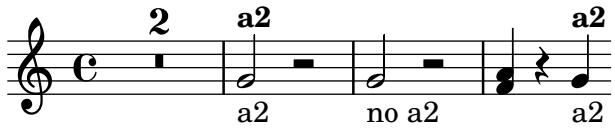
It is possible to use the part combiner for three voices with \partCombineUp and \partCombineDown.

`part-combine-3voices.ly`



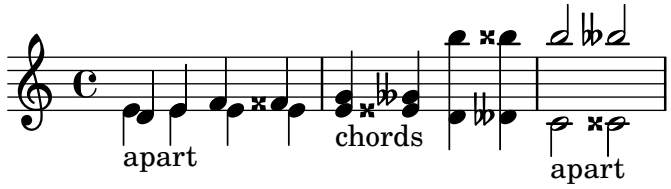
The a2 string is printed only on notes (i.e. not on rests), and only after chords, solo or polyphony.

`part-combine-a2.ly`
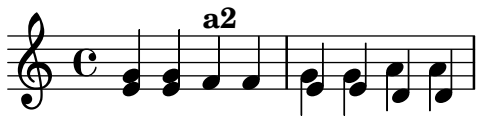


The part combiner has an option to set the range of differences in steps between parts that may be combined into chords.

`part-combine-chord-range.ly`
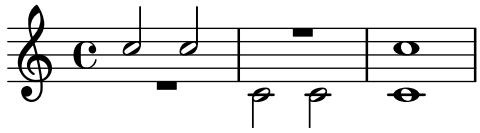


The part combiner stays apart for crossing voices.

part-combine-cross.ly



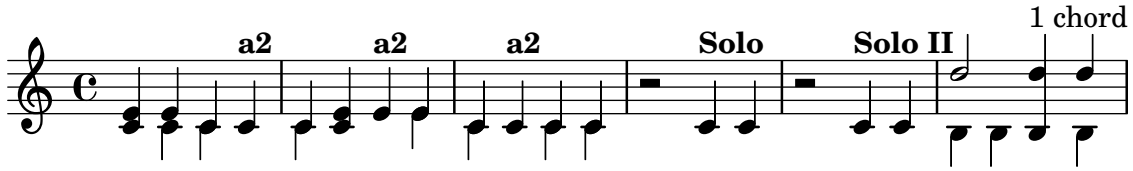
If the part-combiner shows two separate voices, multi-measure rests are supposed to use the same settings as `\voiceOnce` and `\voiceTwo`.

part-combine-force-mmrest-position.ly



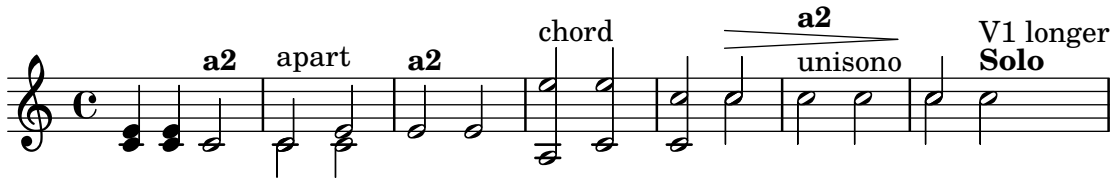
Overrides for the part-combiner, affecting only one moment. The `partCombine...Once` override applies only to one moment, after which the old override – if any – is in effect again.

part-combine-force-once.ly



Overrides for the part-combiner. All functions like `\partCombineApart` and `\once \partCombineApart` are internally implemented using a dedicated `partCombineForced` context property.

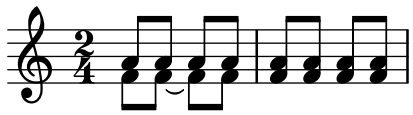
part-combine-force.ly



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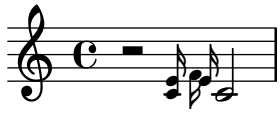
In the 2nd measure, the pattern without the tie, leads to combined voices.

part-combine-global.ly
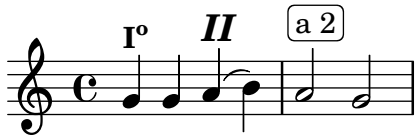


The notes of the first chord share a stem but the notes of the second chord do not.

part-combine-inside-grace.ly



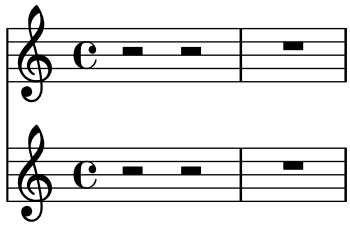
Part combine texts accept markup.

part-combine-markup.ly
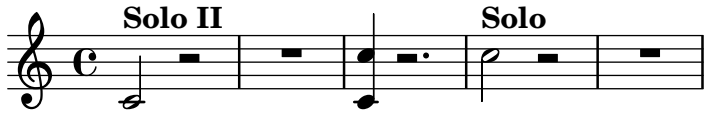


Normal rests are preferred over multi-measure rests. A multi-measure rest beginning in one part in the middle of a multi-measure rest in the other part appears as expected.

part-combine-mmrest-after-apart-silence.ly



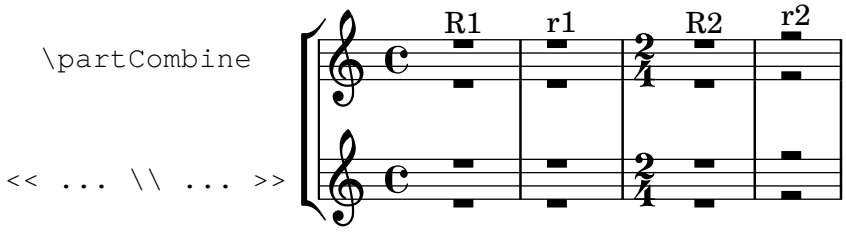
Multimeasure rests are printed after solos, both for solo1 and for solo2.

part-combine-mmrest-after-solo.ly



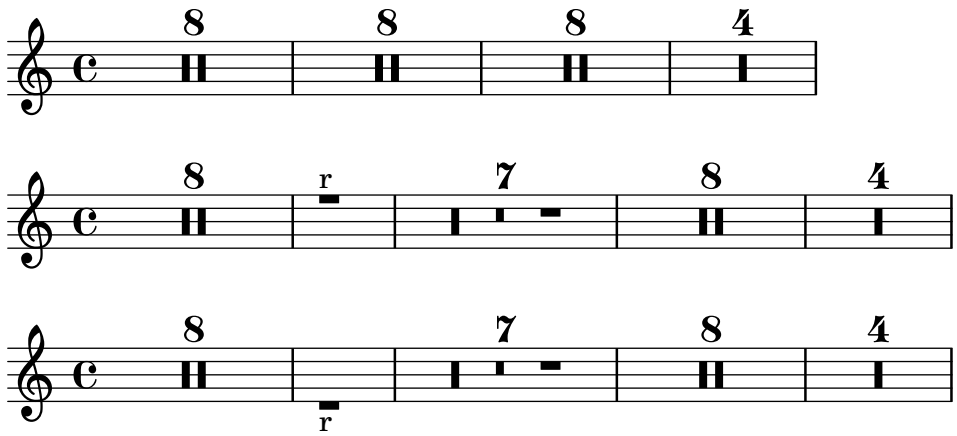
The positioning of multimeasure rests in `\partCombineApart` passages corresponds with `\voiceOne` and `\voiceTwo` even when using non-standard staves.

part-combine-mmrest-apart.ly



Multi-measure rests do not have to begin and end simultaneously to be combined.
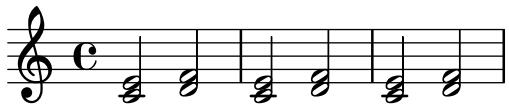
part-combine-mmrest-shared.ly



`\partCombine` needs to be given pitches in their final octaves, so if `\relative` is used it must be applied inside `\partCombine`. The pitches in `\partCombine` are unaffected by an outer `\relative`, so that the printed output shows the pitches that `\partCombine` used.

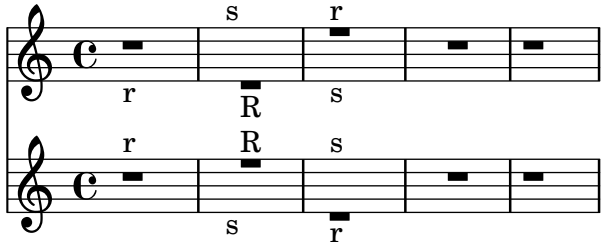
The expected output of this test is three identical measures.

`part-combine-relative.ly`
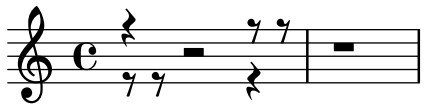


Different kinds of silence are not merged into the shared voice even if they begin and end simultaneously; however, when rests and skips are present in the same part, the skips are ignored.
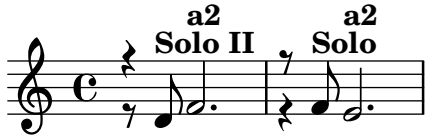
`part-combine-silence-mixed.ly`
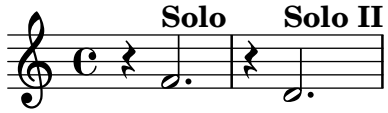


Rests must begin and end simultaneously to be merged into the shared voice.

`part-combine-silence.ly`
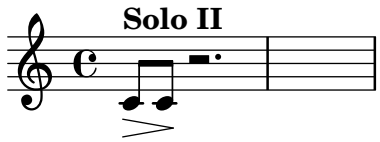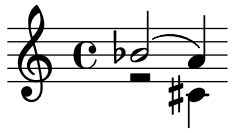






SOLO is printed even if the solo voice ends before the other one. Unfortunately, the multi-rest of the 1st voice (which is 2 bars longer than the 2nd voice) does not get printed.

`part-combine-solo-end.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-global.ly
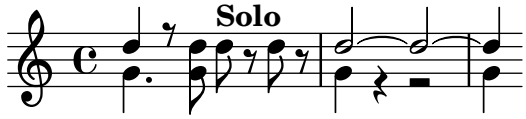


A solo string can only be printed when a note starts. Hence, in this example, there is no Solo-2 although the 2nd voice has a dotted quarter, while the first voice has a rest.

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

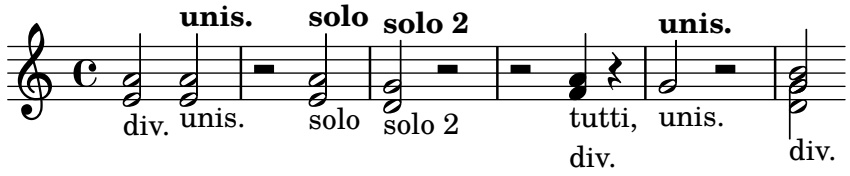
Solo 1/2 can not be used when a spanner is active, so there is no solo over any of the tied notes.

part-combine-solo.ly
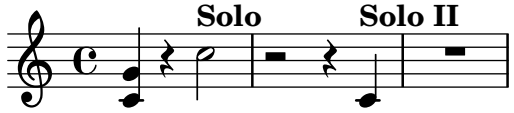


Test some transitions that might be found in string parts produced with \partCombine.

part-combine-strings.ly
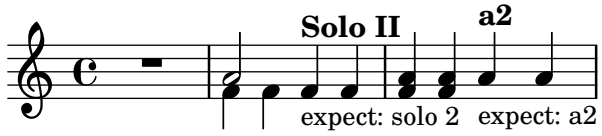


Wait for the next real note for part-combine texts (i.e. don't print part-combine texts on rests). This is needed because the part-combiner needs an override if one voice has a full-bar rest while the other has some rests and then a solo.

part-combine-text-wait.ly
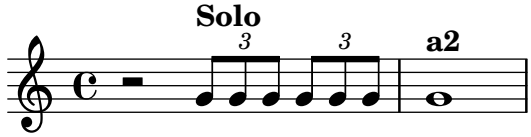


The part combiner detects a2, solo1 and solo2, and prints texts accordingly.

part-combine-text.ly



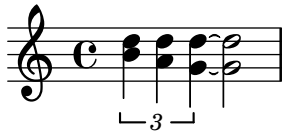
End tuplets events are sent to the starting context, so even after a switch, a tuplet ends correctly.

part-combine-tuplet-end.ly
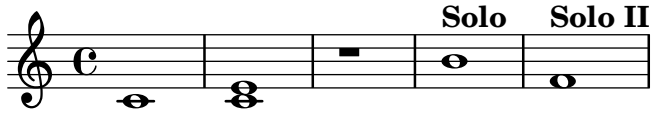


Tuplets in combined parts only print one bracket.

`part-combine-tuplet-single.ly`
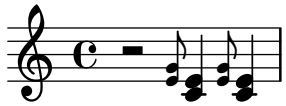


The part combiner can combine parts of unequal lengths.

`part-combine-unequal-lengths.ly`



Grace notes in parts are combined.
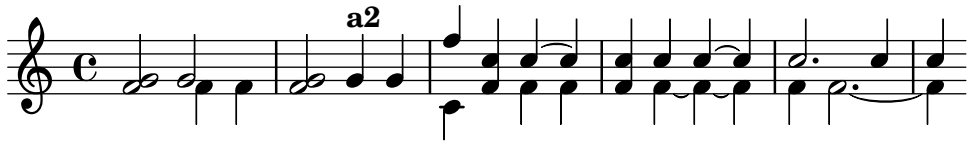
`part-combine-with-grace.ly`



The new part combiner stays apart from:
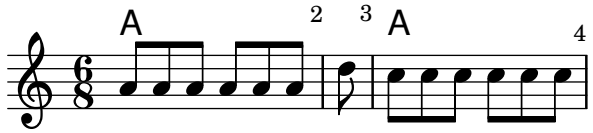
- different durations,
- different articulations (taking into account only slur/beam/tie), and
- wide pitch ranges.

`part-combine.ly`



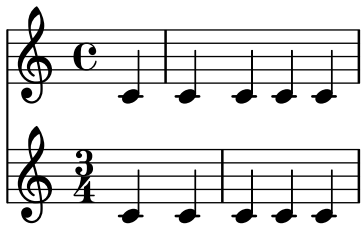
`\partial` can be can be called in mid-piece in multiple contexts.

`partial-in-mid-piece.ly`



`\partial` works with polymetric staves.

`partial-polymetric.ly`



`\pattern` and `\fill-with-pattern` markup commands should interpret their arguments only once. This test calls them with a markup command that counts how often it is evaluated. The first line is supposed to show just ‘1’ multiple times, the second line uses numbers ‘2’ to ‘4’.

`pattern-markup-evaluation.ly`

1 1 1 1 1 1 1 1 1 1

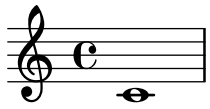
3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4

PDF metadata need either Latin1 encoding (not UTF8) or full UTF-16BE with BOM. The title field uses full UTF-16 (russian characters, euro, etc), while the composer uses normal european diacrits (which need to be encoded as Latin1, not as UTF8). Closing parenthesis need to be escaped by a backslash AFTER encoding!

`pdfmark-metadata-unicode.ly`

UTF-16BE title:² € ĂĄœĘŮůſЖюль)\ ¡

UTF-16BE with parentheses:) € ĂĄœĘŮůſЖюль) composer (with special chars): Jöhånñ Strauß



The PDF backend uses several header fields to store metadata in the resulting PDF file. Header fields with the prefix pdf override those without the prefix for PDF creation (not for visual display on the page).

`pdfmark-metadata.ly`

Title of the piece

Subtitle of the piece

The Genius Composer
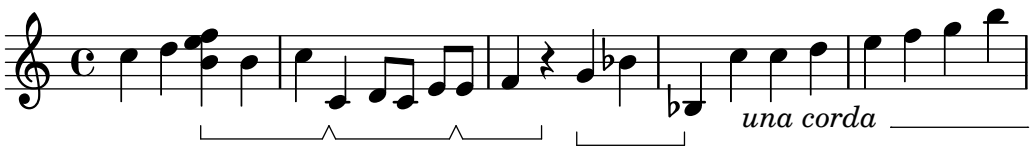
The Arranger ⓕ
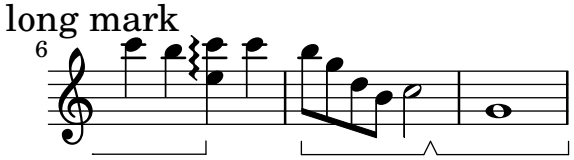


The brackets of a piano pedal should start and end at the left side of the main note-column. If a note is shared between two brackets, these ends are flared.

At a line-break, there are no vertical endings. Pedal changes can be placed at spacer rests.

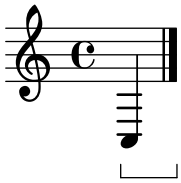
`pedal-bracket.ly`
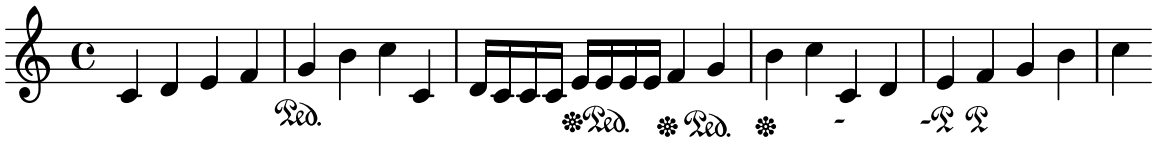




Unterminated piano pedal brackets run to the end of the piece.

`pedal-end.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.

`pedal-ped.ly`
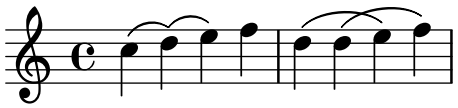


The appearance of phrasing slurs may be changed from solid to dotted or dashed.

`phrasing-slur-dash.ly`
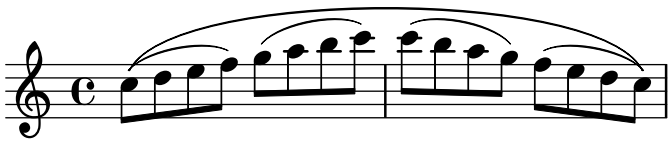


LilyPond does not support multiple concurrent phrasing slurs with the parentheses syntax. In this case, warnings will be given and the nested slur will not be generated. However, one can can create a second slur with a different spanner-id.

`phrasing-slur-multiple.ly`



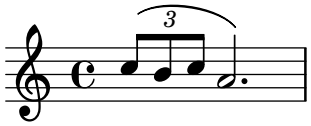
`PhrasingSlur`s go over normal slurs.

`phrasing-slur-slur-avoid.ly`



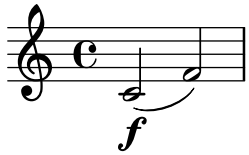
Phrasing slurs do not collide with tuplet numbers.

`phrasing-slur-tuplet.ly`
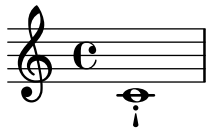


Point-and-click information can be generated only for certain event types.

`point-and-click-types.ly`



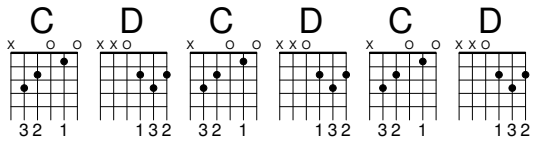
Multiple post events can be attached from Scheme expressions.

`post-events-from-scheme.ly`



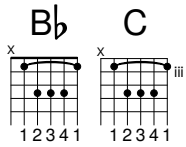
Transposition by less than one octave up or down should not affect predefined fretboards.

`predefined-fretboards-transpose.ly`



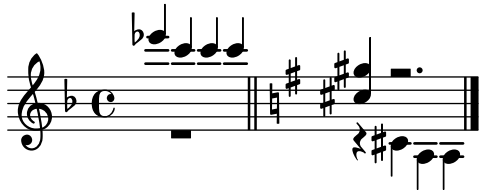
Predefined fretboards and chord shapes can be added.

`predefined-fretboards.ly`
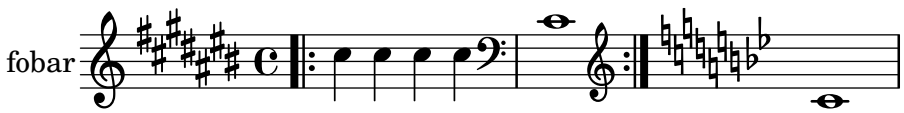


Prefatory items maintain sufficient separation from musical notation for readability, even in tight spacing. The notes should remain generally on the correct side of the time signature, key signature and barlines. A key change to G major should be legible.

`prefatory-separation.ly`
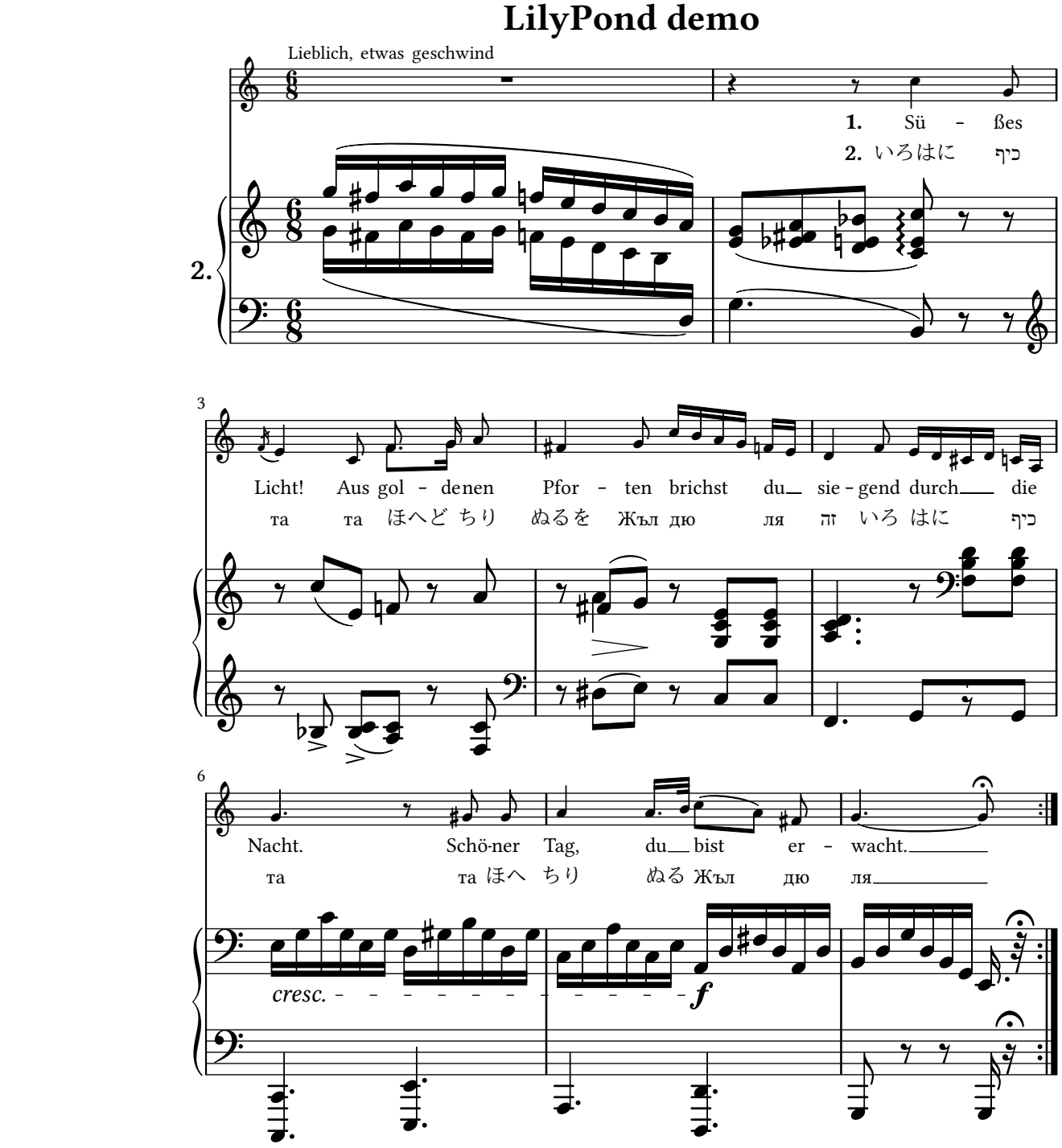


Distances between prefatory items (e.g. clef, bar, etc.) are determined by engraving standards. These distances depend on which items are combined. Mid-line, the order for clef and bar-line is different from the start of line.

`prefatory-spacing-matter.ly`
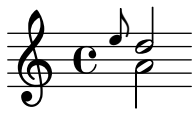


heavily mutilated Edition Peters Morgenlied by Schubert

profile-property-access.ly



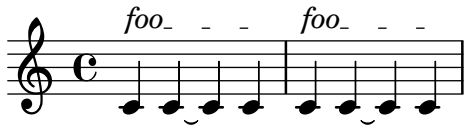
Property overrides and reverts from `\grace` do not interfere with the overrides and reverts from polyphony.

`property-grace-polyphony.ly`
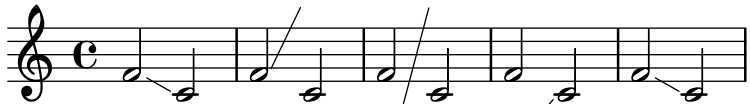


Nested properties may be overridden using Scheme list syntax. This test performs two property overrides: the first measure uses standard `\override` syntax; the second uses a list.

`property-nested-override.ly`
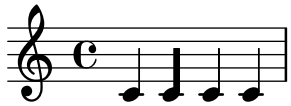


nested properties may also be reverted. This uses Scheme list syntax.

`property-nested-revert.ly`
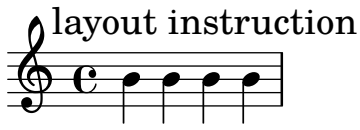


Once properties take effect during a single time step only.
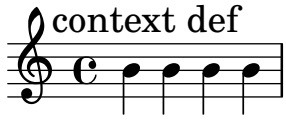
`property-once.ly`
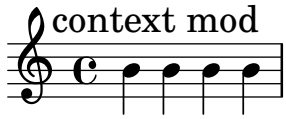


`\unset` should be able to unset the '`DrumStaff`'-specific '`clefGlyph`' equally well as layout instruction, in a context definition, or as context modification. All systems here should revert to the '`Score`'-level violin clef.

`property-unset.ly`
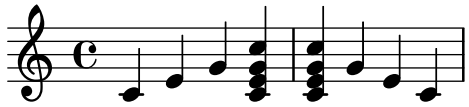






Adding material to a tag in sequential and simultaneous expressions using `\pushToTag` and `\appendToTag`. One should get the equivalent of

```
{ c' e' g' <<c' e' g' c''>> <<c'' g' e' c'>> g' e' c' }
```
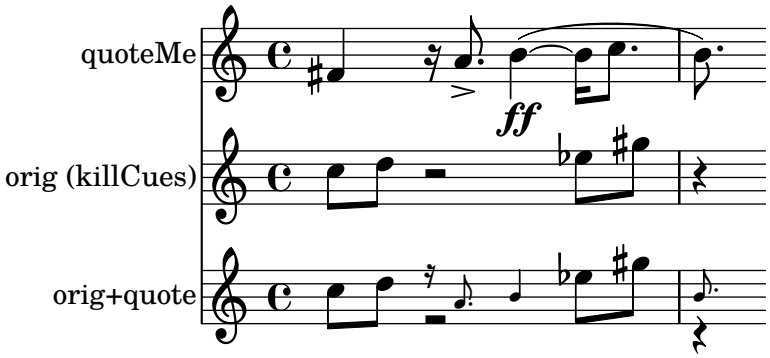
`push-to-tag.ly`



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.

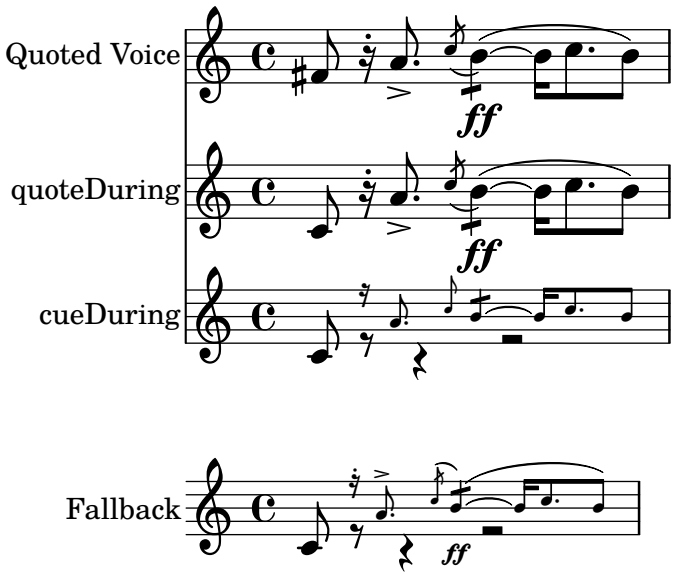
`quote-cue-during.ly`



The `cueDuring` and `quoteDuring` forms of quotation use the variables `quotedCueEventTypes` and `quotedEventTypes` to determine which events are quoted. This allows different events to be quoted for cue notes in comparison to normal quotes.

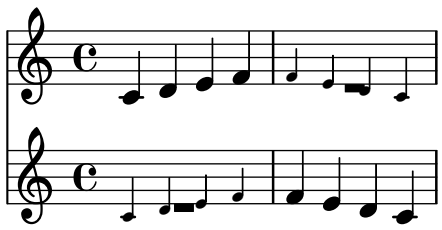
`quotedEventTypes` is also the fallback for cue notes if `quotedCueEventTypes` is not set.

`quote-cue-event-types.ly`
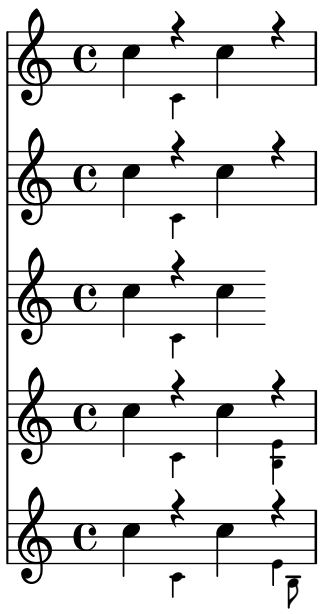


Two quoted voices may refer to each other. In this example, there are notes with each full-bar rest.

`quote-cyclic.ly`
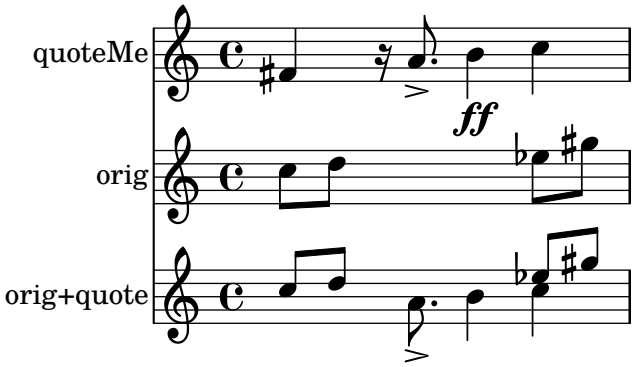


`\quoteDuring` and `\cueDuring` shall properly quote voices that create a sub-voice. The sub-voice will not be quoted, though. Exceptions are sections of parallel music `<< {...} \ {...} >>`, which will be quoted.

`quote-during-subvoice.ly`
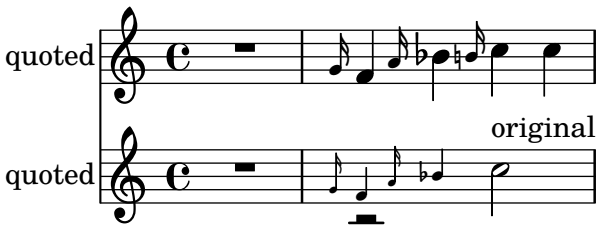


With `\cueDuring` and `\quoteDuring`, fragments of previously entered music may be quoted. `quotedEventTypes` will determines what things are quoted. In this example, a 16th rest is not quoted, since `rest-event` is not in `quotedEventTypes`.

`quote-during.ly`



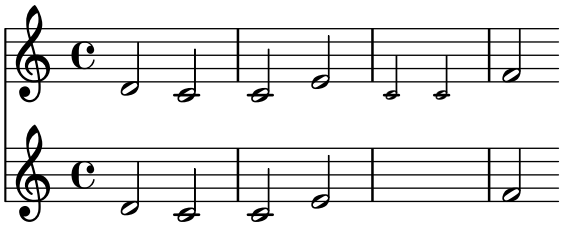
Quotes may contain grace notes. The grace note leading up to an unquoted note is not quoted.

quote-grace.ly



`\killCues` shall only remove real cue notes generated by `\cueDuring`, but not other music quoted using `\quoteDuring`.
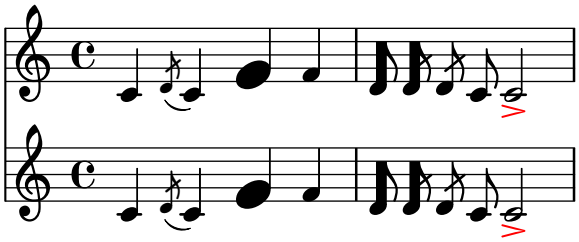
quote-kill-cues.ly



The `\quoteDuring` command shall also quote correctly all `\override`, `\once \override`, `\revert`, `\set`, `\unset` and `\tweak` events. The first line contains the original music, the second line quotes the whole music and should look identical.
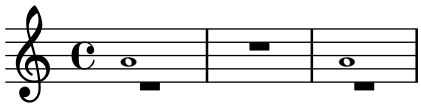
By default, not all events are quoted. By setting the quoted event types to `'(StreamEvent)`, everything should be quoted.

quote-overrides.ly



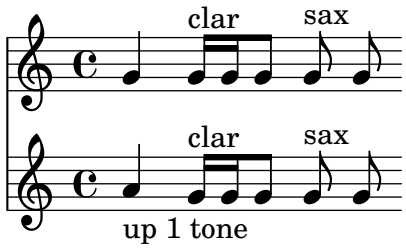
Voices from different cues must not be tied together. In this example, the first note has a tie. This note should not be tied to the second visible note (following the rest). Note that this behavior will not hold for cues in direct succession, since only one `CueVoice` context is created (with `context-id` 'cue').

quote-tie.ly



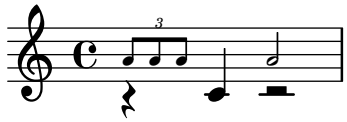
Quotations take into account the transposition of both source and target. In this example, all instruments play sounding central C, the target is a instrument in F. The target part may be `\transposed`. The quoted pitches will stay unchanged.

`quote-transposition.ly`



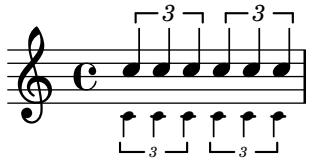
Tuplet bracket ends properly when quoting.

`quote-tuplet-end.ly`
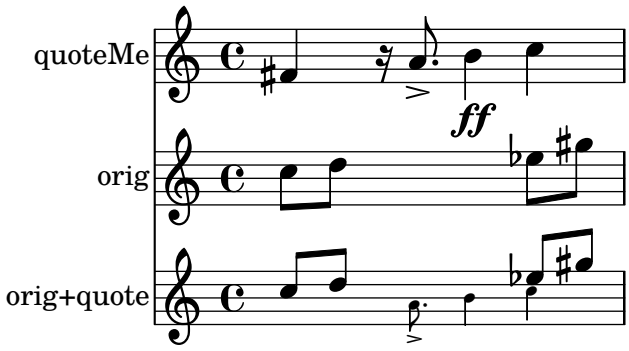


In cue notes, Tuplet stops are handled before new tuplets start.

`quote-tuplet.ly`
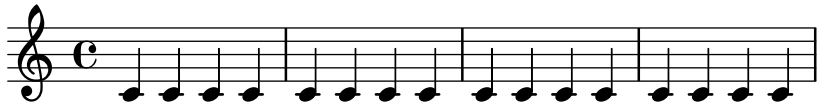


With `\quote`, fragments of previously entered music may be quoted. `quotedEventTypes` will determines what things are quoted. In this example, a 16th rest is not quoted, since `rest-event` is not in `quotedEventTypes`.

`quote.ly`
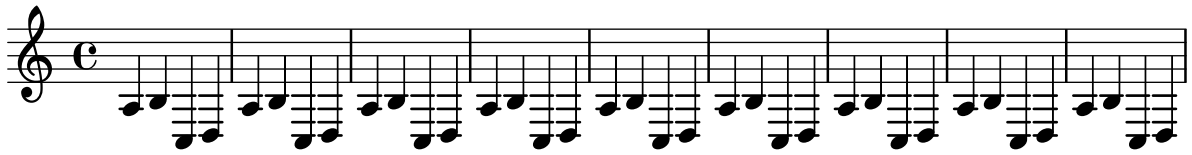


For a one-page score, ragged-bottom should have the same effect as ragged-last-bottom.

`ragged-bottom-one-page.ly`



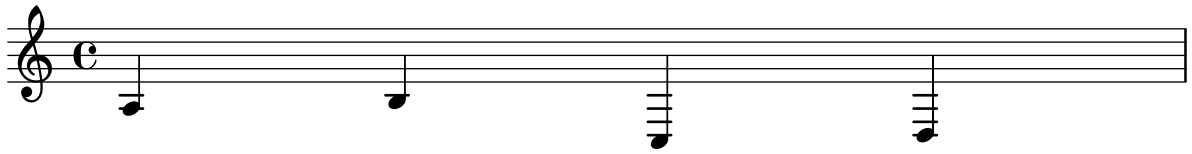
When a score takes up only a single line and it is compressed, it is not printed as ragged.

`ragged-right-compressed.ly`



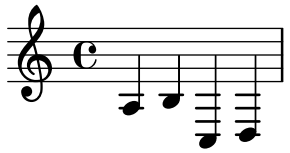
When ragged-right is specifically disabled, a score with only one line will not be printed as ragged.

`ragged-right-disabled.ly`



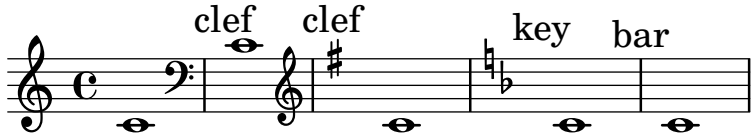
When a score takes up only a single line and it is stretched, it is printed as ragged by default.

`ragged-right-one-line.ly`
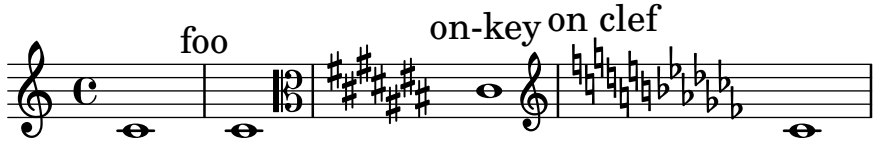


When the break-align-symbols property is given as a list, the alignment depends on which symbols are visible.

`rehearsal-mark-align-priority.ly`



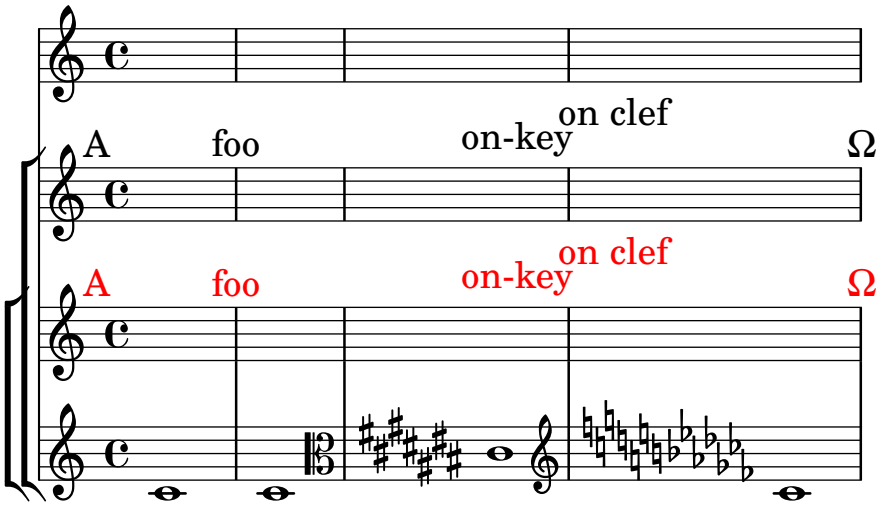
RehearsalMarks still align correctly if Mark_engraver is moved to Staff context.

`rehearsal-mark-align-staff-context.ly`



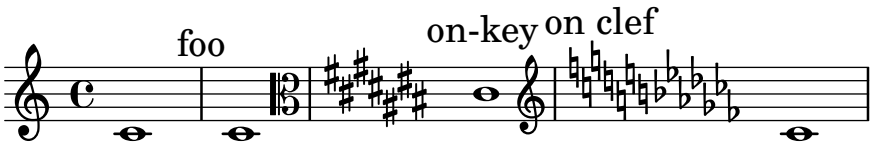
Rehearsal_number_engraver may be moved to staff-group contexts. Five marks should appear in black above the second staff from the top. The same marks should appear in red above the third staff from the top.

`rehearsal-mark-align-staff-group-context.ly`
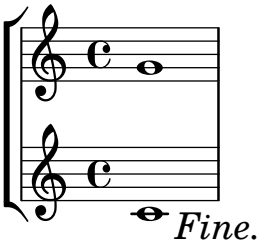


The rehearsal mark is put on top a breakable symbol, according to the value of `break-align-symbols` value of the `RehearsalMark`. The same holds for `BarNumber` grobs.

`rehearsal-mark-align.ly`



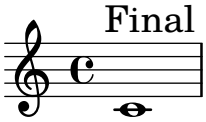
Rehearsal marks with direction DOWN get placed at the bottom of the score.

`rehearsal-mark-direction.ly`



Rehearsal marks at the end of the last measure of a score are automatically made visible.

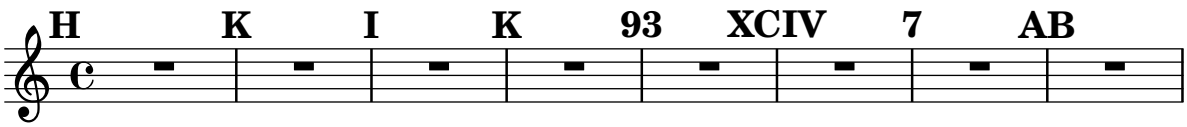
`rehearsal-mark-final-score.ly`

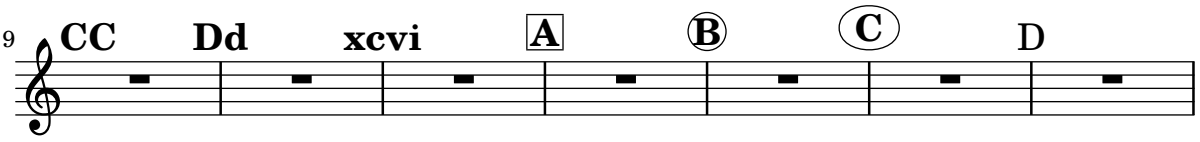


Comparison of markFormatter functions.

The marks should read H, K, I, K, 93, XCIV, 7, AB, CC, Dd, xcvi, boxed A, circled B, ovalled C, medium font D.

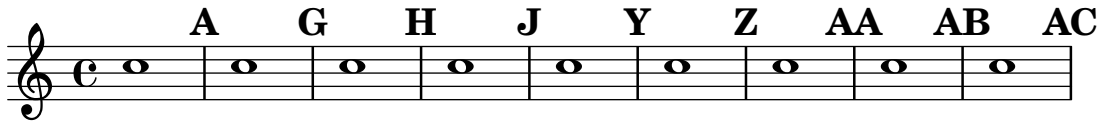
`rehearsal-mark-formatters.ly`





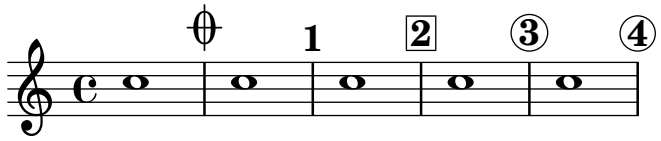
Rehearsal marks in letter style: the I is skipped, and after Z, double letters are used. The mark may be set with `\mark NUMBER`, or with `Score.rehearsalMark`.

`rehearsal-mark-letter.ly`



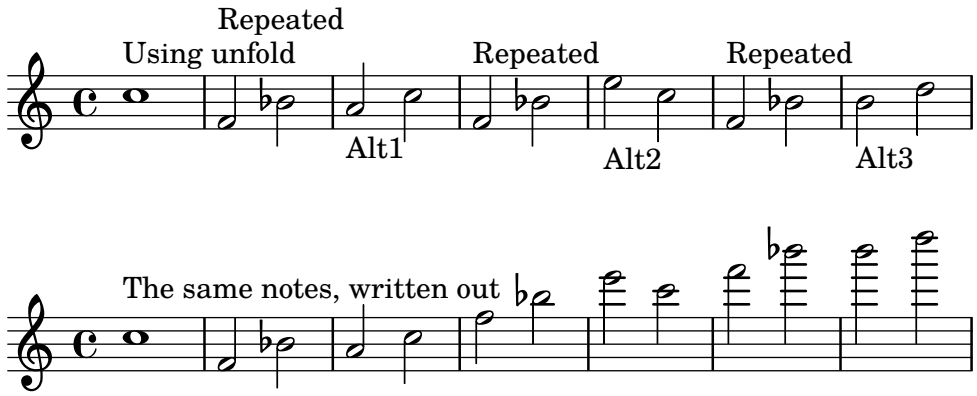
Marks can be printed as numbers. By setting `markFormatter` we may choose a different style of mark printing. Also, marks can be specified manually, with a markup argument.

`rehearsal-mark-number.ly`
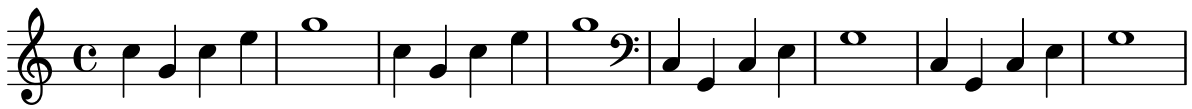


Using repeat unfold within a relative block gives a different result from writing the notes out in full. The first system has all the notes within the stave. In the second, the notes get progressively higher.

`relative-repeat.ly`



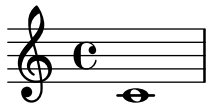
Notes are entered using absolute octaves, octaves relative to the previous note, or relative to a fixed octave.

`relative.ly`



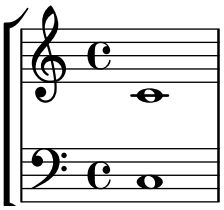
`\RemoveEmptyStaves` is defined separately from context definitions so it can be used outside of `\layout` blocks.
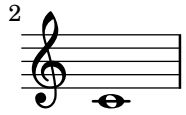
`remove-empty-context-mod.ly`

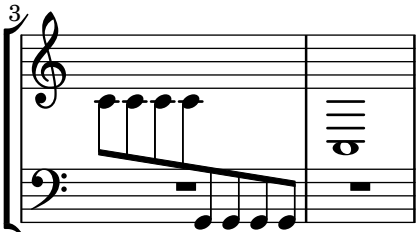


RemoveEmptyStaves should keep the pre-existing value of `auto-knee-gap`. In this case, the cross-staff beam should be between the two staves.

`remove-empty-staves-auto-knee.ly`
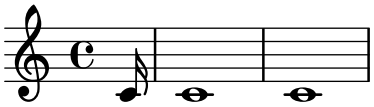






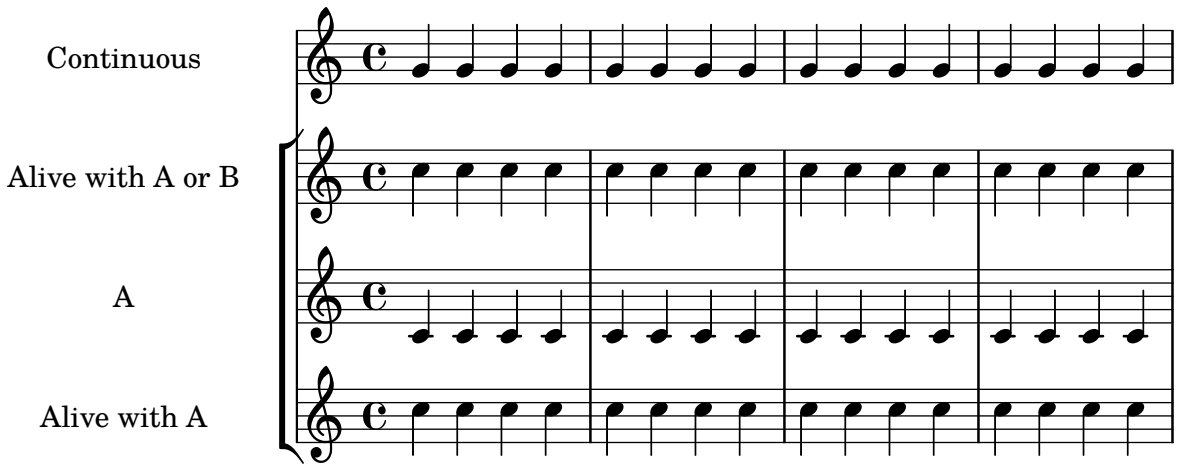
Rests should not keep staves alive when `\RemoveEmptyStaffContext` is active. The following example should have only one staff.
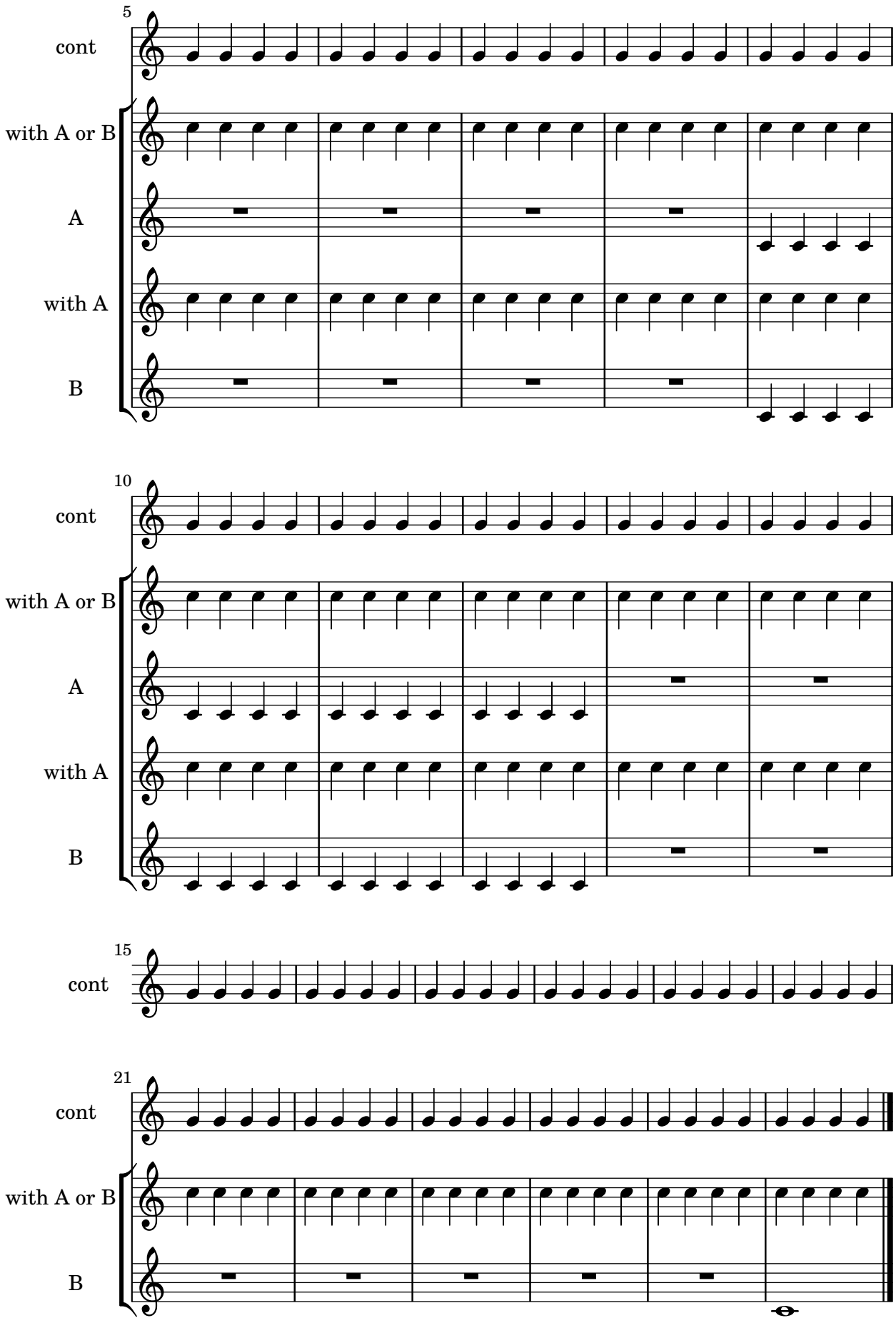
`remove-empty-staves-with-rests.ly`



The `VerticalAxisGroup.remove-layer` property can be used to keep staves alive with reference to other staves in the `Keep_alive_together_engraver` group.

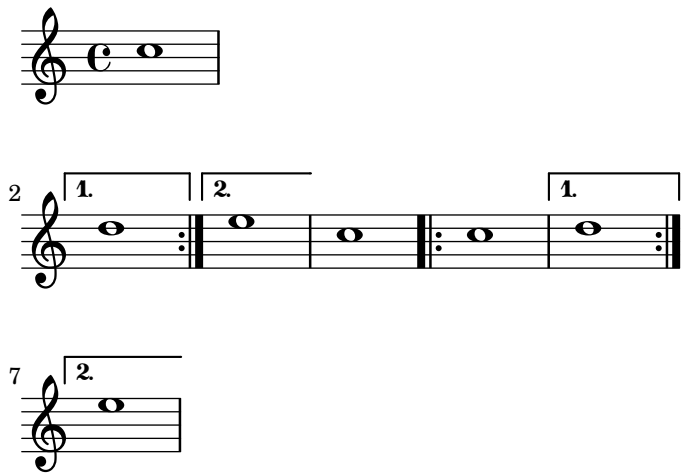
`remove-layer-symbol.ly`





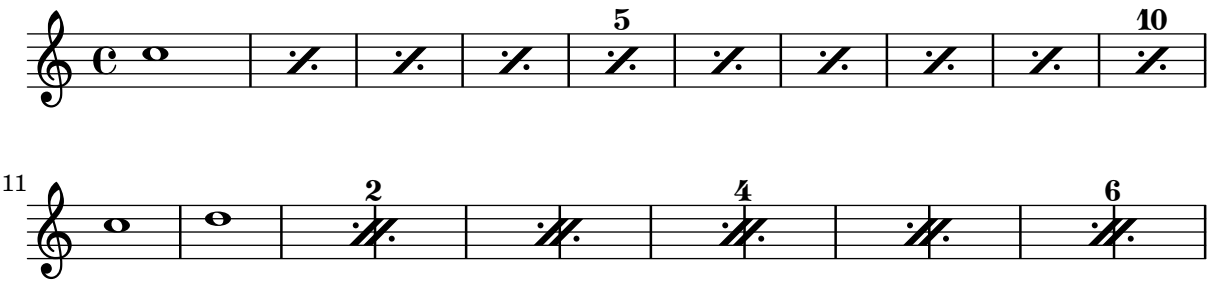
Across linebreaks, the left edge of a first and second alternative bracket should be equal.

`repeat-line-break.ly`
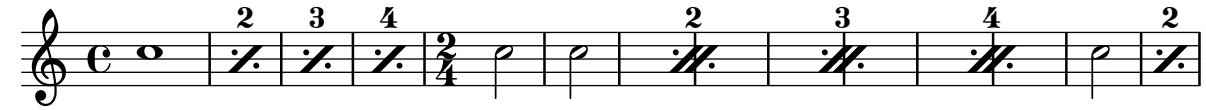


Percent repeat counters can be shown at regular intervals by setting `repeatCountVisibility`.

`repeat-percent-count-visibility.ly`



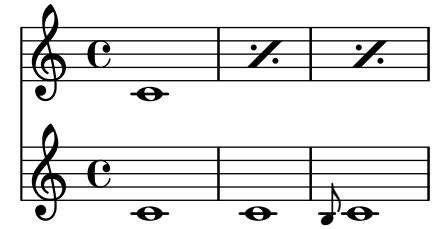
Percent repeats get incremental numbers when `countPercentRepeats` is set, to indicate the repeat counts, but only if there are more than two repeats.

`repeat-percent-count.ly`



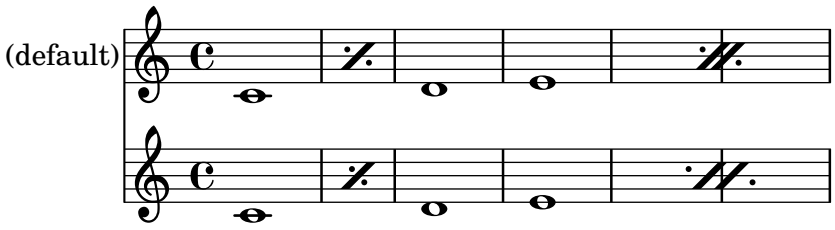
Percent repeats are also centered when there is a grace note in a parallel staff.

`repeat-percent-grace.ly`
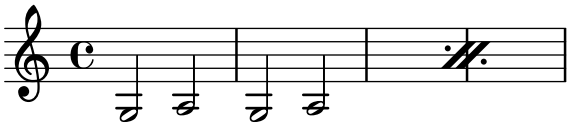


The positioning of dots and slashes in percent repeat glyphs can be altered using `dot-negative-kern` and `slash-negative-kern`.

`repeat-percent-kerning.ly`



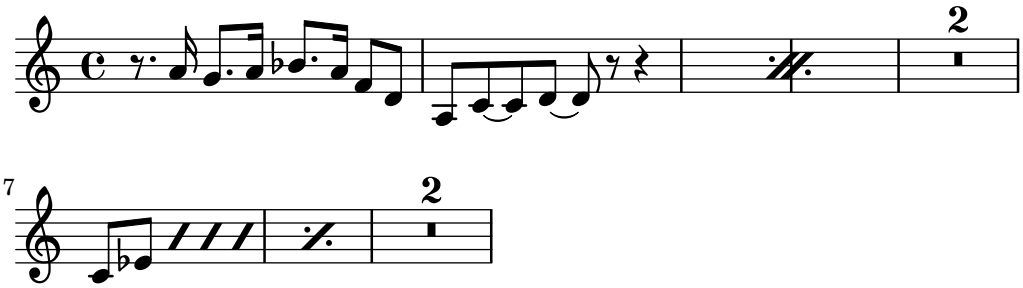
Percent repeats are not skipped, even when `skipBars` is set.

`repeat-percent-skipbars.ly`
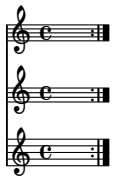


Measure repeats may be nested with beat repeats.

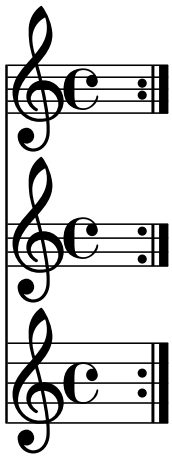
`repeat-percent.ly`



The two dots of a repeat sign should be symmetric to the staff centre and avoid staff lines even for exotic staves. Test set-global-staff size 10 (with layout-set-staff-size).

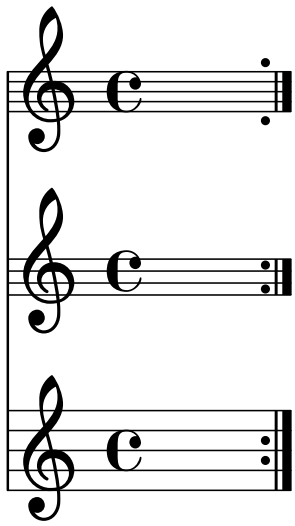
`repeat-sign-global-size-10.ly`

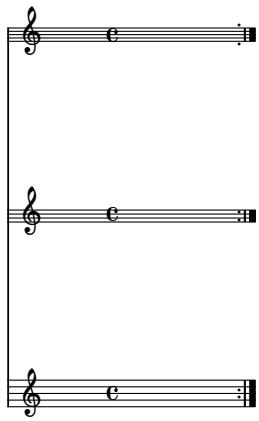




The two dots of a repeat sign should be symmetric to the staff centre and avoid staff lines even for exotic staves. Test set-global-staff size 30 (with layout-set-staff-size).

`repeat-sign-global-size-30.ly`

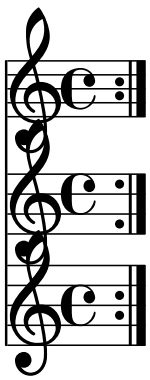




The two dots of a repeat sign should be symmetric to the staff centre and avoid staff lines even for exotic staves. Test set-global-staff size 10 (with layout-set-staff-size).

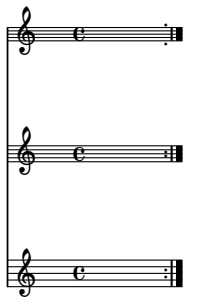
`repeat-sign-global-size-5.ly`

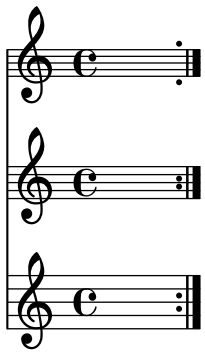




The two dots of a repeat sign should be symmetric to the staff centre and avoid staff lines even for exotic staves. Test layout-set-staff-size.

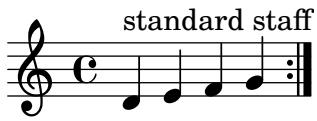
`repeat-sign-layout-size.ly`

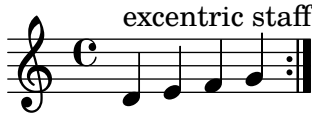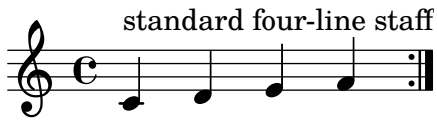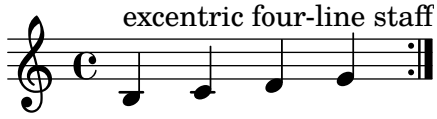




The two dots of a repeat sign should be symmetric to the staff centre and avoid staff lines even for exotic staves.

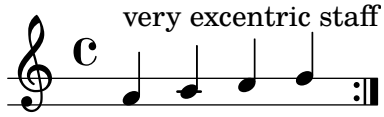
`repeat-sign.ly`
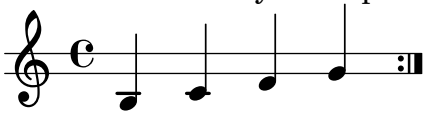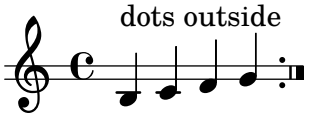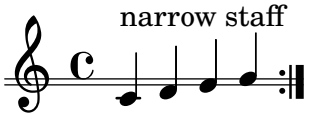










widened by staff-space
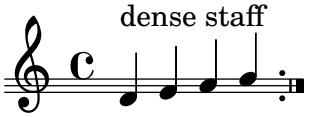


dots outside
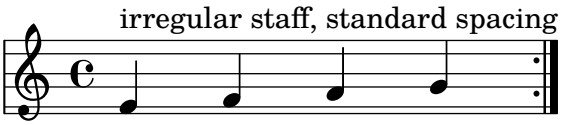


narrow staff



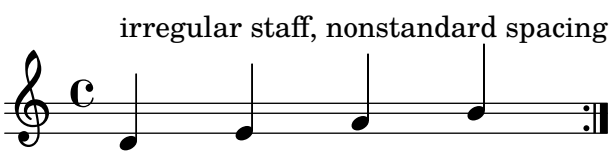
dense staff



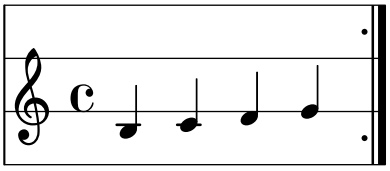
irregular staff, standard spacing



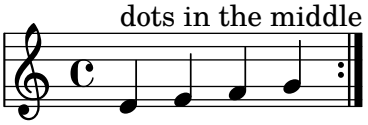
irregular staff, nonstandard spacing
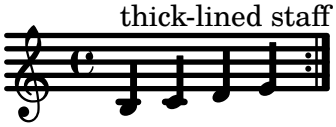


dots in outer spaces



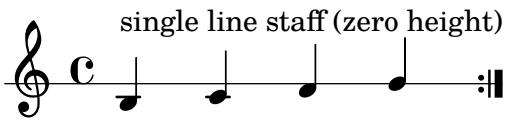
dots in the middle
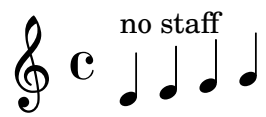


thick-lined staff



single line staff (zero height)

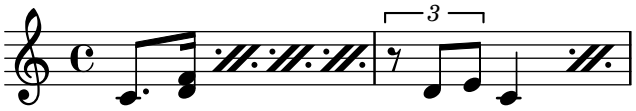


no staff
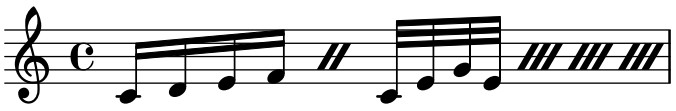


Beat repeats for patterns containing mixed durations use a double percent symbol.

`repeat-slash-mixed.ly`
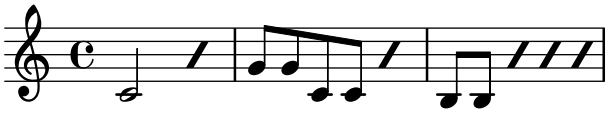


Beat repeats for patterns containing identical durations shorter than an eighth note use multiple slashes.

`repeat-slash-multi.ly`



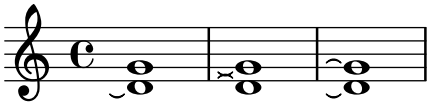
Within a bar, beat repeats denote that a music snippet should be played again.

`repeat-slash.ly`
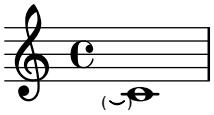


`\repeatTie` ties should also work on individual notes of a chord.

`repeat-tie-chords.ly`
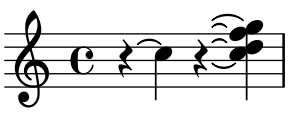


A `\repeatTie` may be parenthesized.

`repeat-tie-parenthesize.ly`



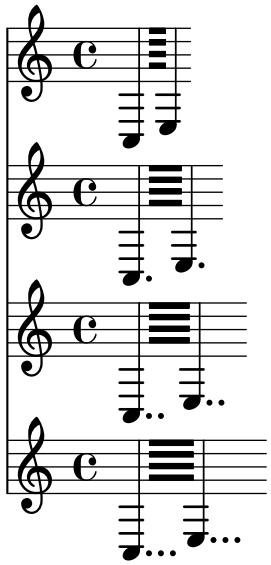
Repeat ties are only connected on the right side to a note head.

`repeat-tie.ly`
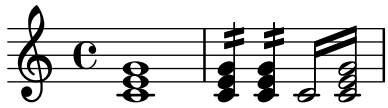


Each of the staves here should have four tremolo beams.

`repeat-tremolo-beams.ly`
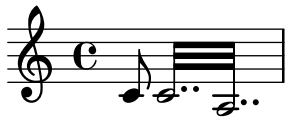


Tremolos work with chord repetitions.

`repeat-tremolo-chord-rep.ly`



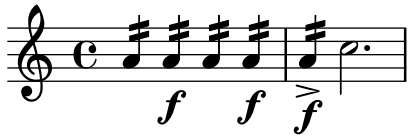
Dots are added to tremolo notes if the durations involved require them.

`repeat-tremolo-dots.ly`
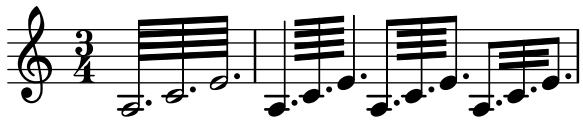


A tremolo repeat containing only one note (no sequential music) shall not be scaled. An articulation or dynamic sign on the note should not confuse lilypond.

`repeat-tremolo-one-note-articulation.ly`
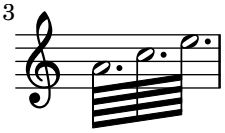


A tremolo can have more than two notes. Also check that linebreaks between tremolos still work and that empty tremolos don't crash.

`repeat-tremolo-three-notes.ly`

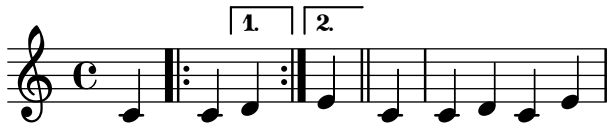


3



Volta repeats may be unfolded through the music function `\unfoldRepeats`.
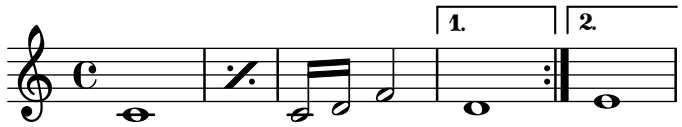
`repeat-unfold-all.ly`



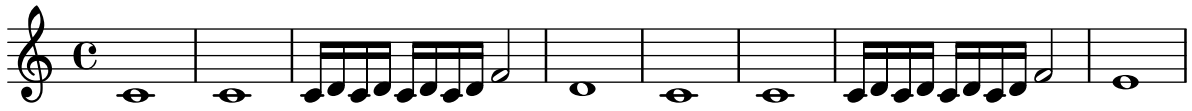
The music function `\unfoldRepeats` can take an optional argument-list specifying which type(s) of repeated music has to be unfolded.

`repeat-unfold-partial.ly`
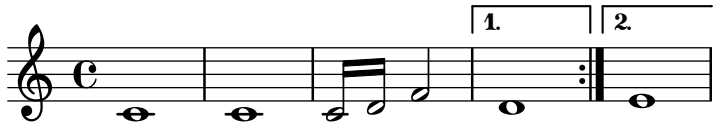
not expanding
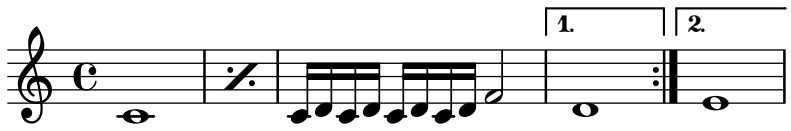


expanding all



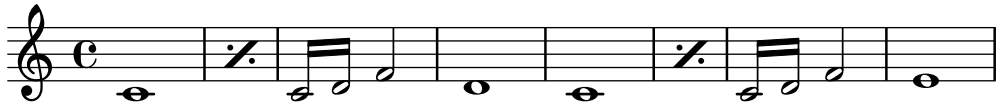
expanding percent-repeated-music
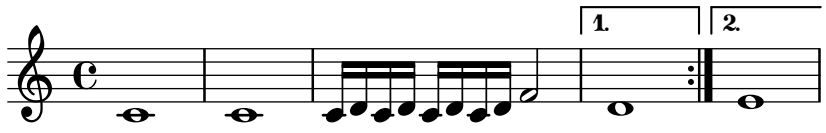


expanding tremolo-repeated-music



expanding volta-repeated-music

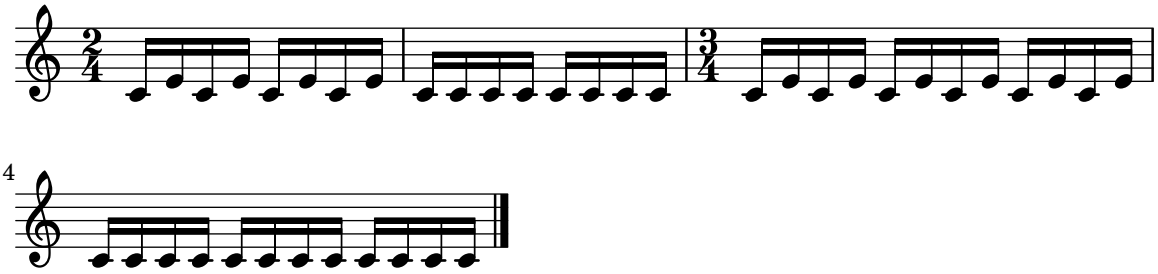


combinations are possible:
expanding percent-repeated-music and tremolo-repeated-music



Unfolding tremolo repeats. All fragments fill one measure with 16th notes exactly.
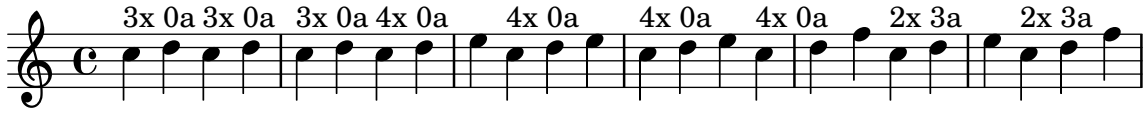
`repeat-unfold-tremolo.ly`



LilyPond has two modes for repeats: unfolded and semi-unfolded. Unfolded repeats are fully written out. Semi unfolded repeats have the body written and all alternatives sequentially. 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.

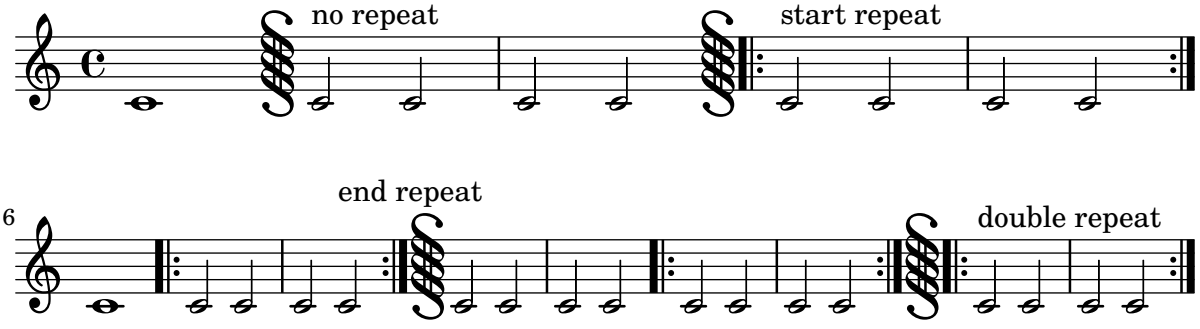
Unfolded behavior:

`repeat-unfold.ly`



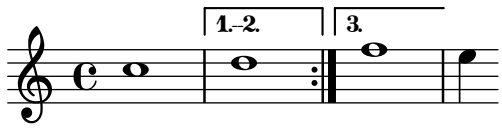
The segno sign should be automatically combined with the appropriate repeat bar line when `\inStaffSegno` is used.

`repeat-volta-segno.ly`



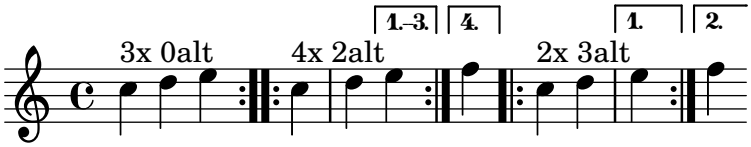
When too few alternatives are present, the first alternative is repeated, by printing a range for the 1st repeat.

`repeat-volta-skip-alternatives.ly`



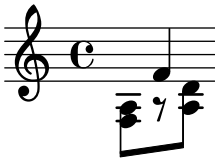
Volta (Semi folded) behavior. Voltas can start on non-barline moments. If they don't barlines should still be shown.

`repeat-volta.ly`
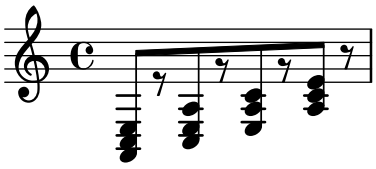


Beam/rest collision resolution and normal rest/note collisions can be combined.

`rest-collision-beam-note.ly`
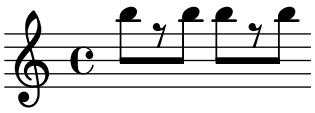


Rests under beams are moved by whole staff spaces.

`rest-collision-beam-quantized.ly`
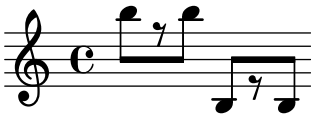


Beam/rest collision takes offset due to `Rest #'direction` into account properly.

`rest-collision-beam-restdir.ly`
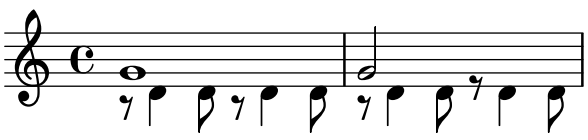


Rests under beams are shifted upon collision.

`rest-collision-beam.ly`



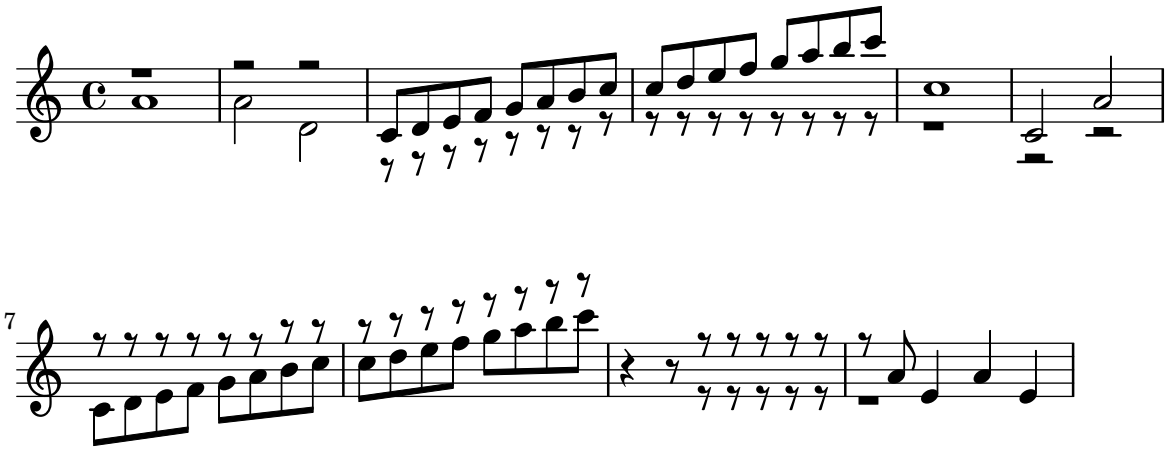
Vertical rest positions in a multi-voice staff should obey the duration of notes; this is, they shouldn't return to a default position too early.

`rest-collision-note-duration.ly`
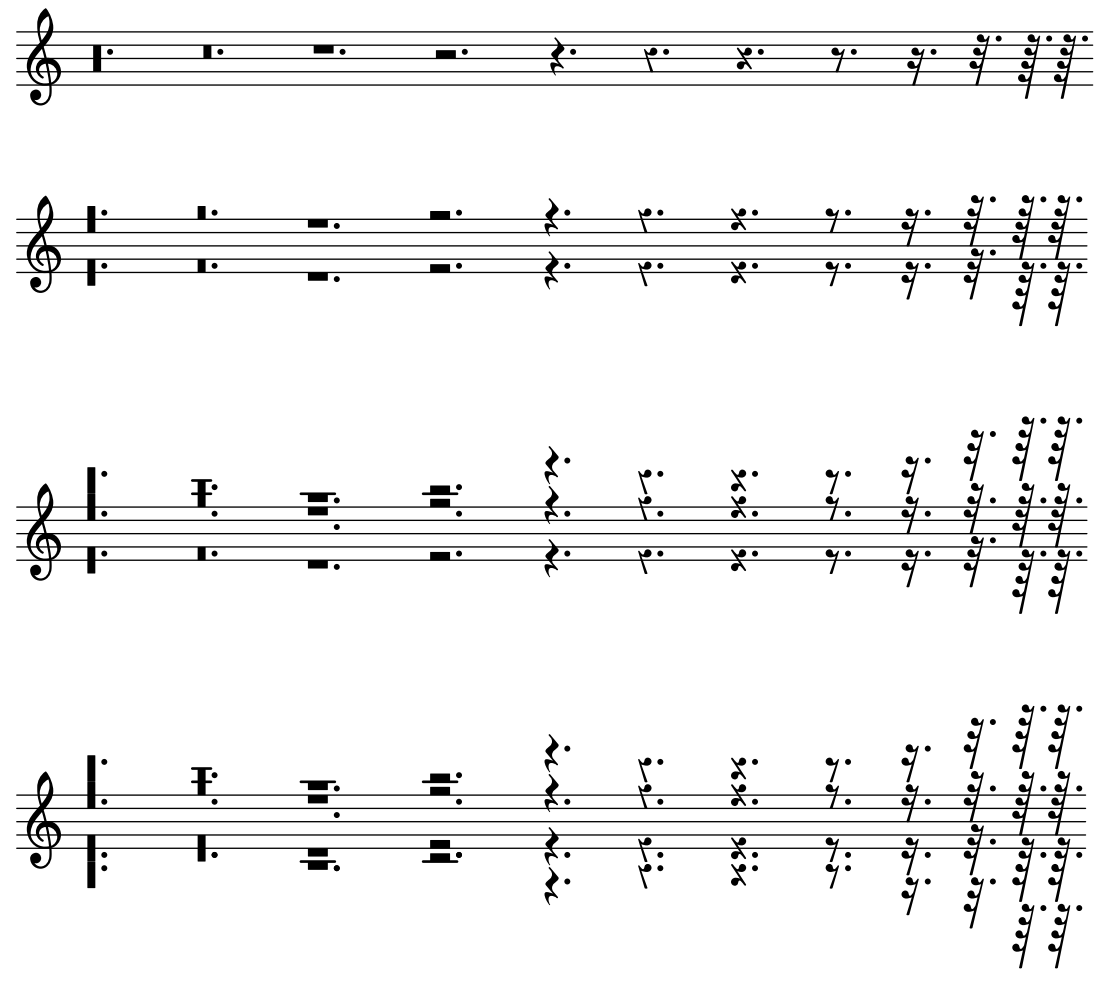


Rests should not collide with beams, stems and noteheads. Rests may be under beams. Rests should be move by integral number of spaces inside the staff, and by half spaces outside. Notice that the half and whole rests just outside the staff get ledger lines in different cases.

`rest-collision.ly`



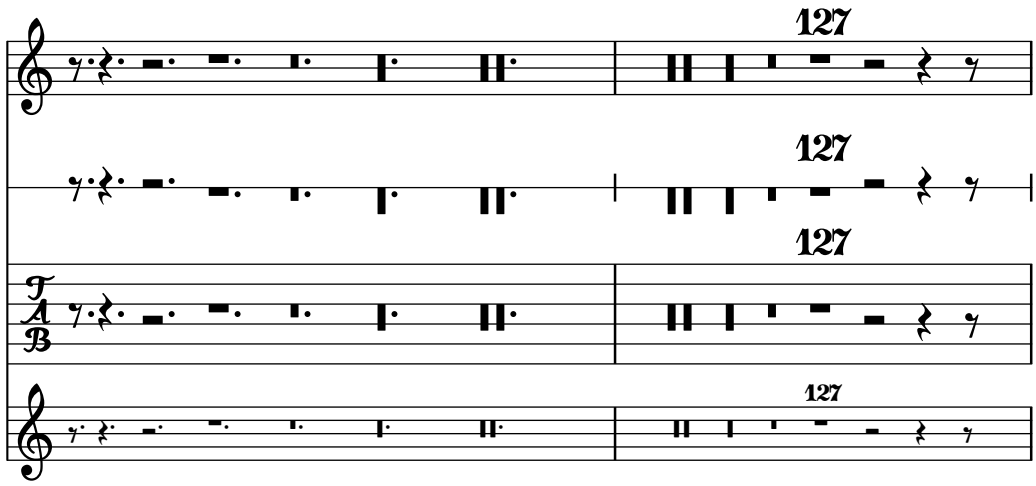
Dots of rests should follow the rest positions.

`rest-dot-position.ly`



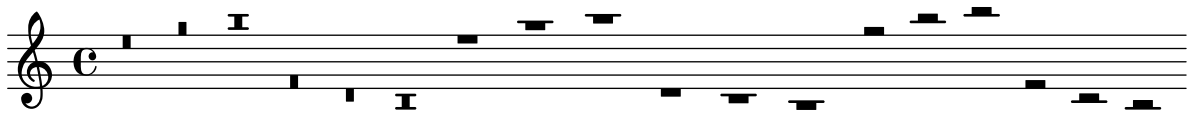
Breve, longa, and maxima rests should hang down from staff lines in one line staves, different staff space and font size.

`rest-hanging-breve.ly`
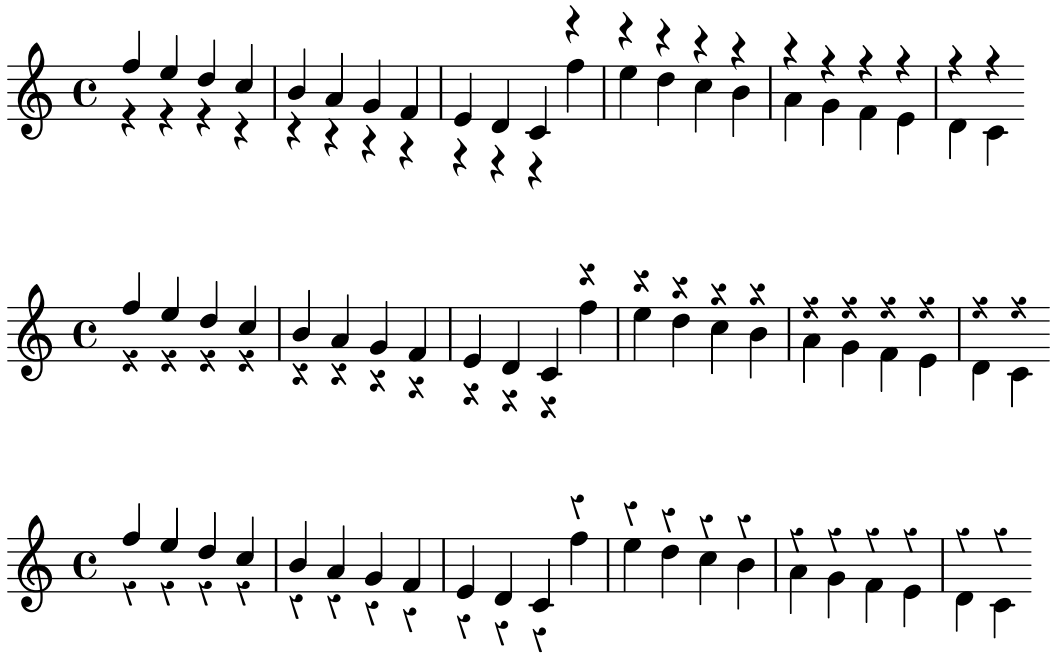


Breve, whole and half rests moving outside the staff should get ledger lines.

`rest-ledger.ly`



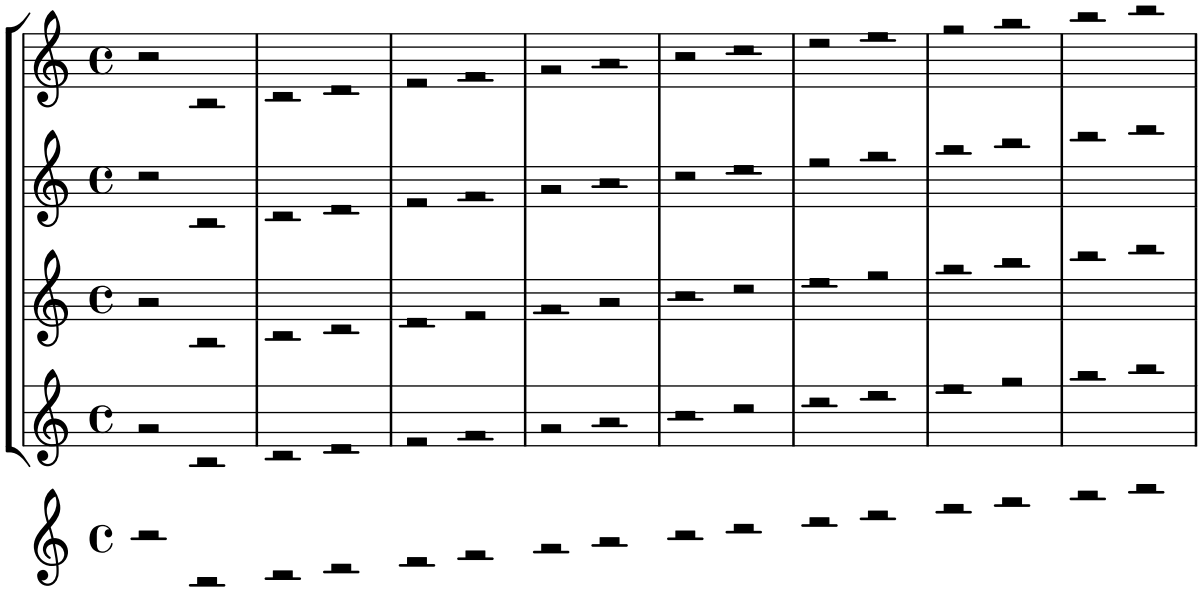
In rest-note collisions, the rest moves in discrete steps, and inside the staff, it moves in whole staff spaces.

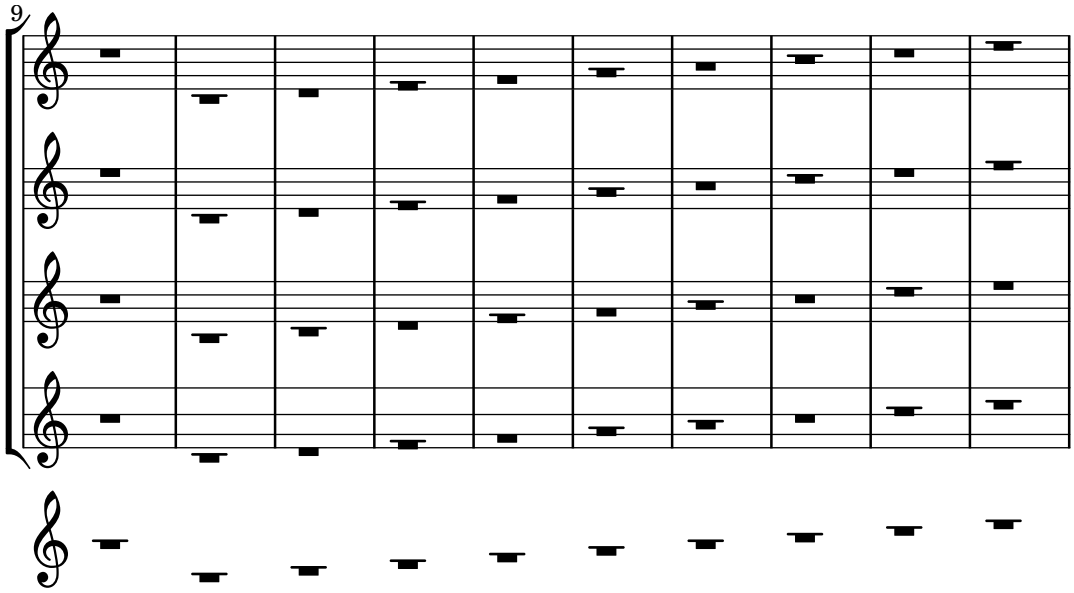
`rest-note-collision.ly`

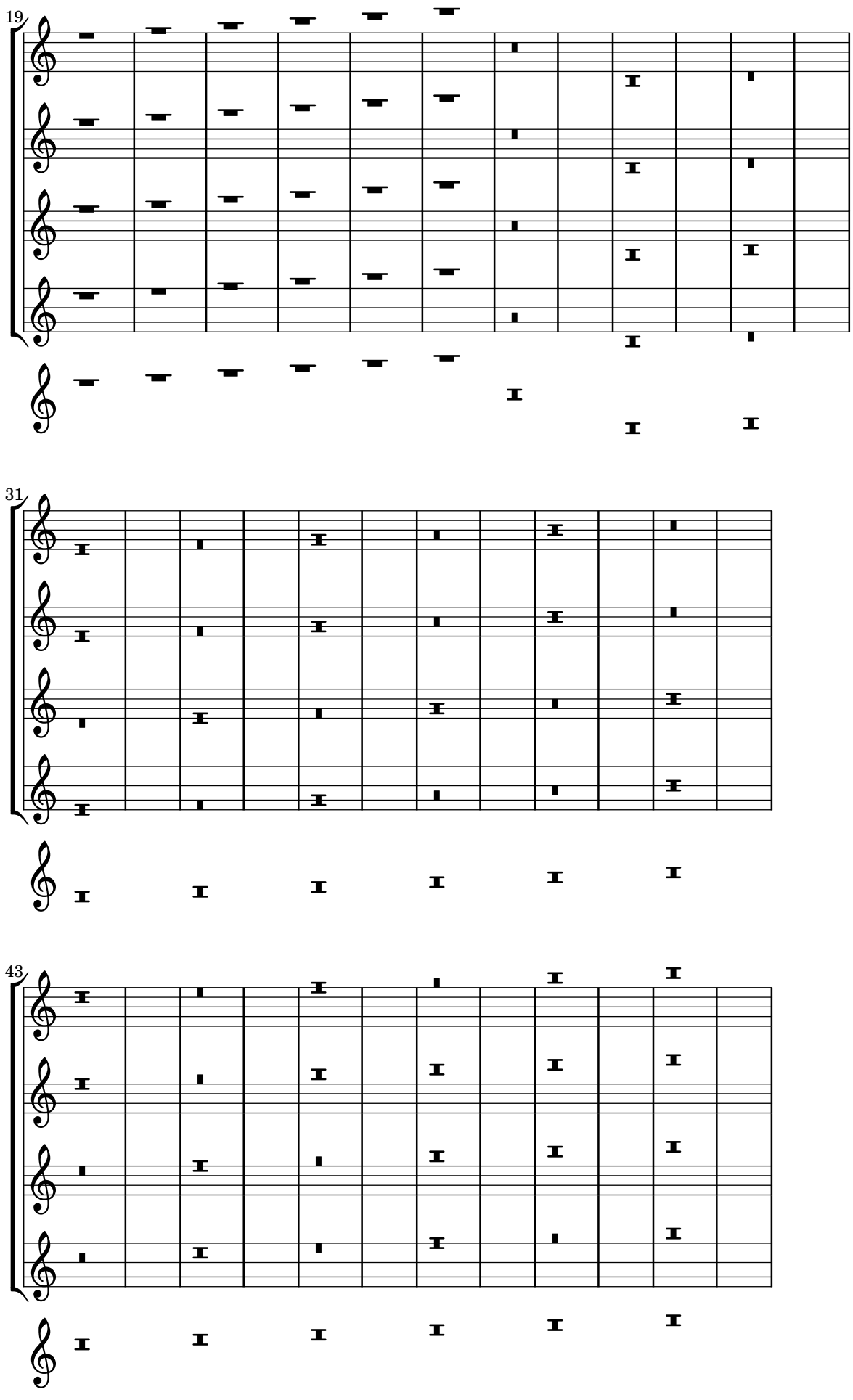


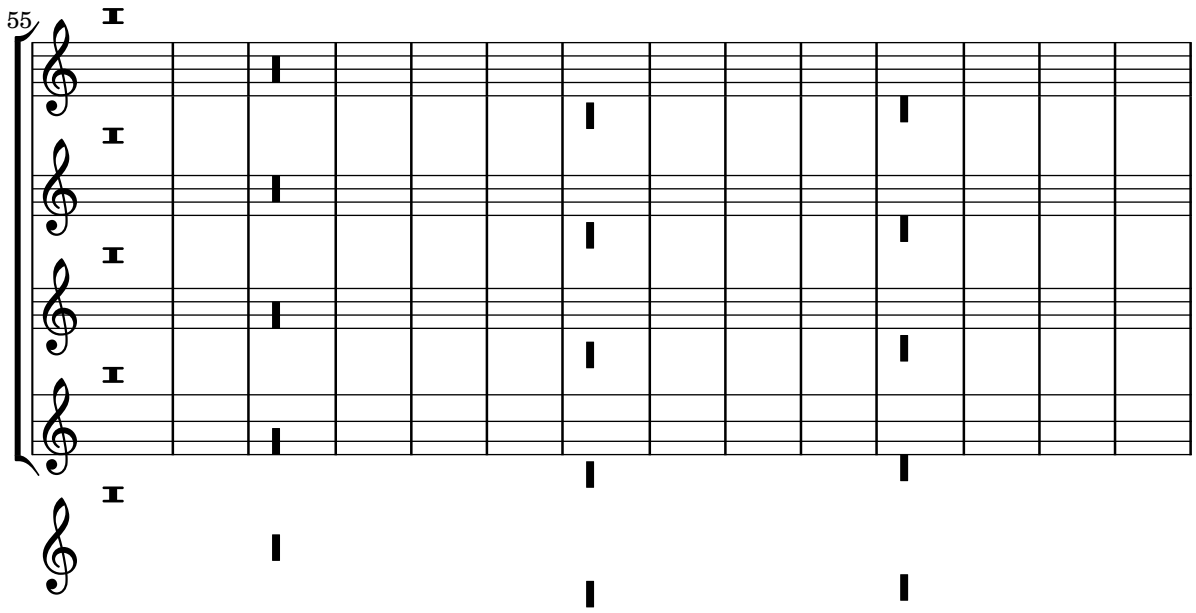
half rests should lie on a staff line, whole rests should hang from a staff line by default even for non-standard staves, except when the position is set by pitch.

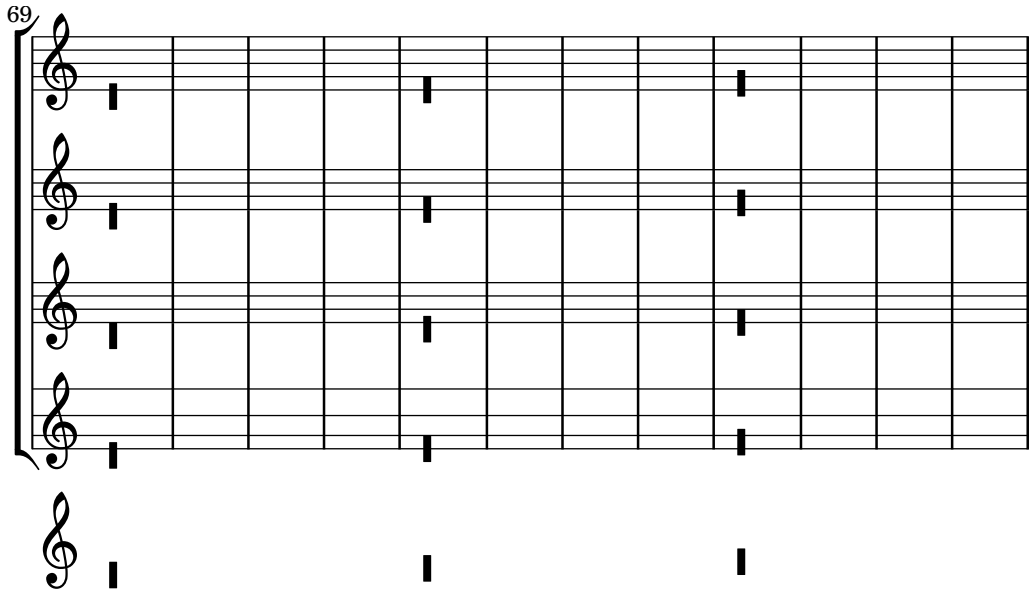
rest-on-nonstandard-staff.ly

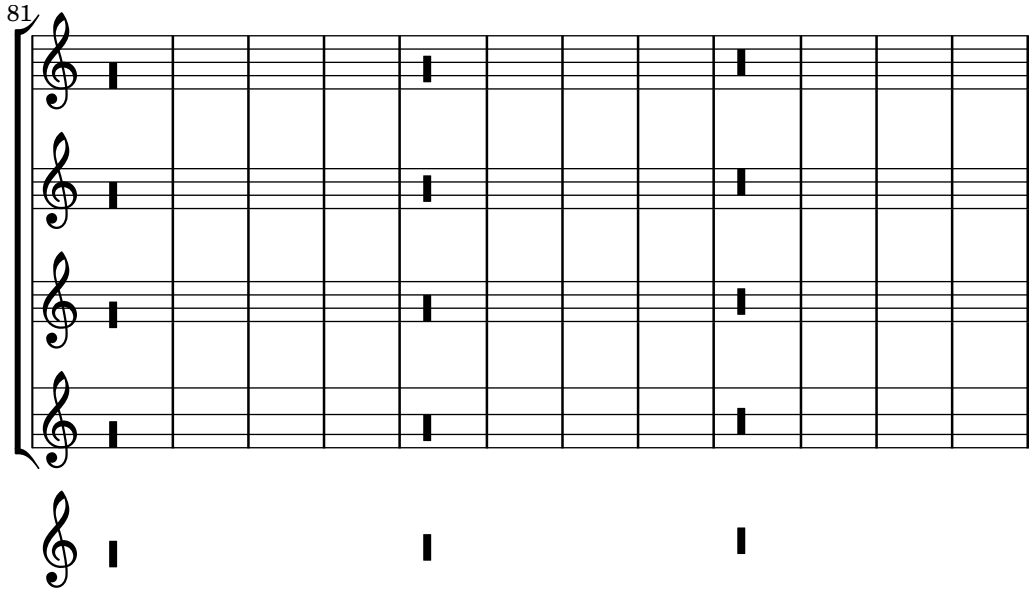


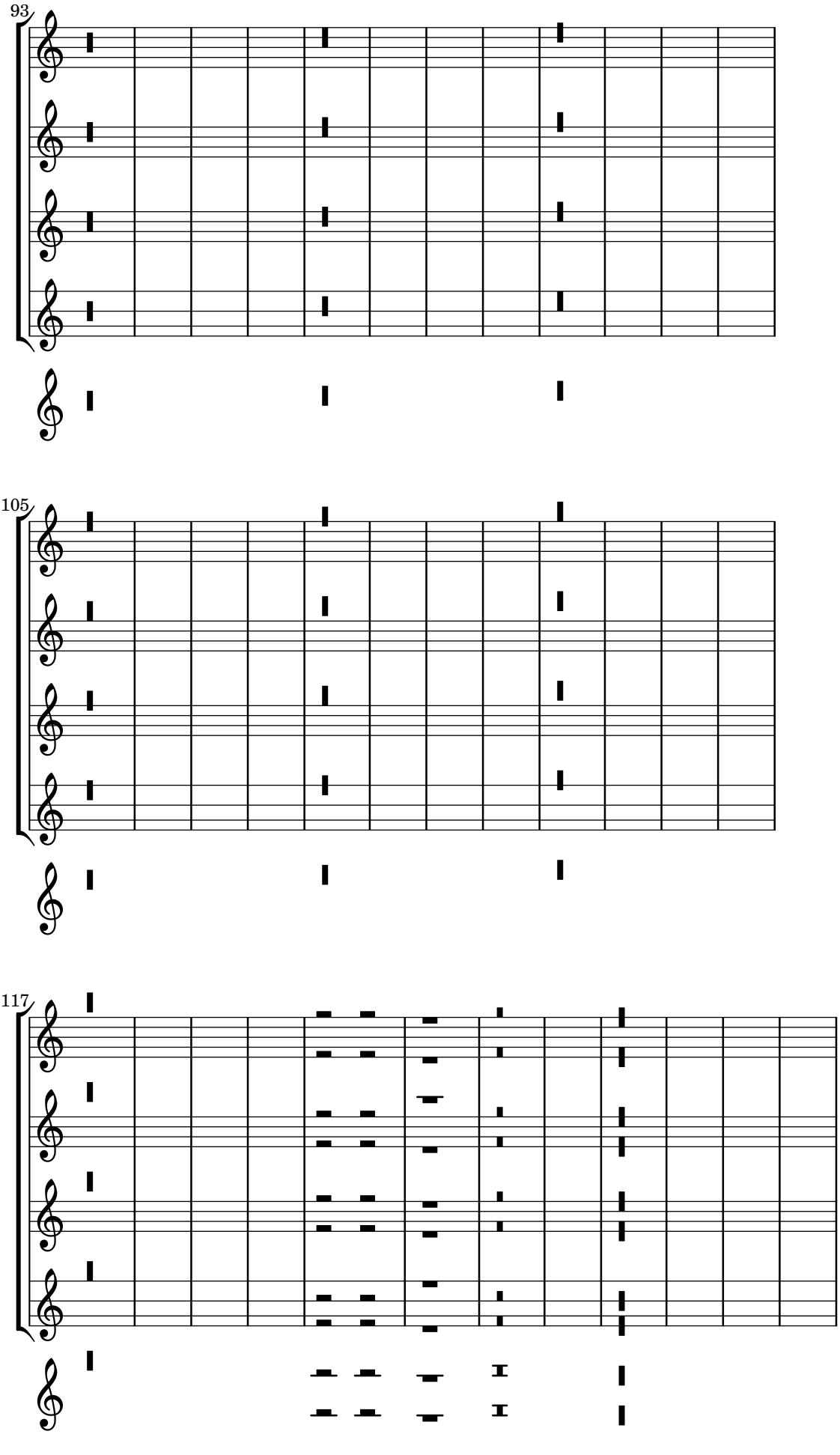



19
31
43


55
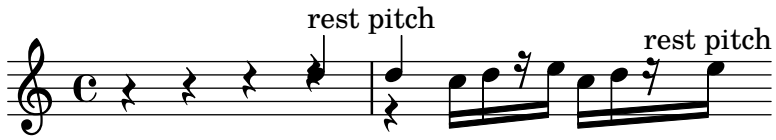

69


81


93
105
117
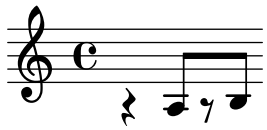
Rests can have pitches – these will be affected by transposition and relativization. If a rest has a pitch, rest/rest and beam/rest collision resolving will leave it alone.

`rest-pitch.ly`
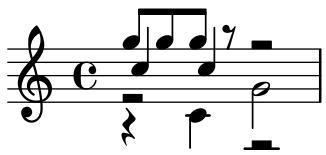


Pitched rests under beams.

`rest-pitched-beam.ly`
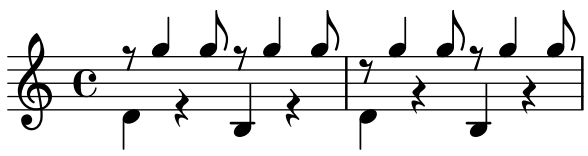


Rests avoid notes. Each rest is moved in the direction of the stems in its voice. Rests may overlap other rests in voices with the same stem direction, in which case a warning is given, but is suppressed if the rest has a pitch.

`rest-polyphonic-2.ly`



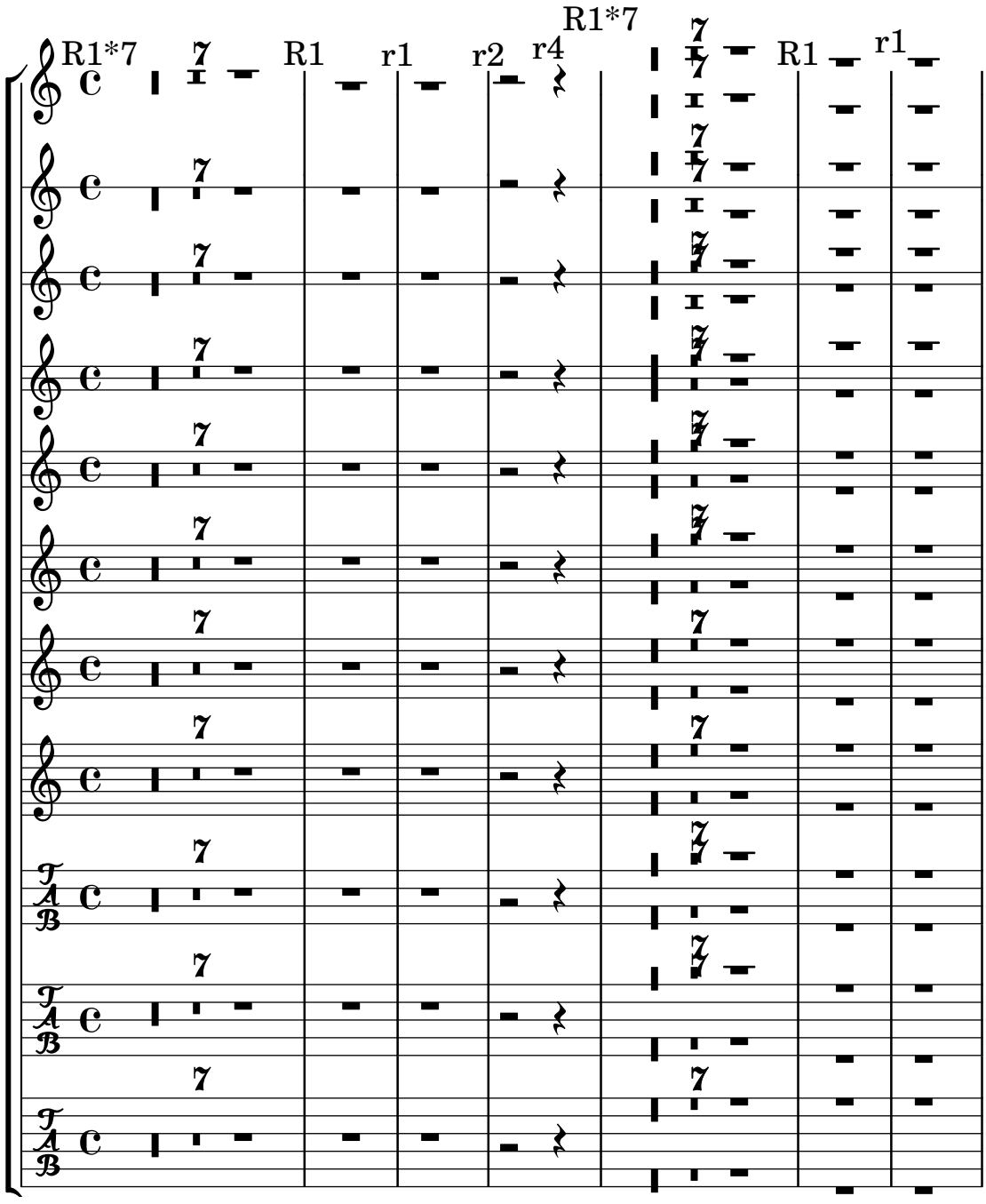
In polyphonic situations, rests are moved according to their `direction` even if there is no opposite note or rest. The amount in `staff-positions` is set by `voiced-position`.

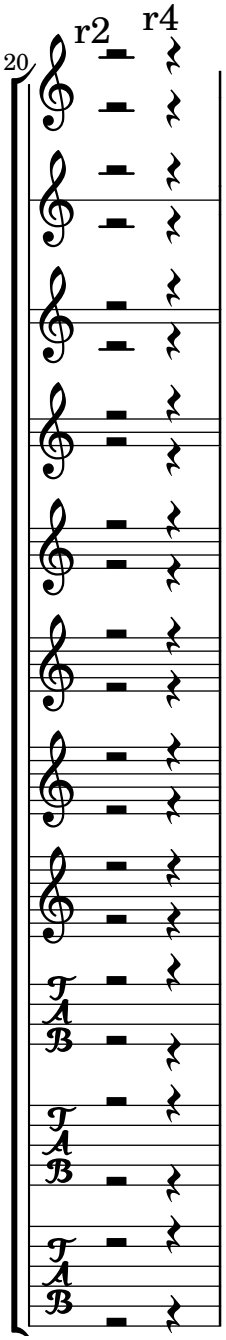
`rest-polyphonic.ly`



This shows the single and multi voice rest positions for various standard and tab staffs.

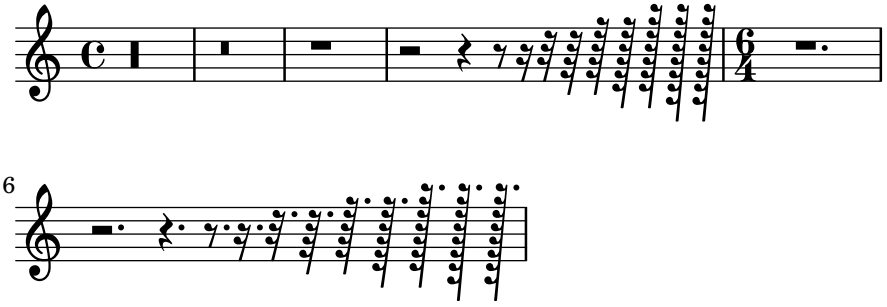
rest-positioning.ly





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.

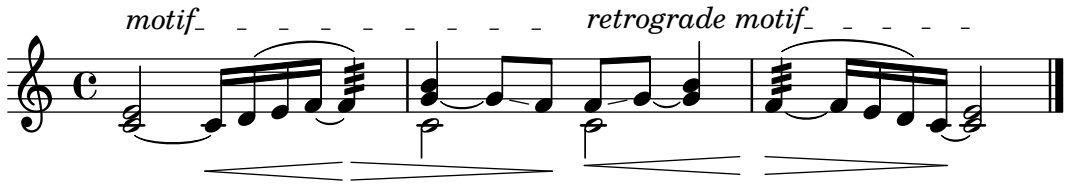
`rest.ly`



`\retrograde` can deal with crescendo and decrescendo as long as they are properly paired with `\endcr`/`\!` and `\enddecr`. Direction modifiers on slurs like `^(` need to be repeated as `^)` at

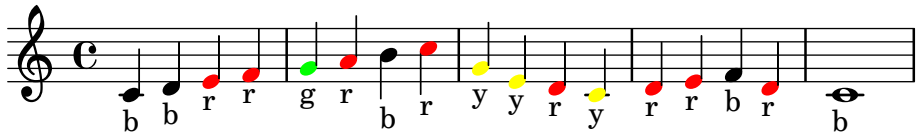
the end. Ties and glissandi work mostly (in-chord ties are turned into ordinary per-chord/note ties, however).

`retrograde.ly`



`\once \revert` can be used for reverting a property once rather than permanently.

`revert-once.ly`



Durations without pitches are placed into note events without pitch information. Those are directly useful in `RhythmicStaff`.

`rhythmic-sequence.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.

`rhythmic-staff.ly`

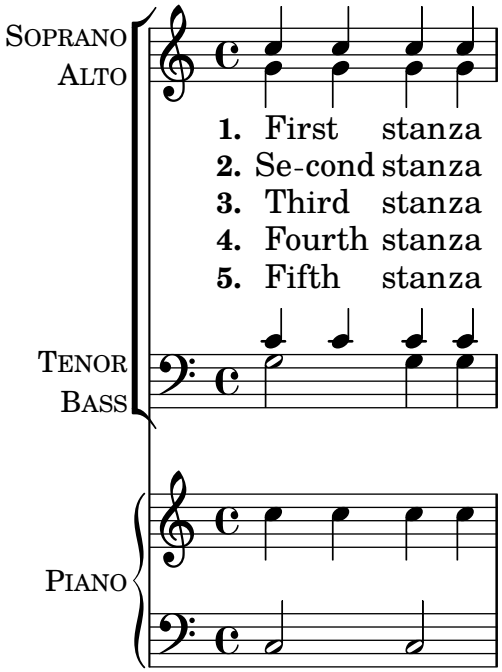


This should not survive lilypond –safe-mode

`safe.ly`

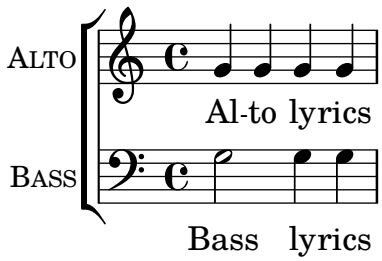
This should produce an SATB score on two staves with 5 verses and piano accompaniment.

satb-template-on-two-staves-with-verses.ly



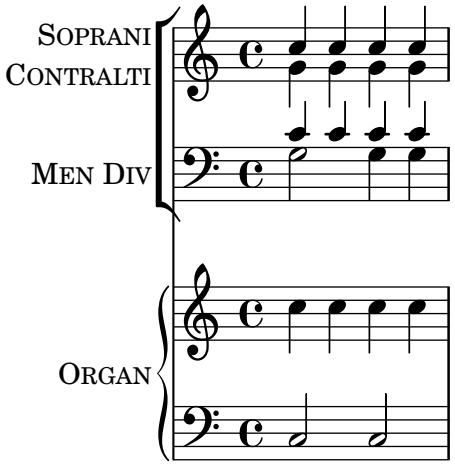
Soprano and tenor voices may be omitted without error, even when TwoVoicesPerStaff is specified and Alto and Bass lyrics are provided.

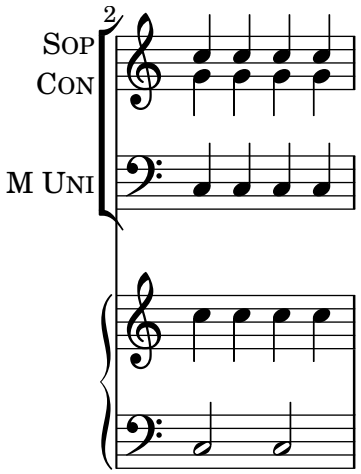
satb-template-soprano-and-tenor-may-be-omitted.ly



Instrument names and short instrument names can be changed when using the satb built-in template.

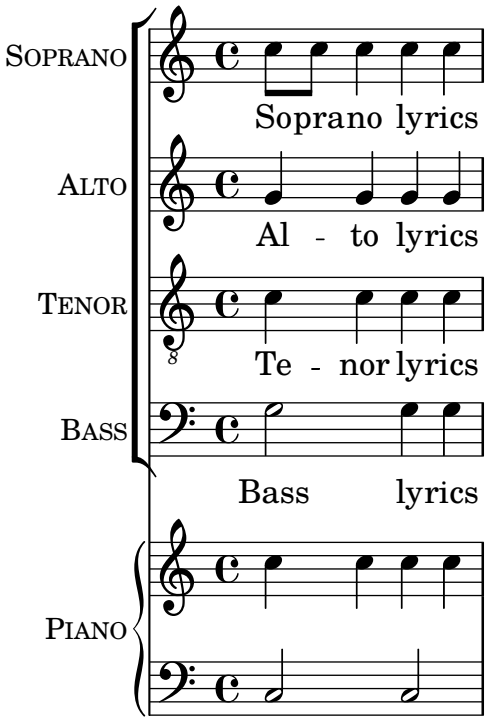
satb-template-with-changed-instrument-names.ly
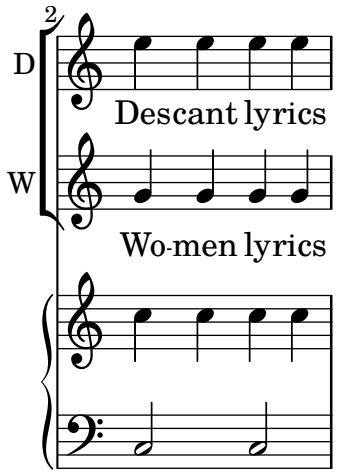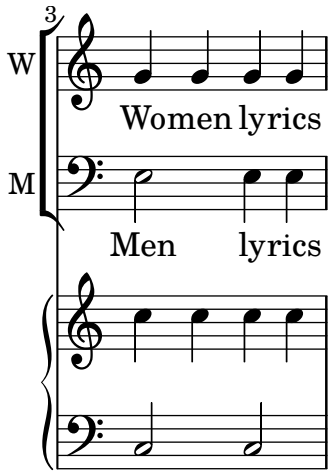




This should produce an SATB score with piano accompaniment, with four voices in the first system, unison women voices with descant in the second system and unison women and unison men voices in the third system.

`satb-template-with-men-women-and-descant.ly`







Scores can be generated with scheme, too, and inserted into the current book(part). Generated and explicit scores can be mixed, the header informations from top- and booklevel stack correctly.

`scheme-book-scores.ly`

Main Title

Main subtitle

Score with a c

Piecetitle
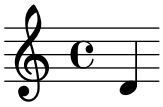


Title 1

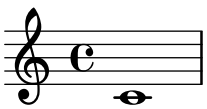
Sub1

Score with a d

Piecetitle
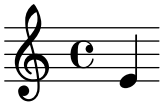


Piecetitle



Score with a e

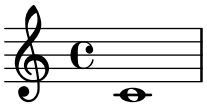
Piecetitle



Main Title
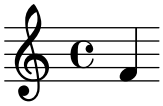
Main subtitle

Piecetitle



Score with a f

Piecetitle
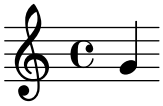


Main Title

Main subtitle

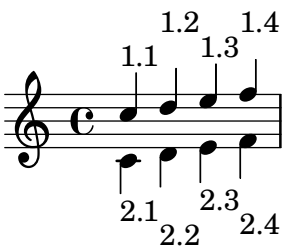
Score with a g

Piecetitle
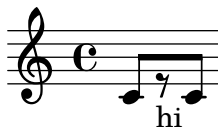


Scheme engravers may be instantiated, with instance-scoped slots, by defining a 1 argument procedure which shall return the engraver definition as an alist, with the private slots defined in a closure. The argument procedure argument is the context where the engraver is instantiated.

`scheme-engraver-instance.ly`
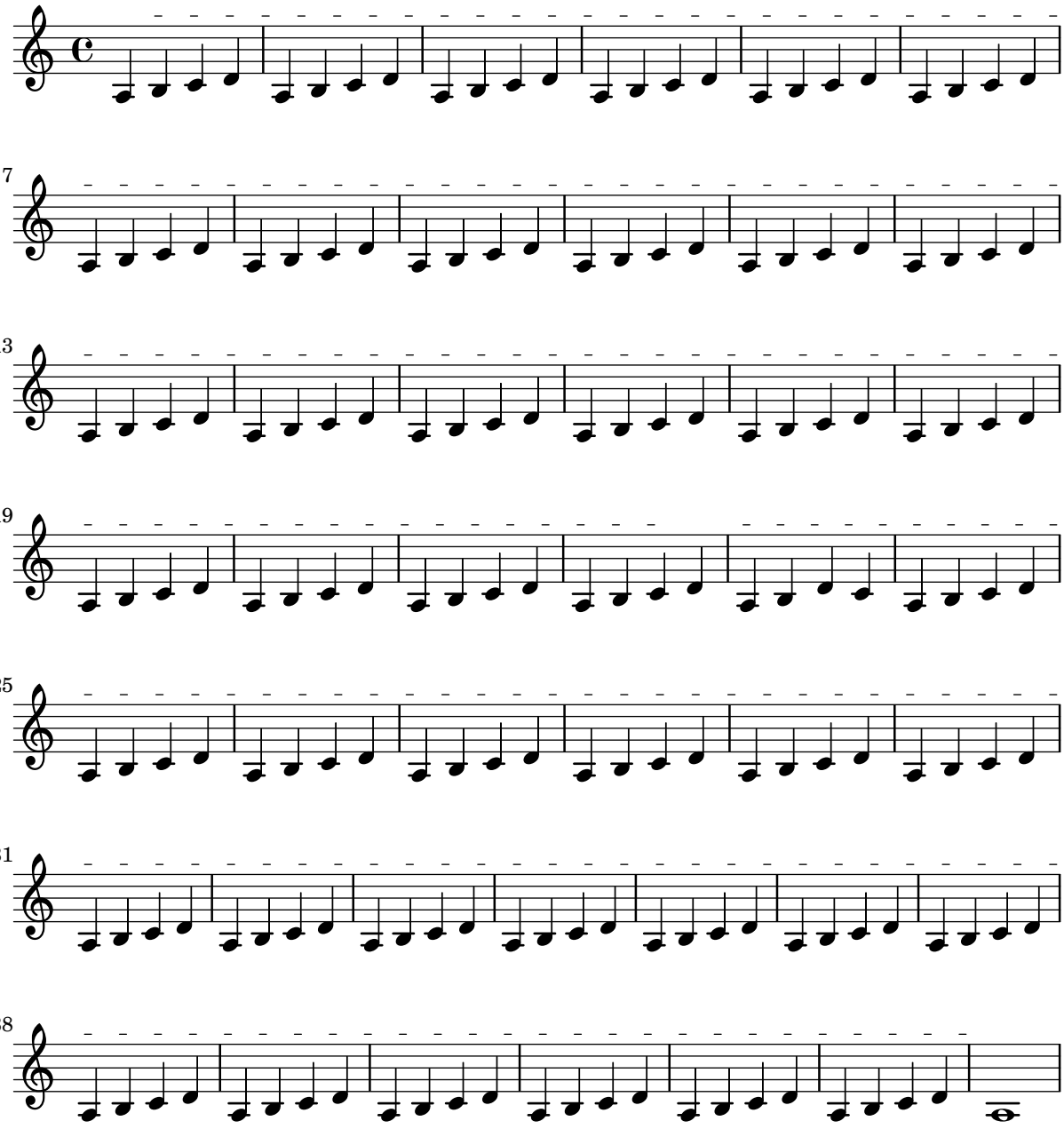


\consists can take a scheme alist as arguments, which should be functions, which will be invoked as engraver functions.

`scheme-engraver.ly`



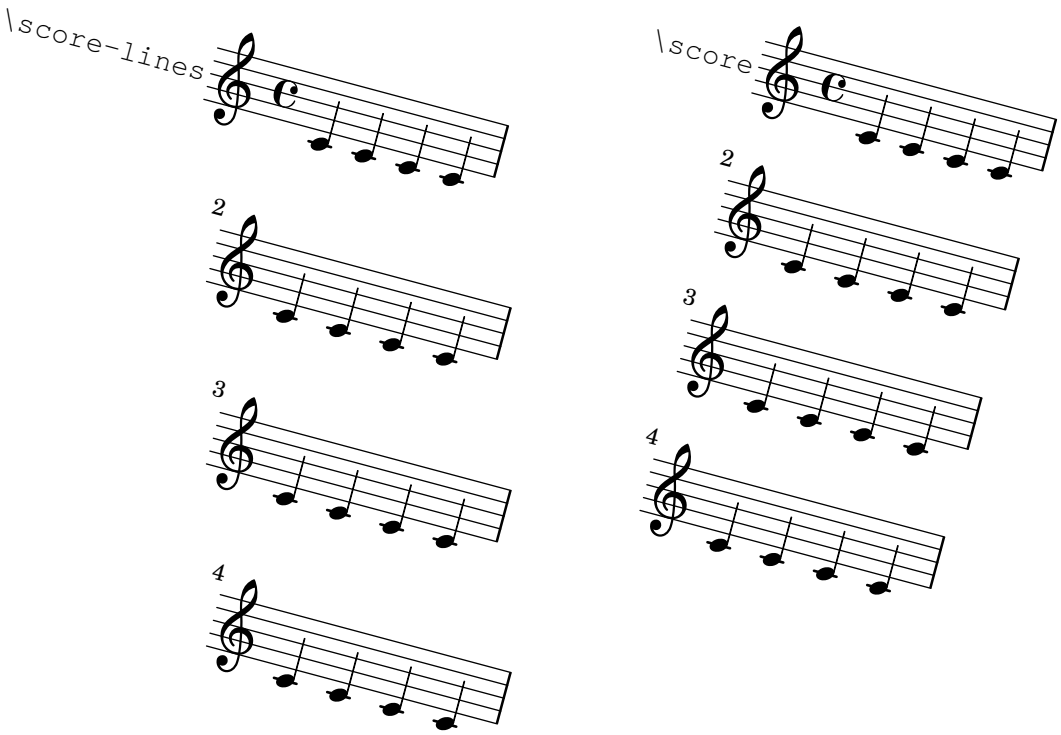
Use `define-event-class`, scheme engraver methods, and grob creation methods to create a fully functional text spanner in scheme.

`scheme-text-spanner.ly`



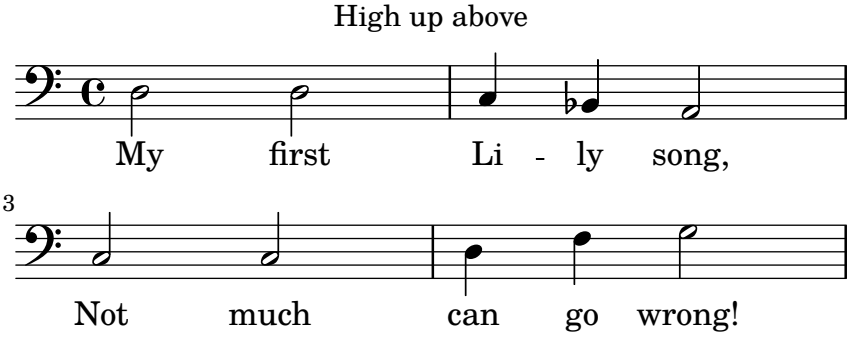
The `\score-lines` markup returns individual score lines as stencils rather than a single stencil. Calling a function like `\rotate` on `\score-lines` rotates the lines individually, as contrasted with rotating an entire `\score` markup.

score-lines.ly



Markup texts are rendered above or below a score.

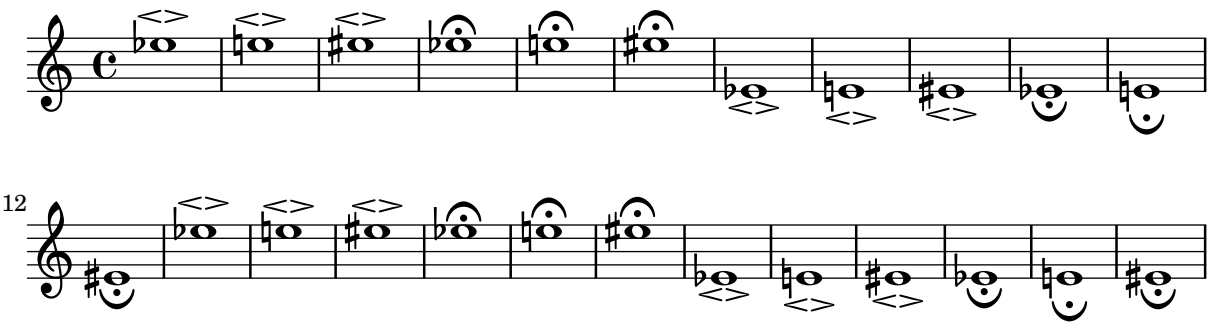
score-text.ly



2. My next Li-ly verse
   Now it's getting worse!
3. My last Li-ly text
   See what will be next!

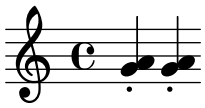
Scripts use skylines with accurate boxes to avoid accidentals.

`script-accidental-collision.ly`
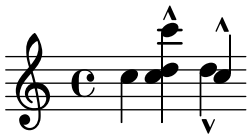


Scripts on chords with seconds remain centered on the extremal note head

`script-center-seconds.ly`
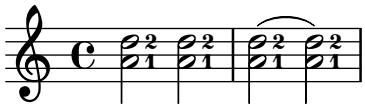


Scripts are put on the utmost head, so they are positioned correctly when there are collisions.

`script-collision.ly`
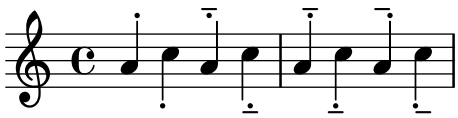


Horizontal scripts don't have `avoid-slur` set.

`script-horizontal-slur.ly`



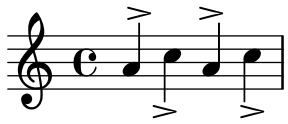
The horizontal placement of staccato dots above an upstem or below a downstem note differs from the placement of other scripts in that different positioning is used when the dot is alone and when it is part of a compound articulation. The property `toward-stem-shift-in-column` ensures good default positioning of the staccato (see first measure below), and allows precise horizontal control of a column containing a staccato and of the staccato within it (second measure). (`0.0` means centered on the note head, `1.0` means centered on the stem.)

`script-shift-staccato.ly`



The `toward-stem-shift` property controls the precise horizontal location of scripts that are placed above an upstem or below a downstem note (`0.0` means centered on the note head, `1.0` means centered on the stem).

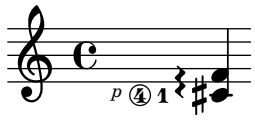
`script-shift.ly`



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

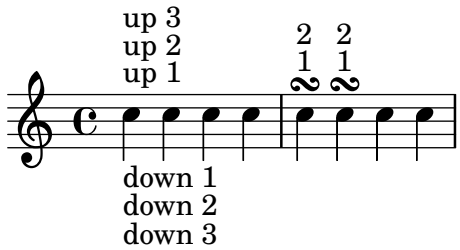
The scripts should not be folded under the time signature.

`script-stack-horizontal.ly`
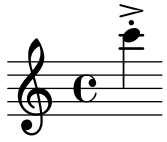


Scripts can be stacked. The order is determined by a priority field, but when objects have the same priority, the input order determines the order. Objects specified first are closest to the note.

`script-stack-order.ly`



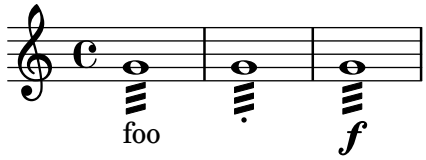
Scripts may be stacked.

`script-stacked.ly`



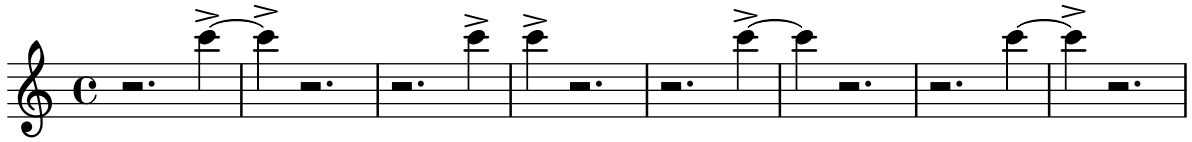
Scripts avoid stem tremolos even if there is no visible stem.

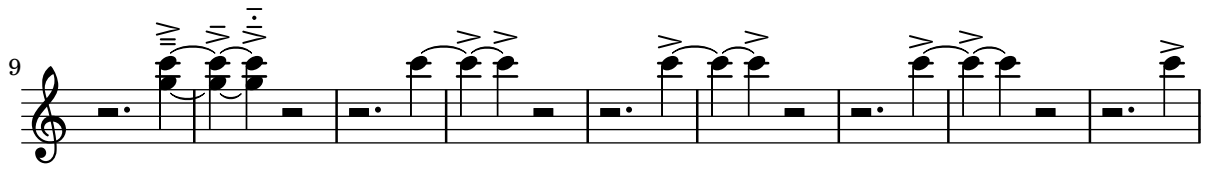
`script-stem-tremolo.ly`



Scripts avoid ties.

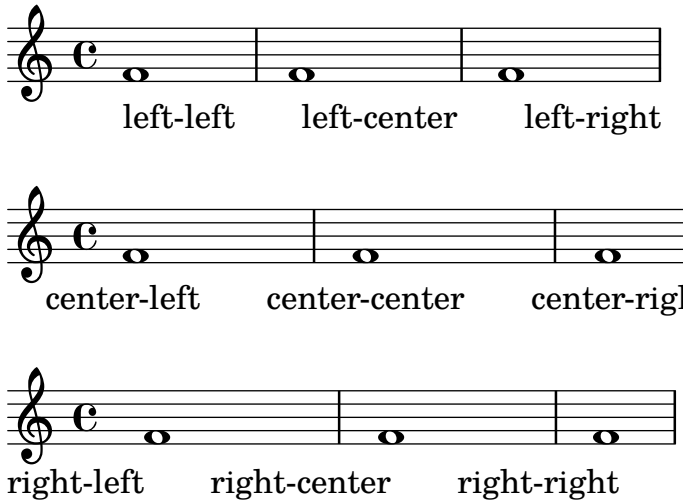
`script-tie-collision.ly`





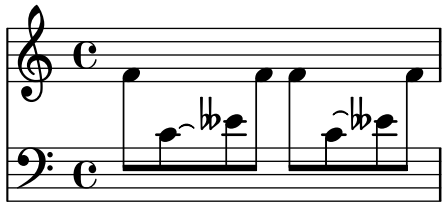
Grobs using `ly:self-alignment-interface::aligned-on-x-parent` and `ly:self-alignment-interface::aligned-on-y-parent` callbacks support separate alignments for self and parent.

`self-alignment-and-parent-alignment.ly`
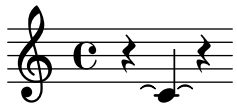


Cross-staff `RepeatTie` and `LaissezVibrerTie` do not trigger programming errors for circular dependencies in direction.

`semi-tie-cross-staff.ly`



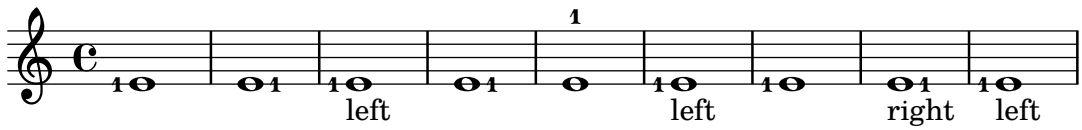
Semi tie directions may be forced from the input.

`semi-tie-manual-direction.ly`
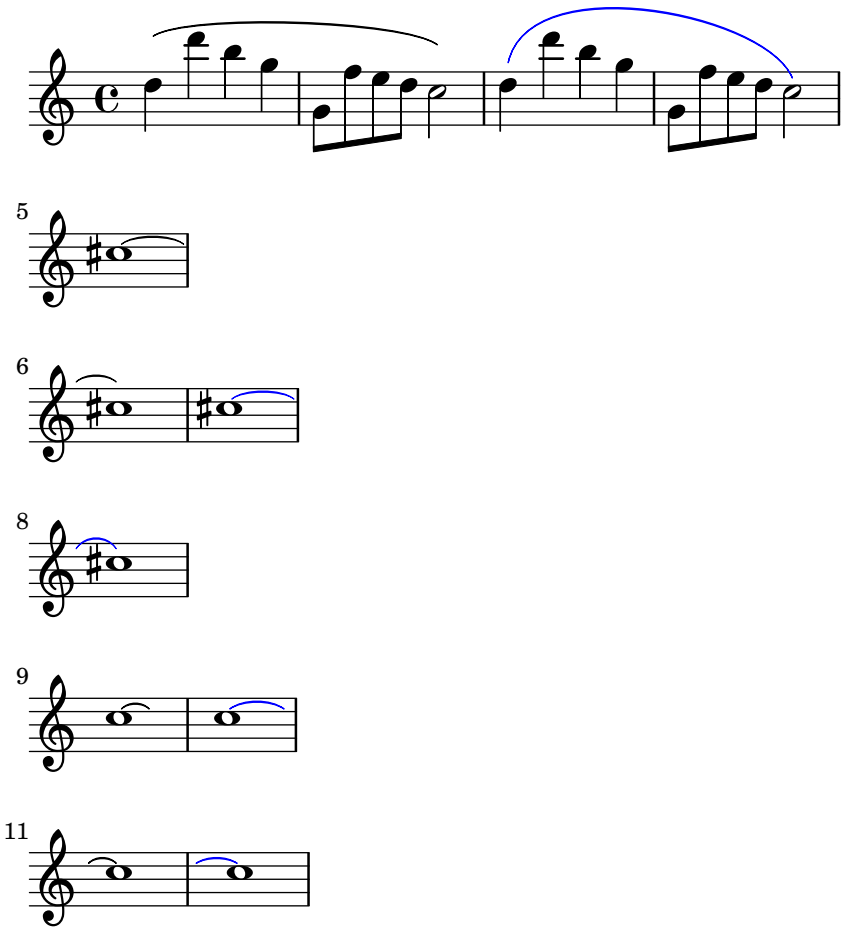


`\once \set` should change a context property value for just one timestep and then return to the previous value.

`set-once.ly`
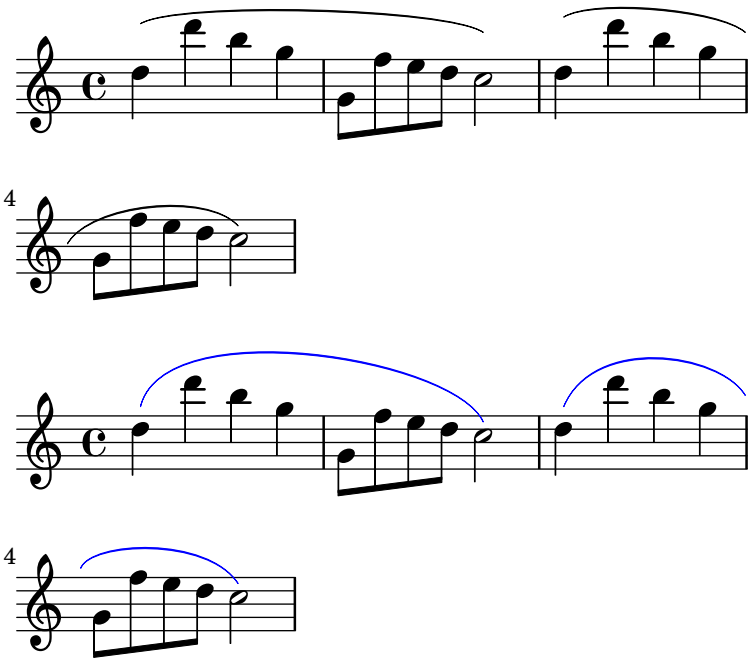


In addition to `Slur`, the music function `\shape` works with `PhrasingSlur`, `Tie`, `LaissezVibrerTie`, and `RepeatTie`. Each is shown below, first unmodified and then (in blue) after application of the function.

shape-other-curves.ly



The control points of a broken or unbroken slur may be offset by `\shape`. The blue slurs are modified from the default slurs shown first.

shape-slurs.ly



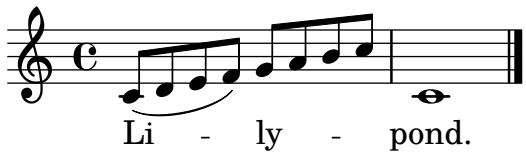
`\shiftDurations` can use negative dot values without causing a crash.

`shift-durations-negative-dots.ly`



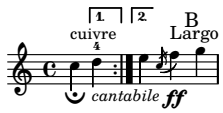
A number of shorthands like `(`, `)`, `|`, `[`, `]`, `~`, `\(`, `\)` and others can be redefined like normal commands. `ly/declarations-init.ly` serves as a regtest for a number of them. This test just demonstrates replacing `(` and `)` with melismata commands which are *not* articulations.

`shorthands.ly`



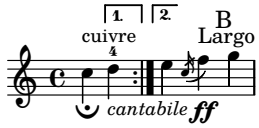
Different text styles are used for various purposes.

`size11.ly`



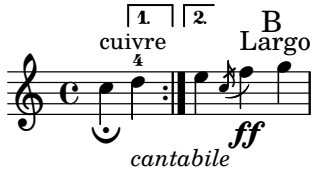
Different text styles are used for various purposes.

`size13.ly`



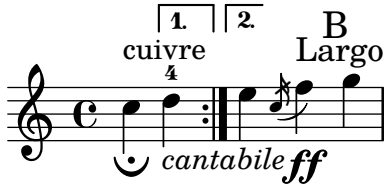
Different text styles are used for various purposes.

`size16.ly`



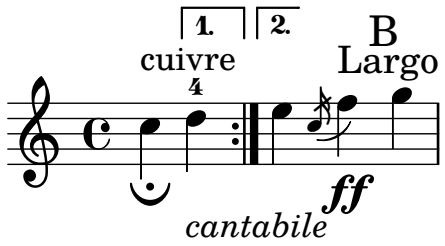
Different text styles are used for various purposes.

`size20.ly`



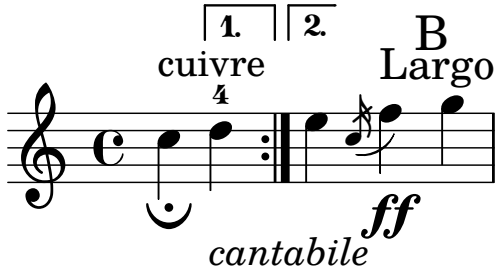
Different text styles are used for various purposes.

size23.ly



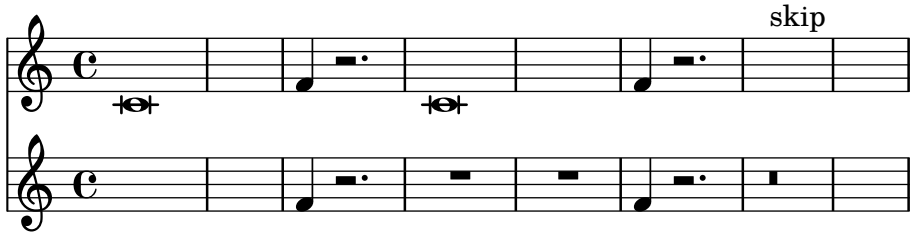
Different text styles are used for various purposes.

size26.ly



skip-of-length and mmrest-of-length create skips and rests that last as long as their arguments.

skip-of-length.ly
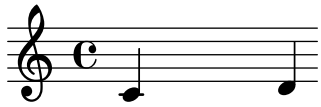


A score with skipTypesetting set for the whole score will not segfault.

skiptypesetting-all-true.ly

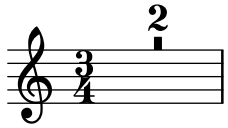
skipTypesetting doesn't affect bar checks.

skiptypesetting-bar-check.ly



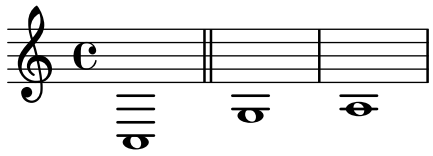
When skipTypesetting is set during a skipBars-induced MultiMeasureRest spanner, no segfault occurs.

skiptypesetting-multimeasurerest.ly



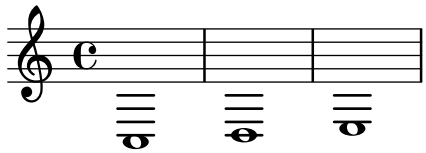
`showFirstLength` and `showLastLength` may be set at the same time; both the beginning and the end of the score will be printed.

`skiptypesetting-show-first-and-last.ly`



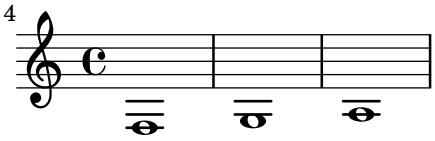
`showFirstLength` will only show the first bit of a score

`skiptypesetting-show-first.ly`
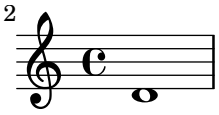


`showLastLength` will only show the last bit of a score
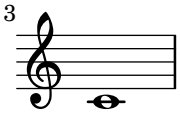
`skiptypesetting-show-last.ly`



Tuplet brackets are also skipped with `skipTypesetting`.

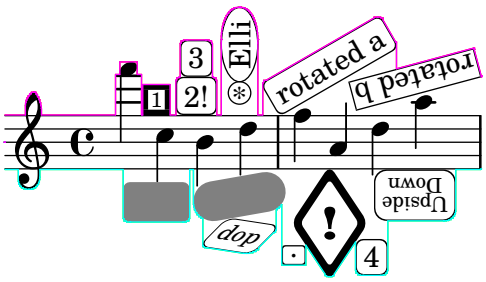
`skiptypesetting-tuplet.ly`





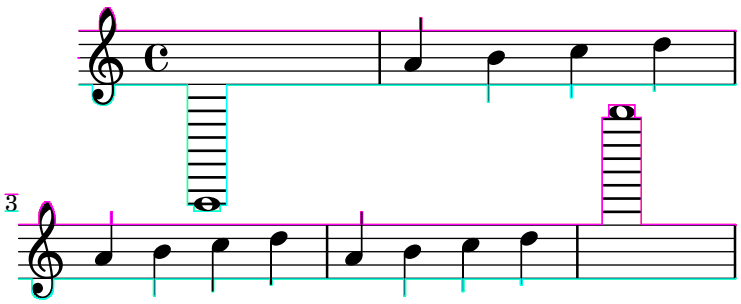
Skylines of boxes with and without rounded corners reflect the actual box outline even if rotated. Skylines of ellipses are stable when rotated.

skyline-boxes-ellipses.ly



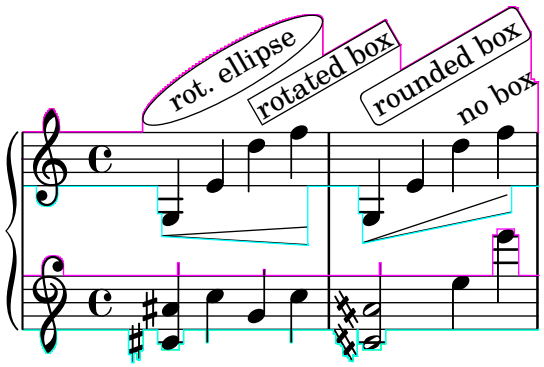
`-ddebug-skyline` draws the outline of the skyline used.

`skyline-debug.ly`



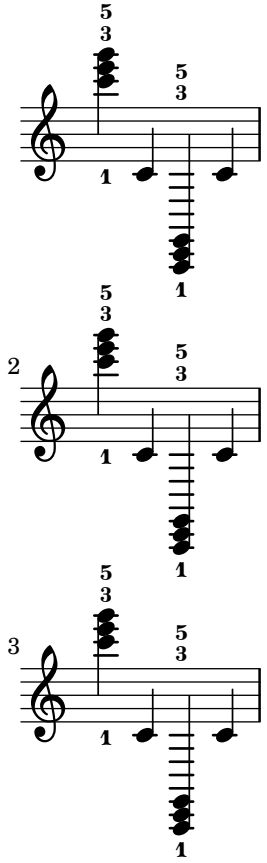
Skylines reflect grob rotation.

skyline-grob-rotation.ly
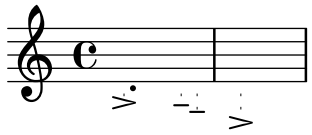


The skyline-horizontal-padding property can be set for System in order to keep systems from being spaced too closely together. In this example, the low notes from a system should not be interleaved with the high notes from the next system.

skyline-horizontal-padding.ly



The `Script` grobs should follow the descending melody line, even though the `NoteHead` stencils are point stencils. The `Stem_engraver` is removed so that the only `side-support-element` is the `NoteHead`.
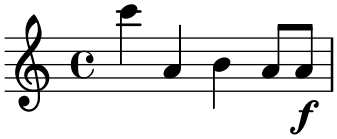
`skyline-point-extent.ly`



Grobs that have outside-staff-priority set are positioned using a skyline algorithm so that they don't collide with other objects.

`skyline-vertical-placement.ly`

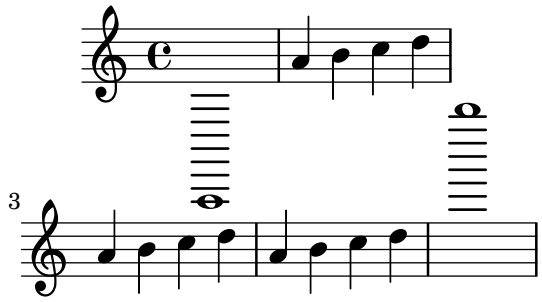
this goes above the previous markup
this doesn't collide with the c



this goes below the dynamic

We use a skyline algorithm to determine the distance to the next system instead of relying only on bounding boxes. This keeps gaps between systems more uniform.

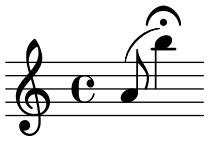
skyline-vertical-spacing.ly



Music engraving by LilyPond 2.21.1—www.lilypond.org

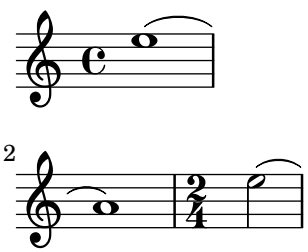
Slurs handle avoid objects better.

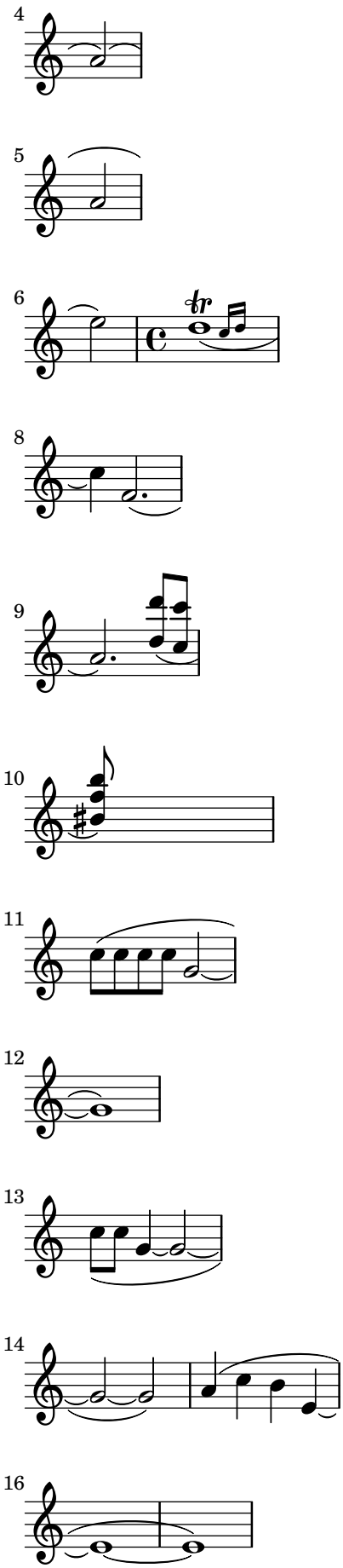
slur-avoid.ly



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.

slur-broken-trend.ly





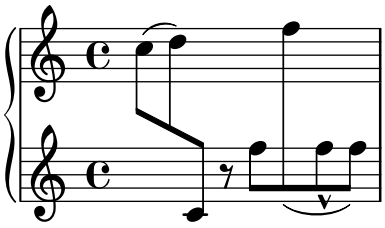
Slurs avoid clefs, but don’t avoid barlines.

`slur-clef.ly`



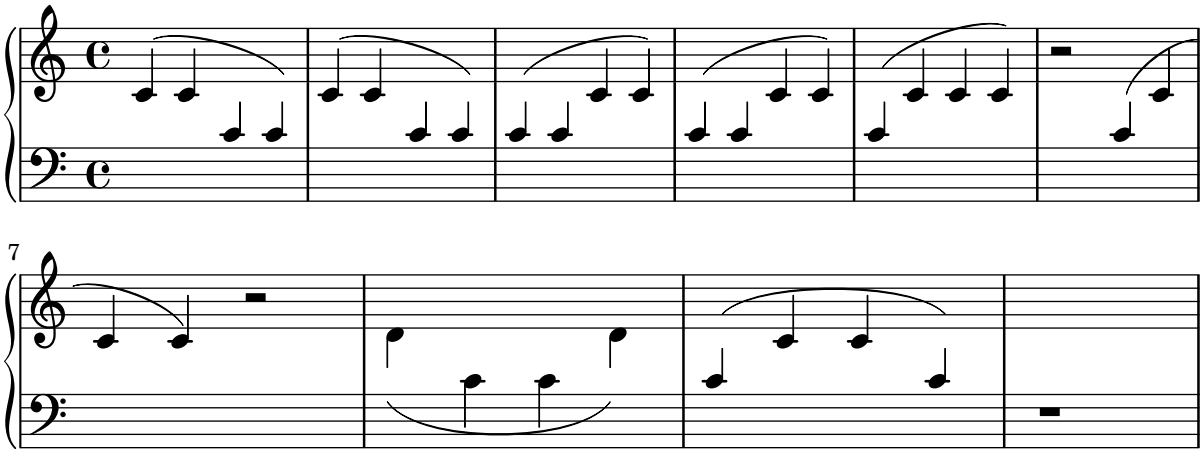
Slurs that depend on a cross-staff beam are not calculated until after line-breaking, and after inside-going articulations have been placed.

`slur-cross-staff-beam.ly`



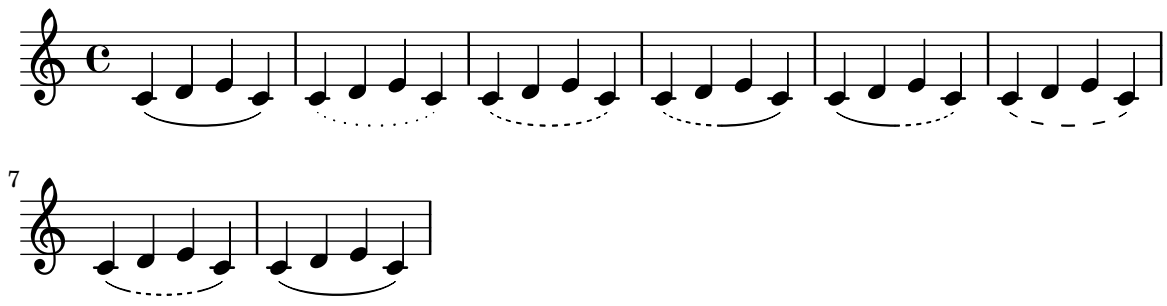
Slurs behave decently when broken across a linebreak.

`slur-cross-staff.ly`
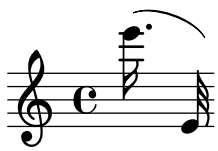


The appearance of slurs may be changed from solid to dotted or dashed.

`slur-dash.ly`



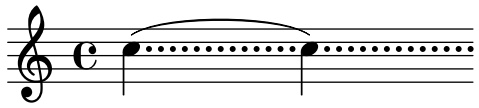
Slurs avoid dots.

`slur-dot-collision.ly`



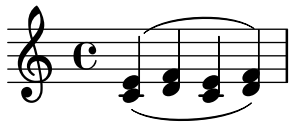
Slurs should not get confused by augmentation dots. With a lot of dots, the problems becomes more visible.

`slur-dots.ly`



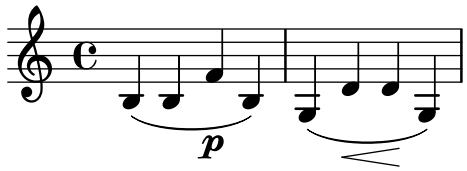
Some composers use slurs both above and below chords. This can be typeset by setting `doubleSlurs`

`slur-double.ly`



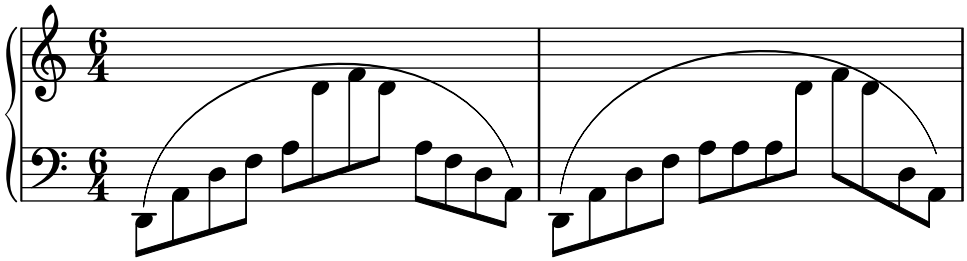
Dynamics avoid collision with slur.

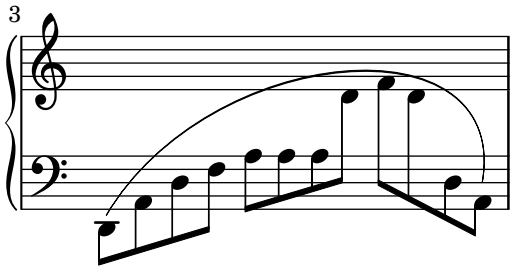
`slur-dynamics.ly`



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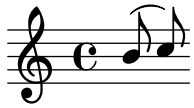
`slur-extreme.ly`
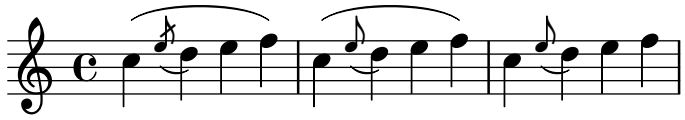




Slurs take flag extents into account.

`slur-flag.ly`
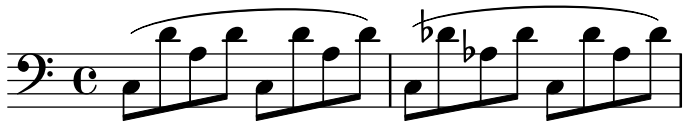


Appoggiatura and acciaccaturas use a different slur than the default, so they produce a nested slur without warnings.

`slur-grace.ly`



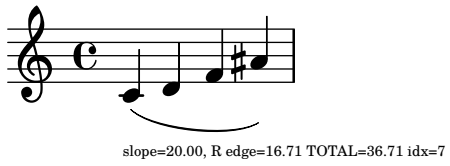
Slur shaping is not adapted to accommodate objects towards the edges of slur. Said objects are thus ignored, which should make the slur in this regtest flat. Objects towards the edges are not, however, ignored in the slur scoring.

`slur-height-capping.ly`



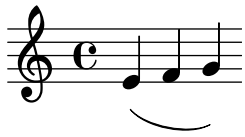
Specifying `inspect-quants`, will print out demerit scores for the given configuration. Here, there are demerits for slur slope going to melody slope, and the slur ending far from the right edge.

`slur-inspect-quants.ly`
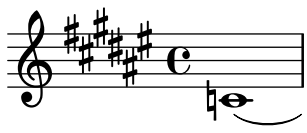


Setting `positions` overrides the automatic positioning of the slur. It selects the slur configuration closest to the given pair.

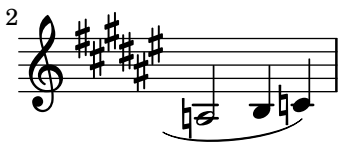
`slur-manual.ly`
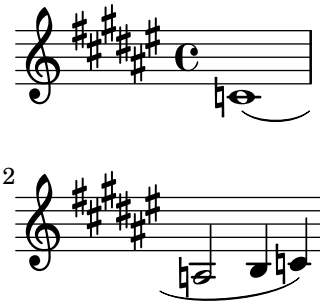


An additional opening slur during a running slur should be ignored (and a warning printed), but never influence the slur's extents.

`slur-multiple-linebreak.ly`
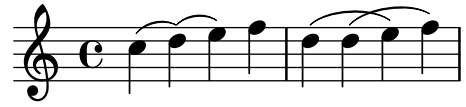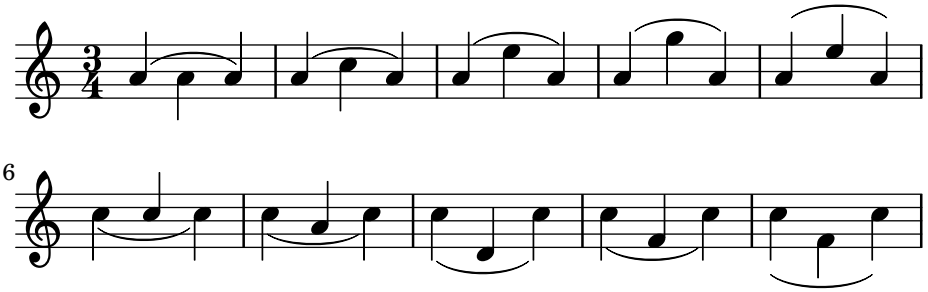






LilyPond does not support multiple concurrent slurs with the parentheses syntax. In this case, warnings will be given and the nested slur will not be generated. However, one can can create a second slur with a different spanner-id.

`slur-multiple.ly`
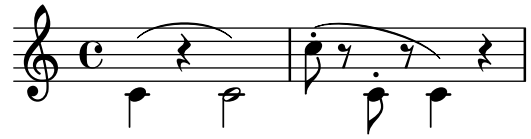


Slurs should look nice and symmetric. The curvature may increase only to avoid noteheads, and as little as possible. Slurs never run through noteheads or stems.

`slur-nice.ly`



Slurs may be placed over rests. The slur will avoid colliding with the rests.
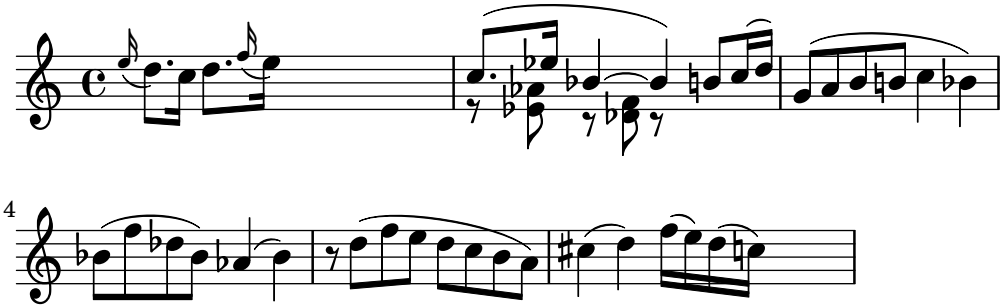
`slur-rest.ly`
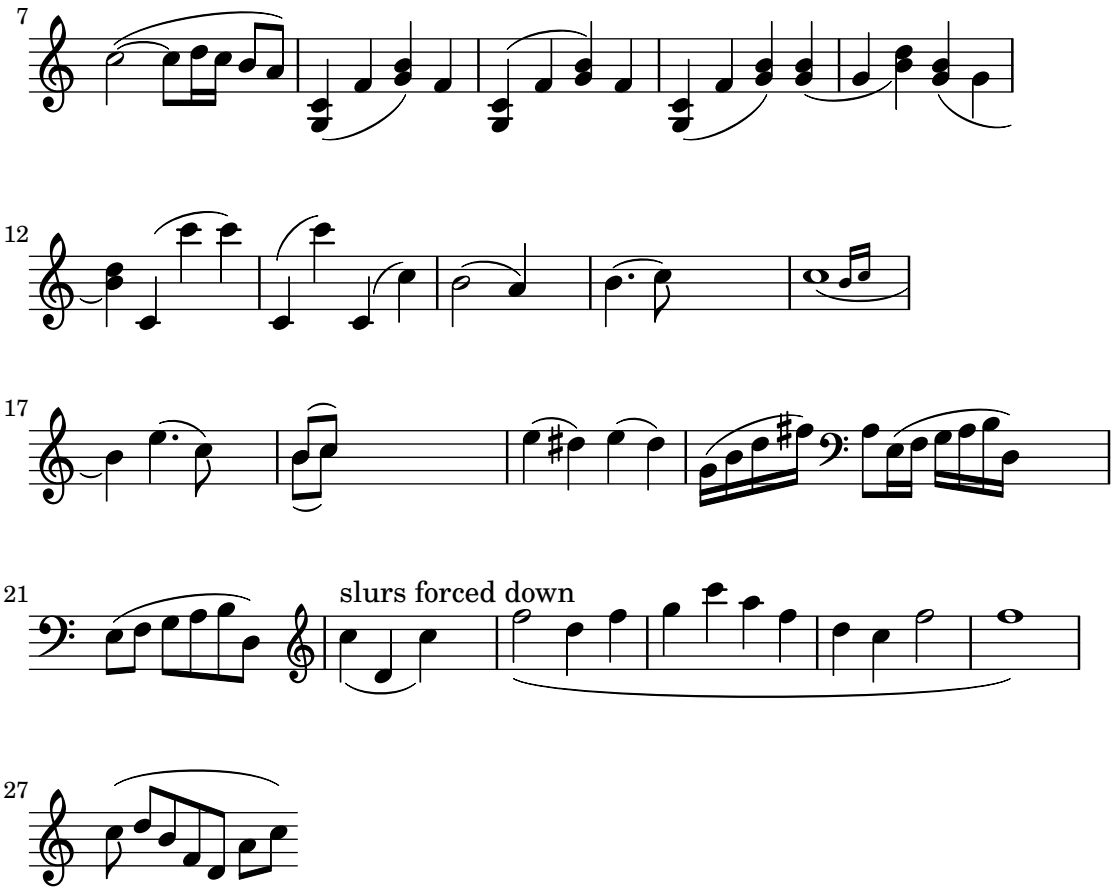


Slur formatting is based on scoring. A large number of slurs are generated. Each esthetic aspect gets demerits, the best configuration (with least demerits) wins. This must be tested in one big file, since changing one score parameter for one situation may affect several other situations.

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

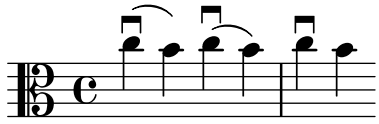
`slur-scoring.ly`





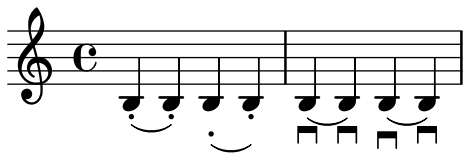
Slurs avoid scripts with `avoid-slur` set to `inside`, scripts avoid slurs with `avoid-slur` set to `around`. Slurs and scripts keep a distance of `slur-padding`.

`slur-script-inside.ly`
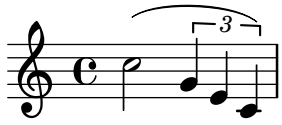


A slur avoids collisions with scripts, which are placed either inside or outside the slur, depending on the script. The slur responds appropriately if a script is moved.

`slur-script.ly`



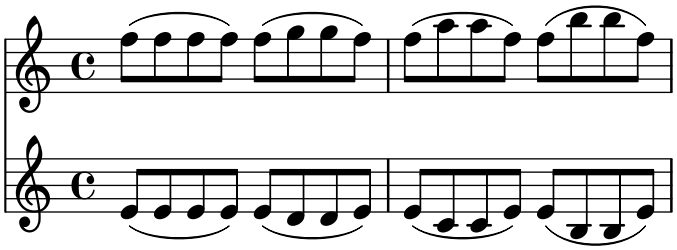
A slur's shift region is automatically made higher to accommodate extra encompass elements.

`slur-shift-region.ly`



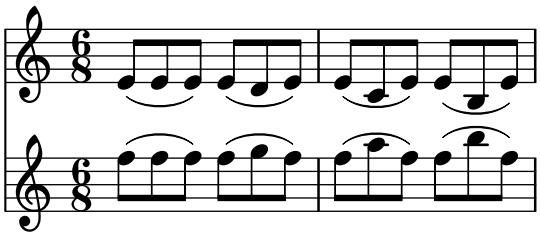
Symmetric figures should lead to symmetric slurs.

`slur-symmetry-1.ly`
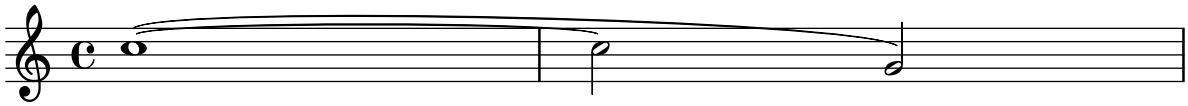


Symmetric figures should lead to symmetric slurs.

`slur-symmetry.ly`
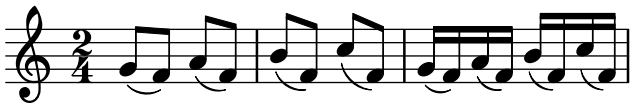


Slurs and ties should never share extremal control points.

`slur-tie-control-points.ly`
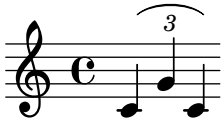


The attachment point for strongly sloped slurs is shifted horizontally slightly. Without this correction, slurs will point into one note head, and point over another note head.

`slur-tilt.ly`



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

`slur-tuplet.ly`
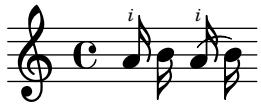


Slurs do not force grobs with outside-staff-priority too high.

`slur-vertical-skylines.ly`
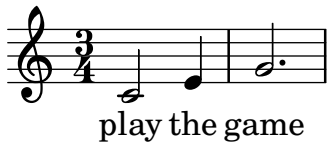


Outside staff callbacks that no longer apply to grobs because they are outside the X boundary of a slur should terminate early. The example below should generate no warnings about Bezier curves and there should be no change in StrokeFinger position between the first and second examples.

`slur-vestigial-outside-staff-callback.ly`



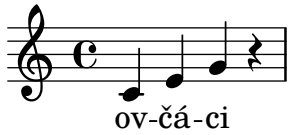
Festival song synthesis output supports associated voices.

`song-associated-voice.ly`



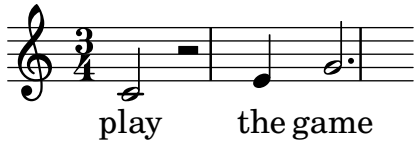
Festival song synthesis output supports non-english syllabels.

`song-basic-nonenglish.ly`
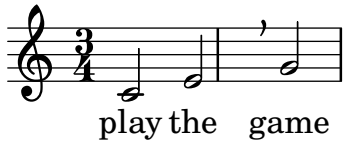


Festival song synthesis output supports basic songs.

`song-basic.ly`
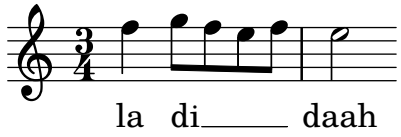


Festival song synthesis output supports breath marks.

`song-breathe.ly`



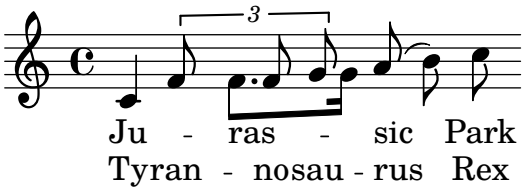
Festival song synthesis output supports melismas.

`song-melisma.ly`



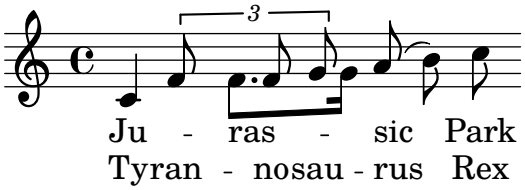
Festival song synthesis output supports reordered lyrics.

`song-reordering.ly`
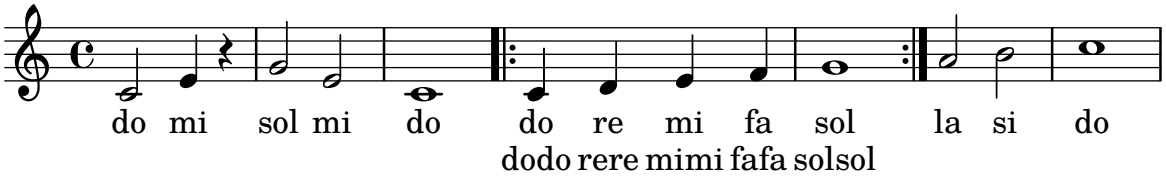


Festival song synthesis output supports reordered lyrics.

`song-reordering2.ly`



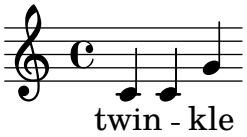
Festival song synthesis output supports repeat signs.

`song-repetition.ly`



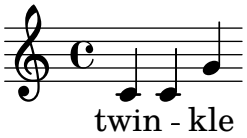
Festival song synthesis output supports lyrics which are not complete words.

`song-skip-noword.ly`



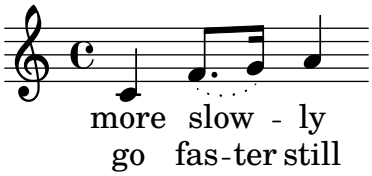
Festival song synthesis output supports skips.

`song-skip.ly`
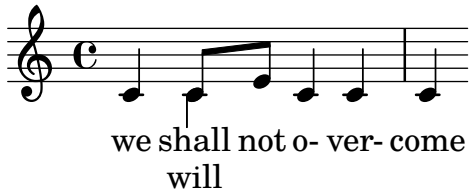


Festival song synthesis output supports slurs.

`song-slurs.ly`
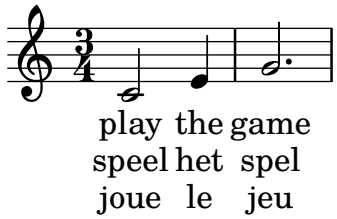


Festival song synthesis output supports divided voices.

song-splitpart.ly
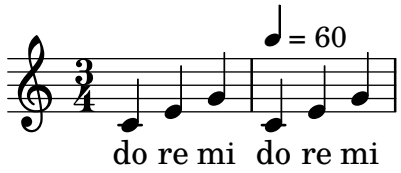


Festival song synthesis output supports multiple stanzas.

song-stanzas.ly
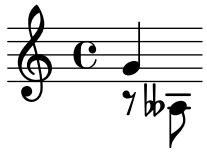


Festival song synthesis output supports changing tempo in the middle of a piece.

song-tempo.ly
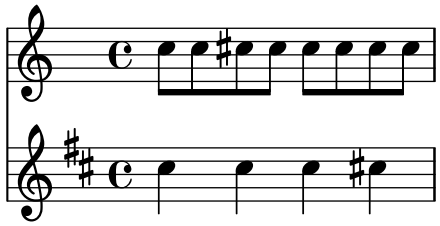


Accidentals don't collide with shifted-down rests.

spacing-accidental-rest.ly
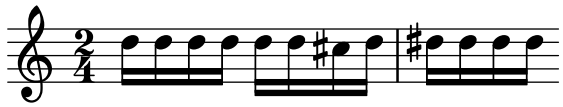


Accidentals in different staves do not affect the spacing of the eighth notes here.

spacing-accidental-staffs.ly
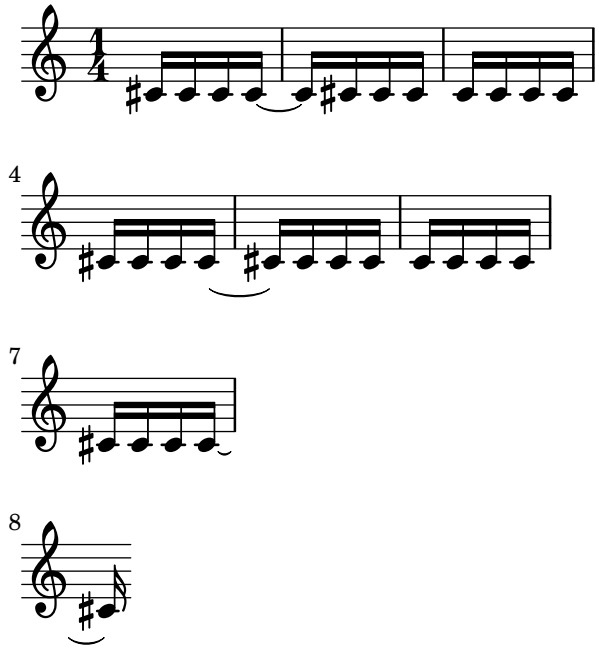


Accidentals do not influence the amount of stretchable space. The accidental does add a little non-stretchable space.

spacing-accidental-stretch.ly
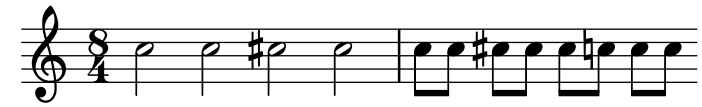


Horizontal spacing works as expected on tied notes with accidentals. No space is reserved for accidentals that end up not being printed, but accidentals that are printed don't collide with anything.

`spacing-accidental-tie.ly`



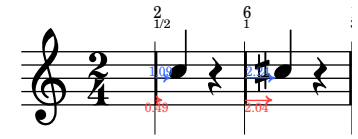
Accidentals sticking out to the left of a note will take a little more space, but only if the spacing is tight.

`spacing-accidental.ly`



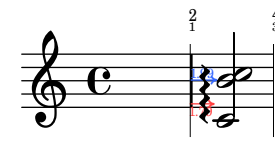
An accidental following a bar gets space so the left edge of the acc is at 0.3 staff space from the bar line

`spacing-bar-accidental.ly`



An arpeggio following a bar gets space

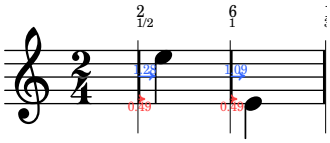
`spacing-bar-arpeggio.ly`



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

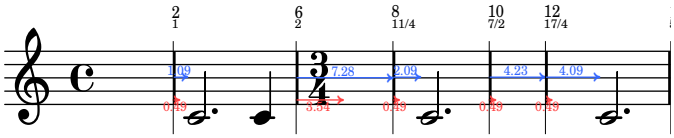
The bar upstem should be approx 1.1 staff space, the bar downstem 1.3 to 1.5 staff space.

`spacing-bar-stem.ly`



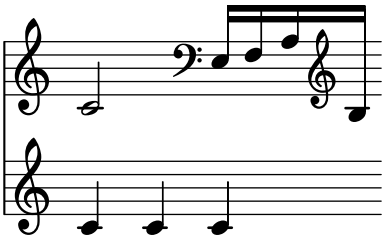
Notes that fill a whole measure are preceded by extra space.

`spacing-bar-whole-measure.ly`
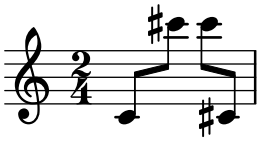


Clef changes at the start of a line get much more space than clef changes halfway the line.

`spacing-clef-first-note.ly`



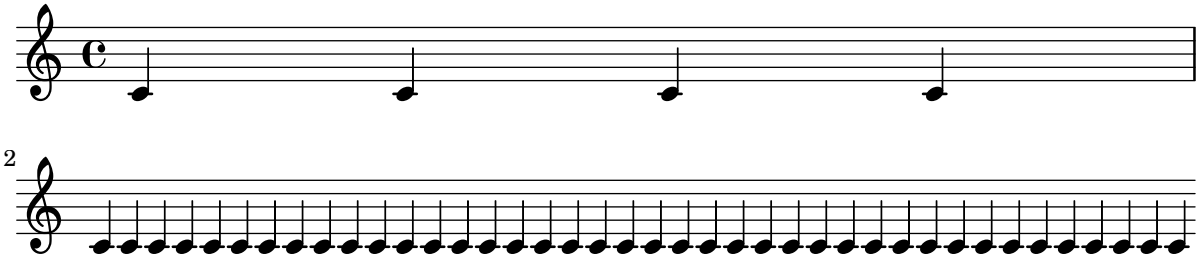
If right hand stems have accidentals, optical spacing correction is still applied, but only if the stem directions are different.

`spacing-correction-accidentals.ly`
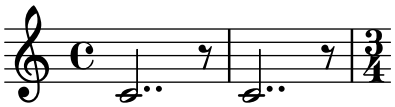


Empty barlines do not affect spacing.
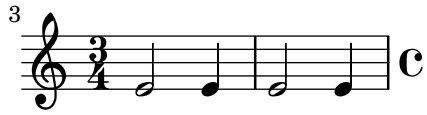
`spacing-empty-bar.ly`



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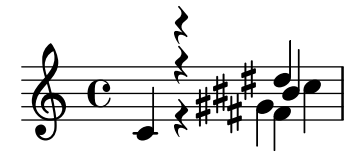
`spacing-end-of-line.ly`
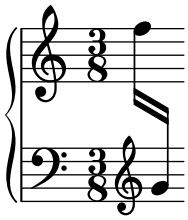




A voicelet (a very short voice to get polyphonic chords correct) should not confuse the spacing engine.

`spacing-ended-voice.ly`
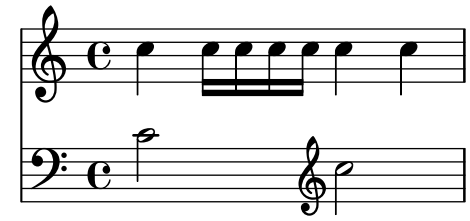


Clefs are also folded under cross staff constructs.

`spacing-folded-clef-cross-staff.ly`



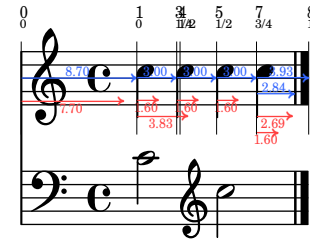
A clef can be folded below notes in a different staff, if this does not disrupt the flow of the notes.

`spacing-folded-clef.ly`
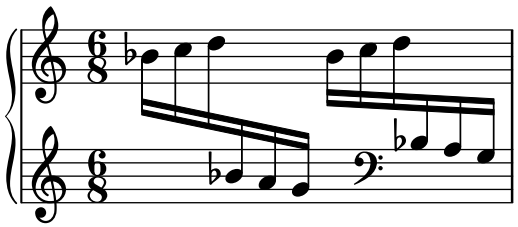


A clef can be folded below notes in a different staff, if there is space enough. With `Paper_column` stencil callbacks we can show where columns are in the score.

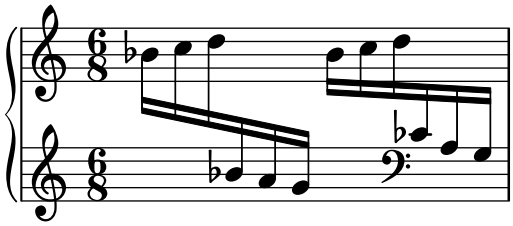
`spacing-folded-clef2.ly`



Voices that go back and forth between staves do not confuse the spacing engine.
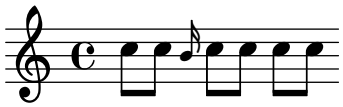
`spacing-folded-clef3.ly`
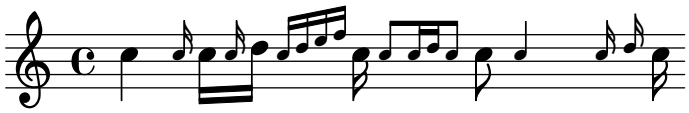




Spacing uses the duration of the notes, but disregards grace notes for this. In this example, the 8ths around the grace are spaced exactly as the other 8th notes.

`spacing-grace-duration.ly`



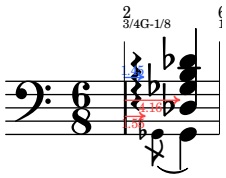
Grace note runs have their own spacing variables in `Score.GraceSpacing`. So differing grace note lengths inside a run are spaced accordingly.

`spacing-grace.ly`
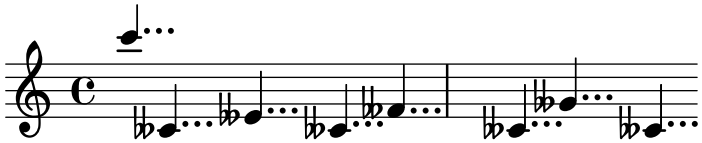


Skyline horizontal spacing may fold non-adjacent columns together, but they still do not collide. In this case, the arpeggio and the barline do not collide.

`spacing-horizontal-skyline-grace.ly`



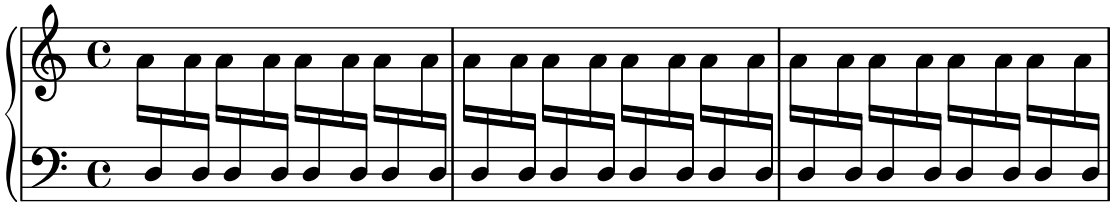
accidentals may be folded under preceding notes.

`spacing-horizontal-skyline.ly`
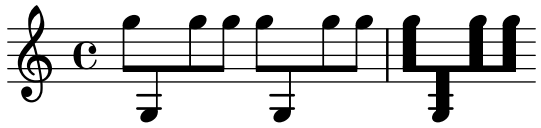


Spacing corrections for kneed beams still work when compression is involved.

`spacing-knee-compressed.ly`
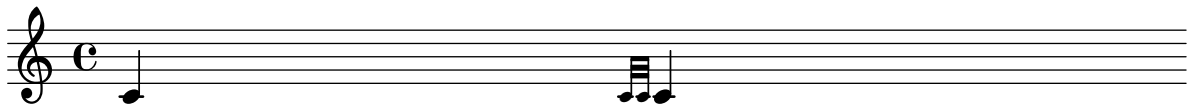


For knees, the spacing correction is such that the stems are put at regular distances. This effect takes into account the width of the note heads and the thickness of the stem.

spacing-knee.ly



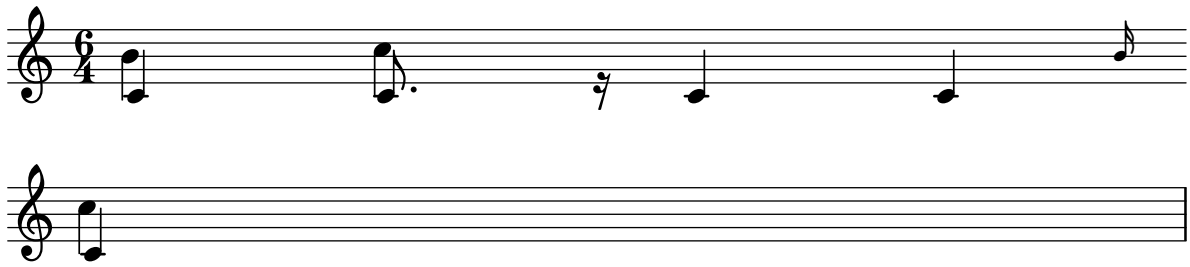
Even in case of incorrect contexts (eg. shortlived contexts) that break linking of columns through spacing wishes, `strict-note-spacing` defaults to a robust solution. This test passes if it does not seg fault; instead it should produce three programming error messages. Note that, in tight music with strict note spacing, grace notes will collide with normal notes. This is expected.

spacing-loose-grace-error.ly



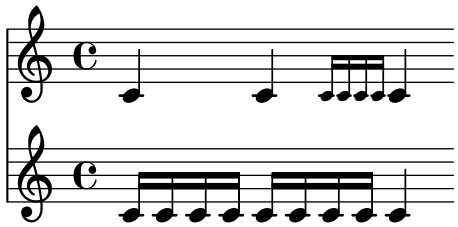
If a floating grace spacing section attaches to a note across a line break, it gets attached to the end of line.

spacing-loose-grace-linebreak.ly



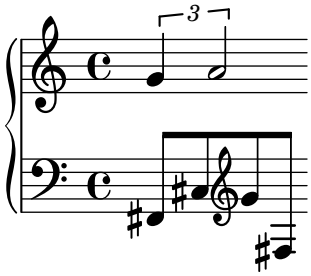
With `strict-grace-spacing`, grace notes don't influence spacing.

spacing-loose-grace.ly



Loose columns (here, the treble clef) are spaced correctly in polyphonic music.
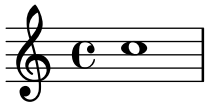
spacing-loose-polyphony.ly



Width of marks does not affect spacing.

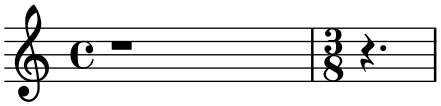
spacing-mark-width.ly

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx



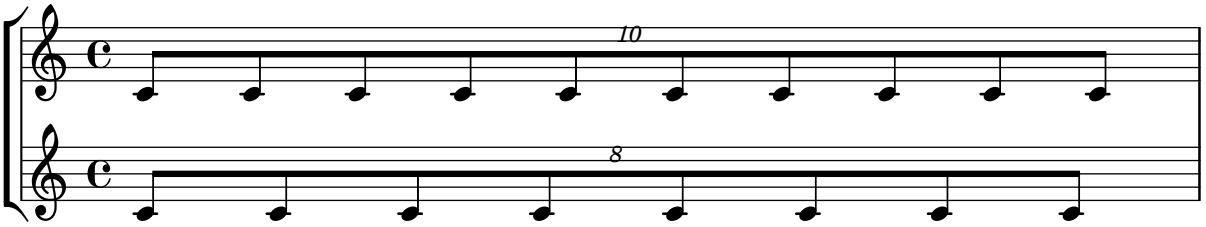
Horizontal spacing is bounded by the current measure length. This means that the 3/8 setting does not affect the whole rest spacing.

spacing-measure-length.ly



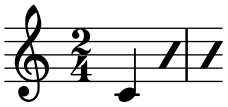
Concurrent tuplets should be equidistant on all staves.

spacing-multi-tuplet.ly
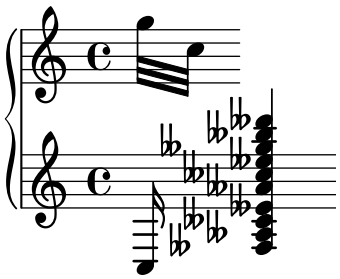


In the absence of NoteSpacings, wide objects still get extra space. In this case, the slash before the barline gets a little more space.

spacing-no-note.ly



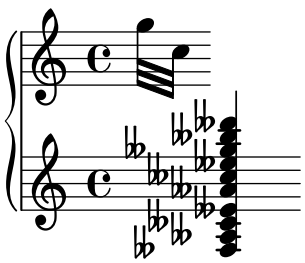
The spacing engine avoids collisions between non-adjacent columns.

spacing-non-adjacent-columns1.ly



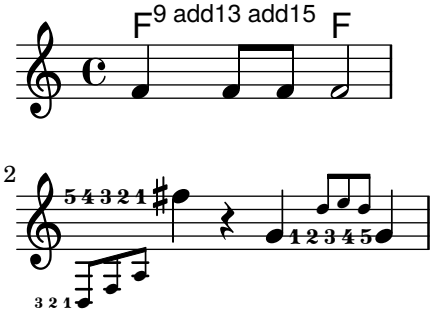
The spacing engine avoids collisions between non-adjacent columns.

spacing-non-adjacent-columns2.ly



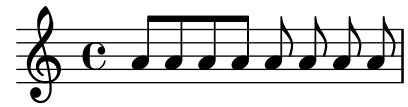
The spacing engine avoids collisions between non-adjacent columns.

`spacing-non-adjacent-columns3.ly`
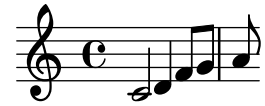


The flags of 8th notes take some space, but not too much: the space following a flag is less than the space following a beamed 8th head.

`spacing-note-flags.ly`



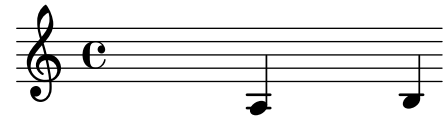
In packed mode, pack notes as tight as possible. This makes sense mostly in combination with ragged-right mode: the notes are then printed at minimum distance. This is mostly useful for ancient notation, but may also be useful for some flavours of contemporary music. If not in ragged-right mode, lily will pack as many bars of music as possible into a line, but the line will then be stretched to fill the whole linewidth.

`spacing-packed.ly`
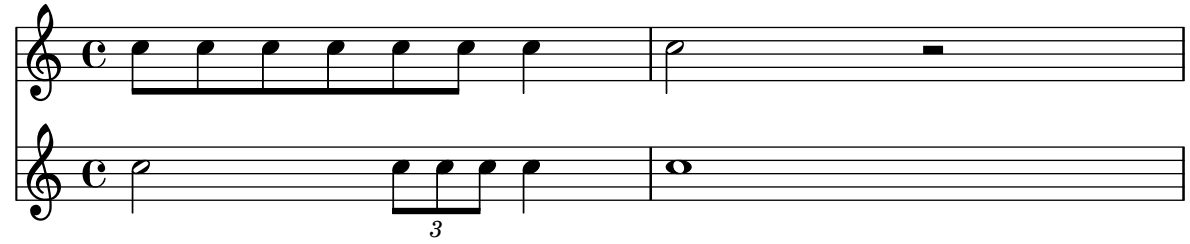


The space after a paper column can be increased by overriding the padding property.

`spacing-paper-column-padding.ly`



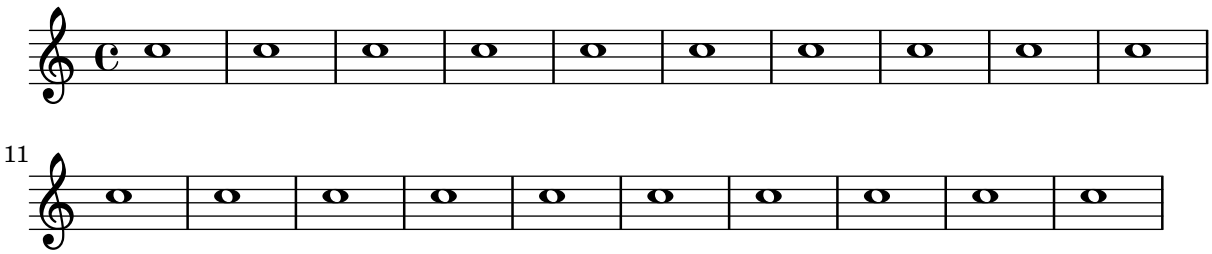
Proportional notation can be created by setting `proportionalNotationDuration`. Notes will be spaced proportional to the distance for the given duration.

`spacing-proportional.ly`
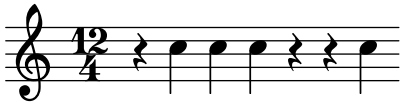


If `ragged-last` is set, the systems are broken similar to paragraph formatting in text: the last line is unjustified.

`spacing-ragged-last.ly`
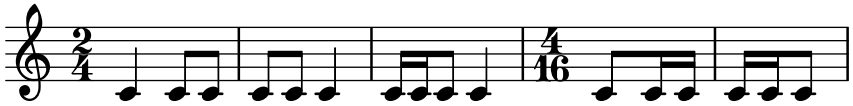


Rests get a little less space, since they are narrower. However, the quarter rest in feta font is relatively wide, causing this effect to be very small.

`spacing-rest.ly`
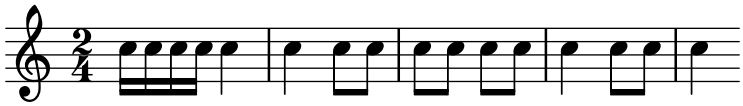


New sections for spacing can be started with `\newSpacingSection`. In this example, a section is started at the 4/16, and a 16th in the second section takes as much space as a 8th in first section.

`spacing-section.ly`
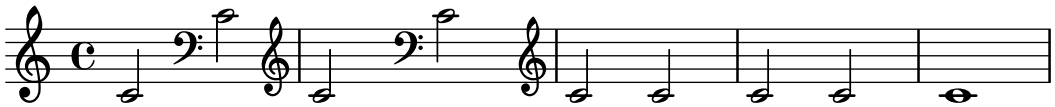


Notes that are shorter than the common shortest note get a space (i.e. without the space needed for the note) proportional to their duration. So, the 16th notes get 1/2 of the space of an eighth note. The total distance for a 16th (which includes note head) is 3/4 of the eighth note.

`spacing-short-notes.ly`



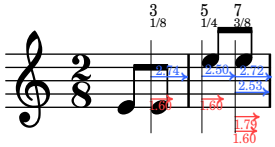
When space-to-barline is false, we measure the space between the note and the start of the clef. When space-to-barline is true, we measure the space between the note and the start of the barline.

`spacing-space-to-barline.ly`



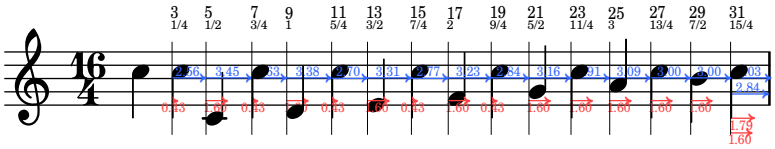
Upstem notes before a barline are printed with some extra space. This is an optical correction similar to juxtaposed stems.

`spacing-stem-bar.ly`



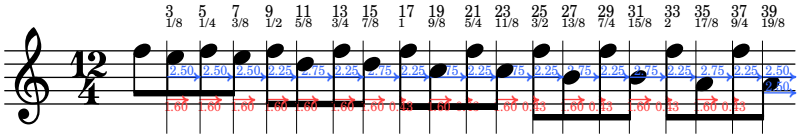
There are optical corrections to the spacing of stems. The overlap between two adjacent stems of different direction is used as a measure for how much to correct.

`spacing-stem-direction.ly`



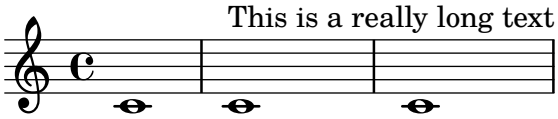
For juxtaposed chords with the same direction, a slight optical correction is used. It is constant, and works only if two chords have no common head-positions range.

`spacing-stem-same-direction.ly`



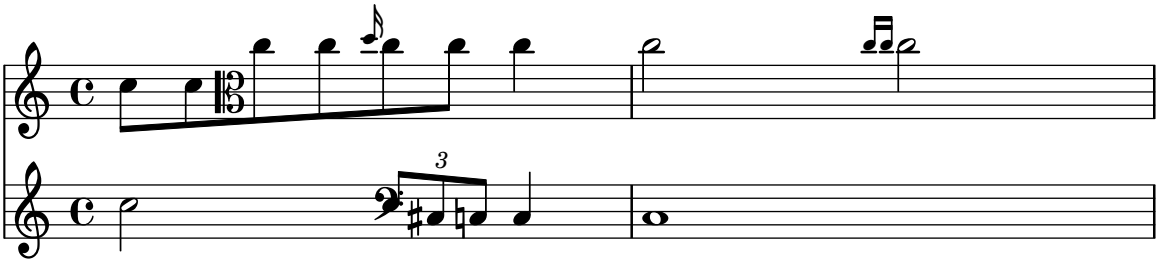
LilyPond will space a line to prevent text sticking out of the right margin unless `keep-inside-line` is false for the relevant PaperColumn.

`spacing-stick-out.ly`



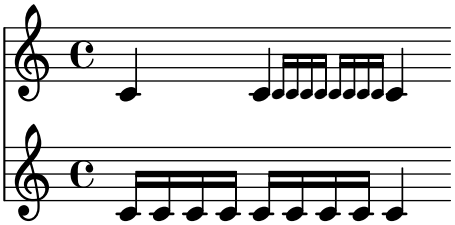
If `strict-note-spacing` is set, then spacing of notes is not influenced by bars and clefs half-way on the system. Rather, they are put just before the note that occurs at the same time. This may cause collisions.

`spacing-strict-notespacing.ly`
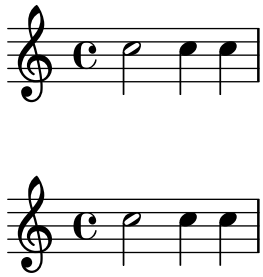


With `strict-note-spacing` spacing for grace notes (even multiple ones), is floating as well.

`spacing-strict-spacing-grace.ly`
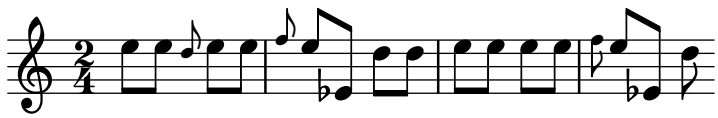


An empty barline does not confuse the spacing engine too much. The two scores should look approximately the same.

`spacing-to-empty-barline.ly`



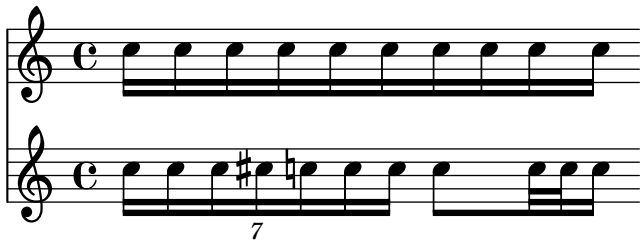
Space from a normal note (or barline) to a grace note is smaller than to a normal note.

`spacing-to-grace.ly`



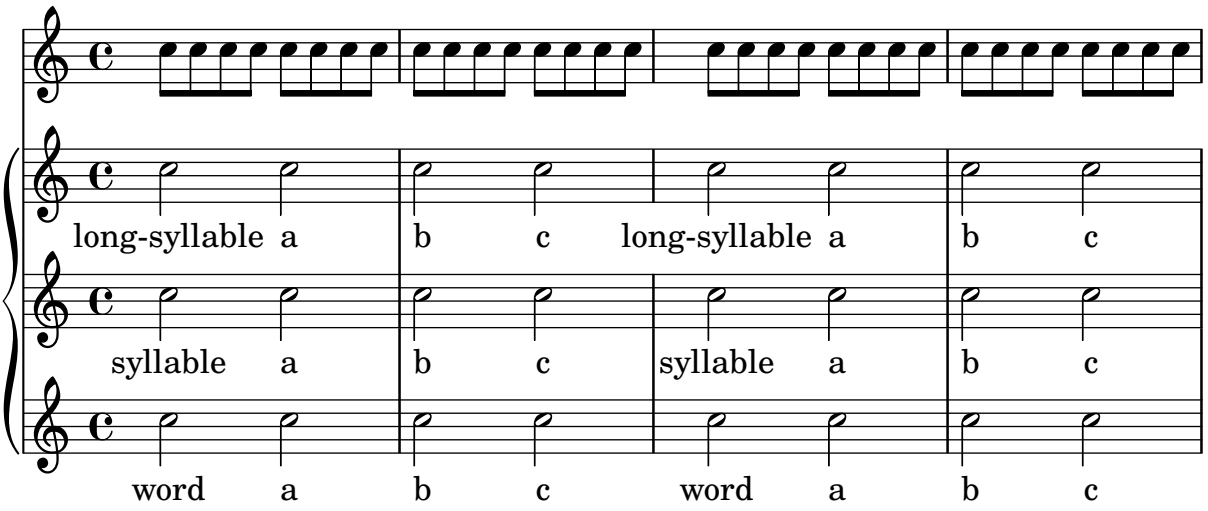
Notes are spaced exactly according to durations, if `uniform-stretching` is set. Accidentals are ignored, and no optical-stem spacing is performed.

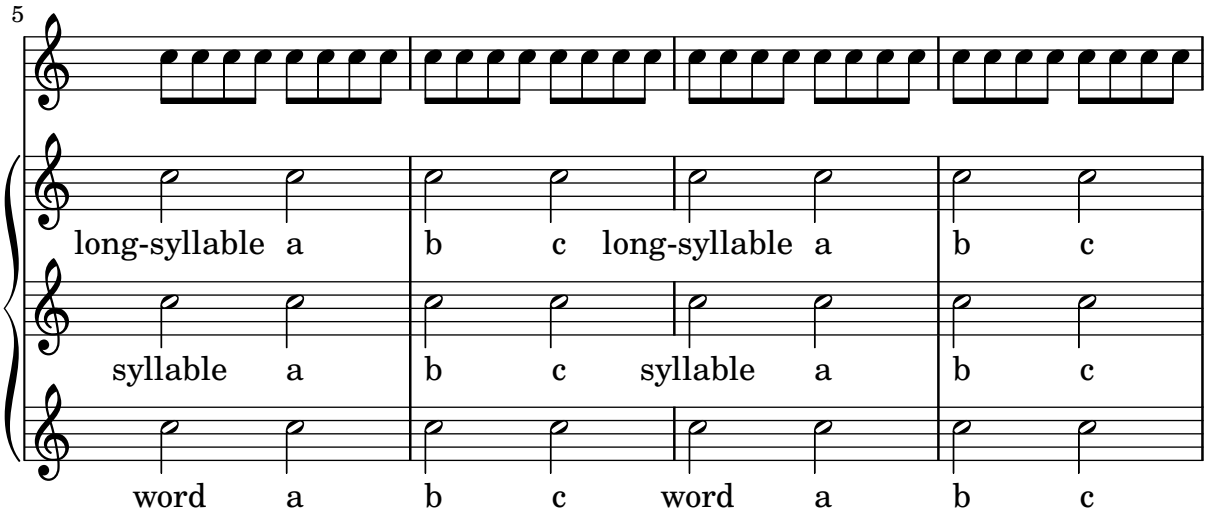
`spacing-uniform-stretching.ly`



The `SpanBarStub` grob takes care of horizontal spacing for `SpanBar` grobs. When the `SpanBar` is disallowed, objects in contexts that the span bar would have otherwise crossed align as if the span bar were not there.

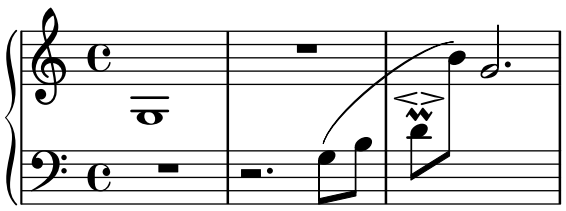
`span-bar-allow-span-bar.ly`





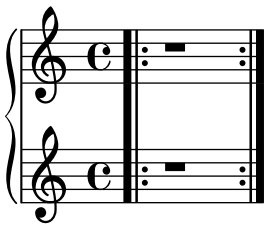
Articulations on cross-staff stems do not collide with span bars.

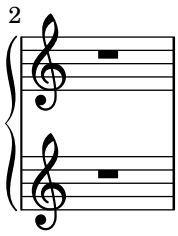
`span-bar-articulation.ly`



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

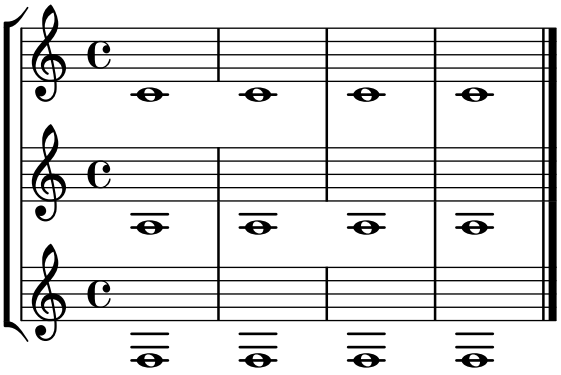
`span-bar-break.ly`





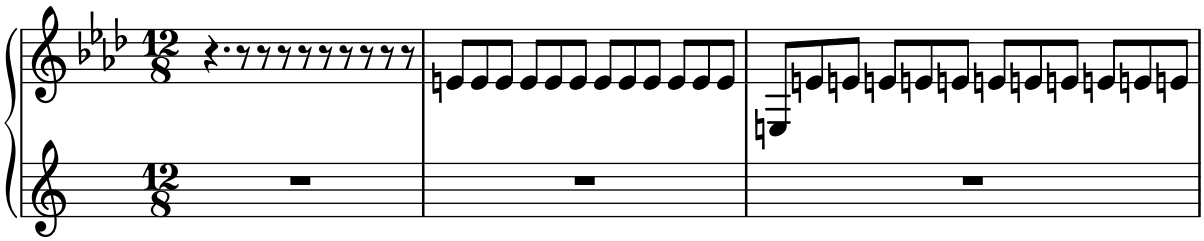
Span bars can be turned on/off on a staff-by-staff basis.

span-bar-partial.ly



Because BarLine grobs take their extra-positioning-height from their neighbors via the pure-from-neighbor-interface, the left edge of an accidental should never fall to the left of the right edge of a bar line. This spacing should also take place when SpanBar grobs are present.

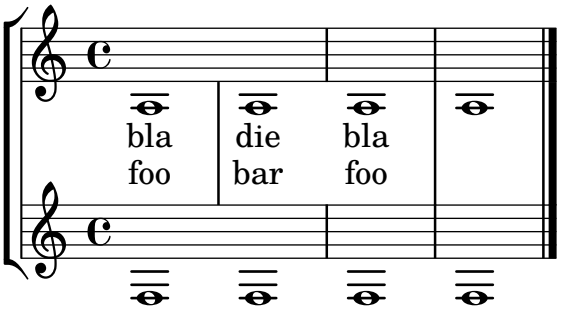
span-bar-spacing.ly



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

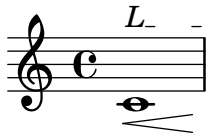
Setting SpanBar transparent removes the barlines between systems.
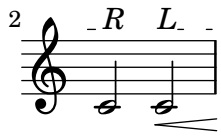
span-bar.ly
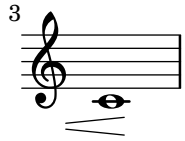


The visibility of left-broken line spanners and hairpins which end on the first note (i.e., span no time between bounds) is controlled by the callback ly:spanner::kill-zero-spanned-time.

spanner-after-line-breaking.ly

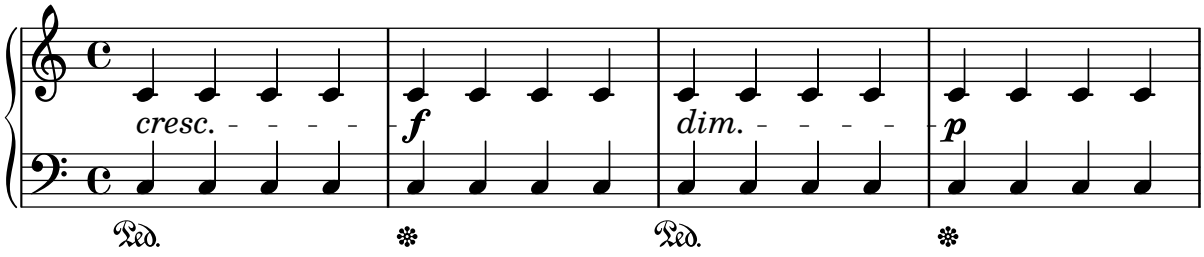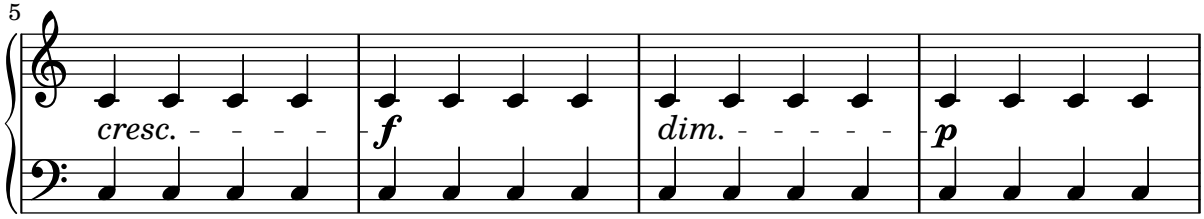






Spanners align to musical grobs in paper columns, ignoring things like pedal marks.

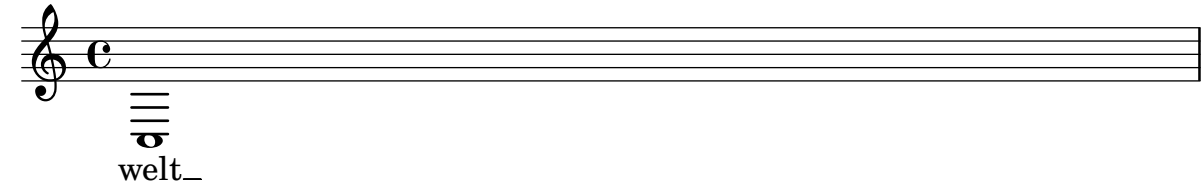
`spanner-alignment.ly`

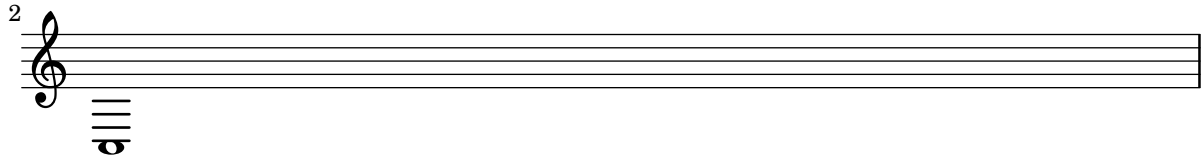




Spanners parts that extend beyond their parents are killed in case of line breaks.
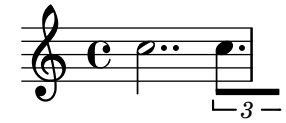
`spanner-break-beyond-parent.ly`
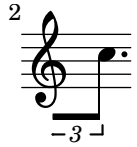




The `break-overshoot` property sets the amount that a spanner (in this case: the beam and tuplet bracket) in case of a line break extends beyond the rightmost column and extends to the left beyond the prefatory matter.
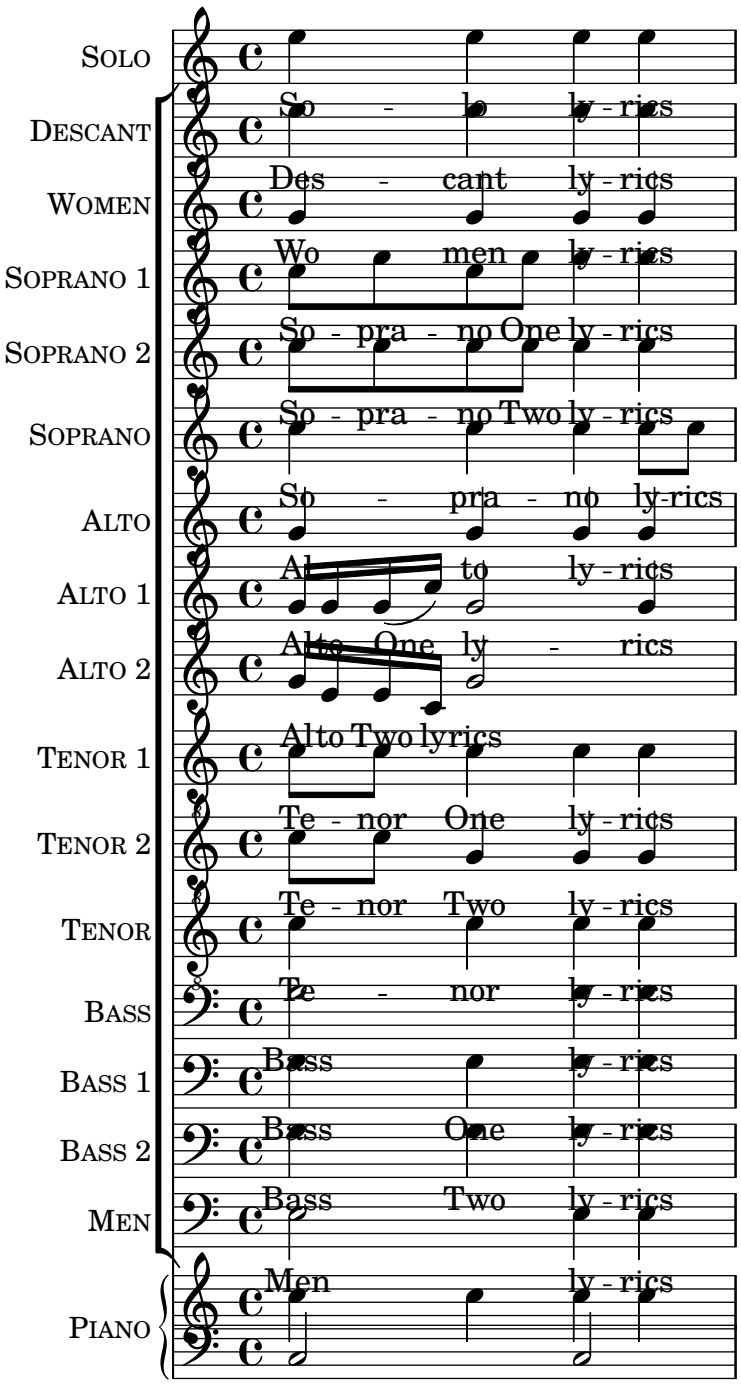
`spanner-break-overshoot.ly`





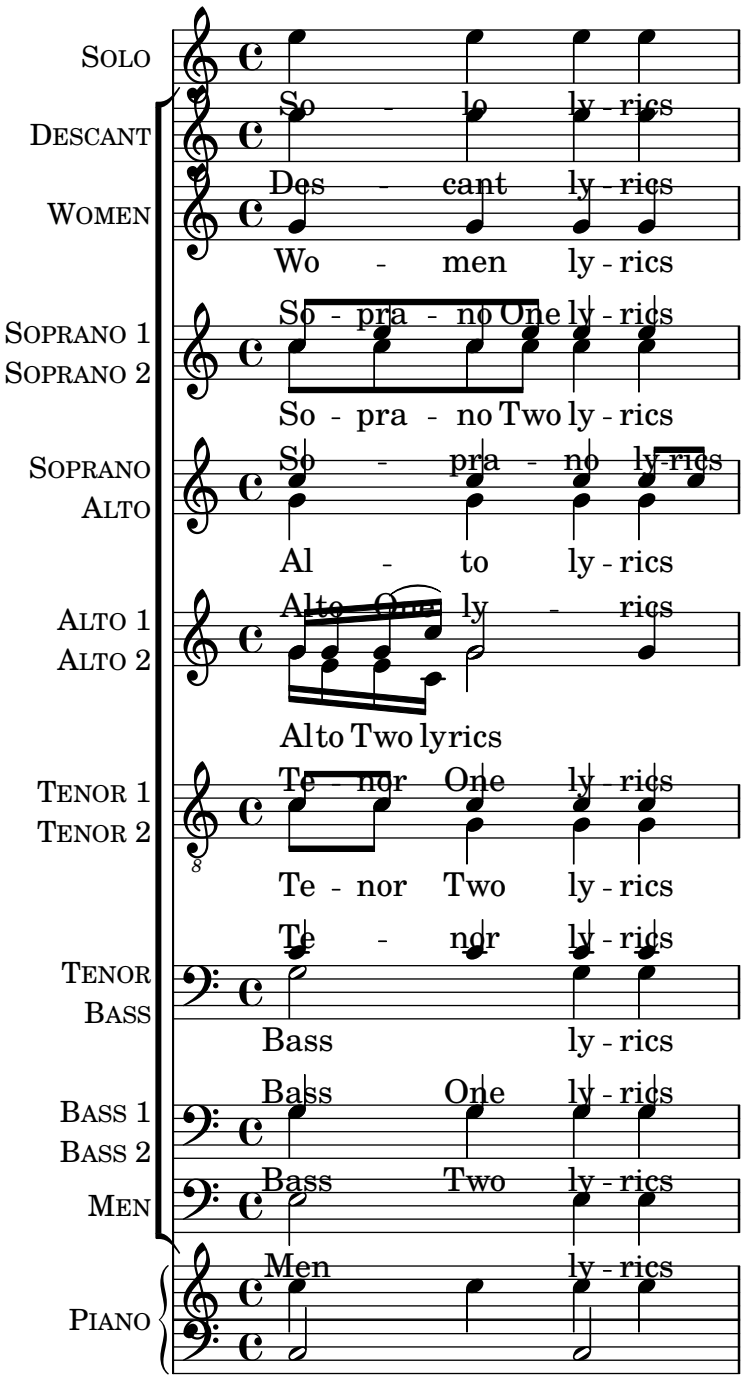
This should produce a choral score with solo, descant, women, sop 1 and 2, sop, alto, alto 1 and 2, tenor 1 and 2, tenor, bass, bass 1 and 2, men and piano staves. Normally the various combinations would appear at different times in the score, not all at once.

`ssaattbb-template-with-all-staves.ly`



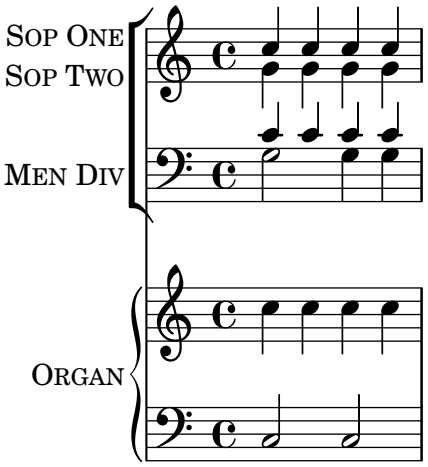
This should produce a choral score with solo, descant, women, sop divisi, sop and alto, alto divisi, tenor divisi, tenor and bass, bass divisi, men and piano staves. Normally the various combinations would appear at different times in the score, not all at once.

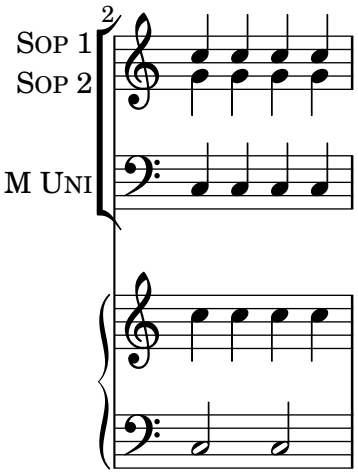
`ssaattbb-template-with-all-voices-on-one-staff.ly`



Instrument names and short instrument names can be changed when using the ssaattbb built-in template.

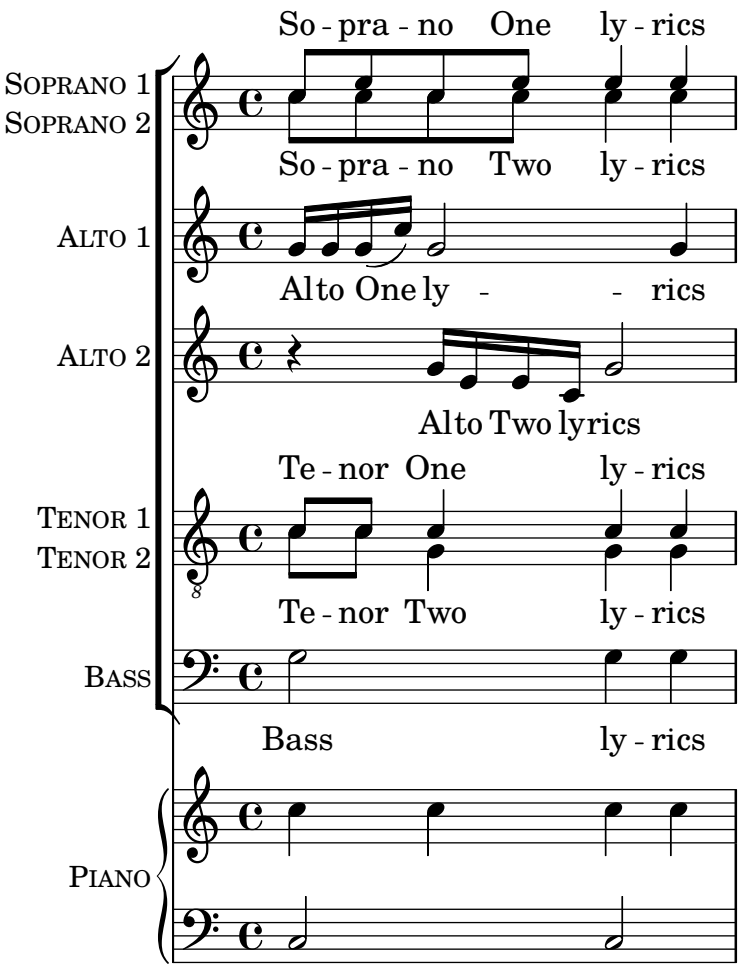
`ssaattbb-template-with-changed-instrument-names.ly`

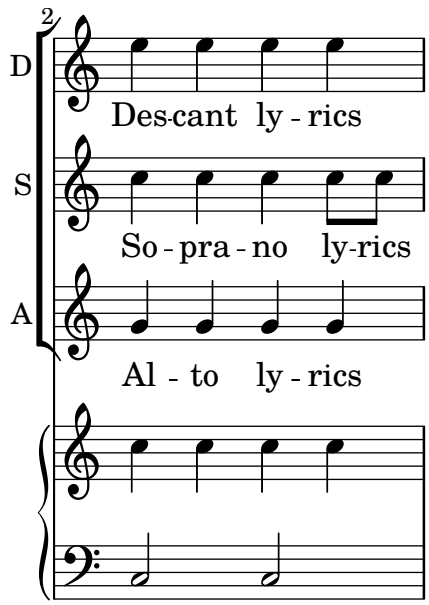


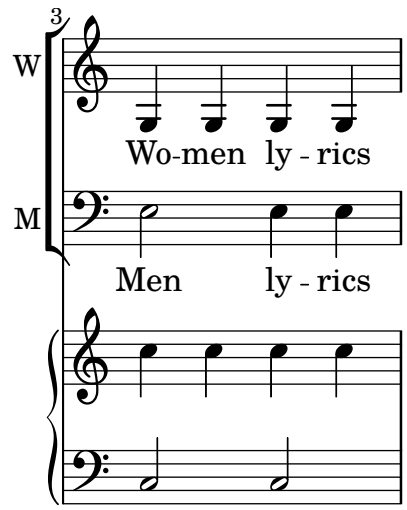


This should produce an SSAATTBB score with piano accompaniment, with divisi soprano and tenor on single staves, alto one and alto two on separate staves and unison bass in the first system, then unison soprano and alto voices with descant in the second system and unison women and unison men voices in the third system.

ssaattbb-template-with-men-women-and-descant.ly
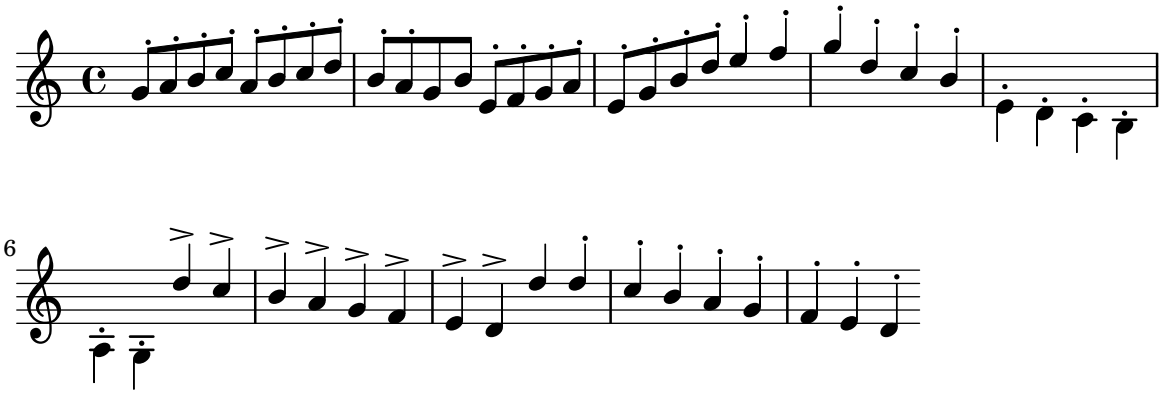






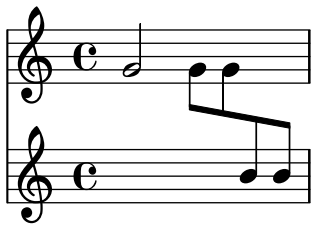
Some scripts must have quantized postions. VErtical position descend monotonously for a descending scale. The staccato dot is close to the notehead. If the head is in a space, then the dot is in the space next to it.

`staccato-pos.ly`
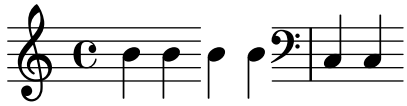


Staves stay alive long enough to complete an automatic beam.

`staff-change-autobeam.ly`
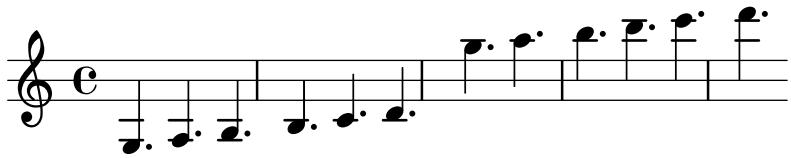


Staves can be started and stopped at command.

`staff-halfway.ly`



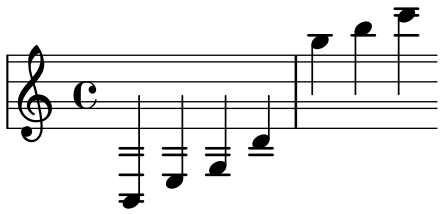
When the vertical positions of ledger lines have been customized by setting the `ledger-positions` property of the StaffSymbol, and a dotted note falls on a ledger line, the dot is shifted up to avoid the ledger line (just as with uncustomized ledger lines).

`staff-ledger-positions-dotted-notes.ly`
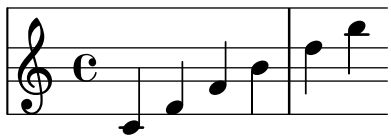


The vertical positions of ledger lines may be customised by setting the `ledger-positions` property of the StaffSymbol. The given pattern is repeated. Bracketed groups are always shown together: either all or none are shown. Ledger lines can be set to appear sooner or later by setting the `ledger-extra` property.

`staff-ledger-positions.ly`



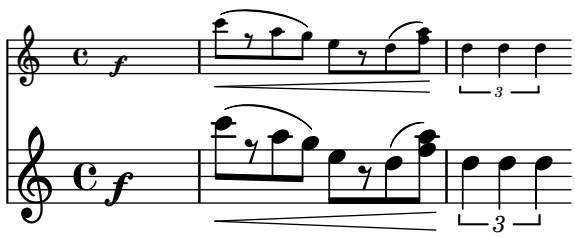
The vertical positions of staff lines may be specified individually, by setting the `line-positions` property of the StaffSymbol.

`staff-line-positions.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-mixed-size.ly`



Symbols that need on-staffline info (like dots and ties) continue to work in absence of a staff-symbol.
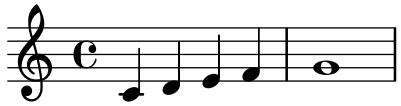
`staff-online-symbol-absence.ly`



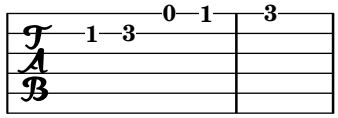
The space between scores containing Staffs and TabStaffs should be consistent. In this example, all of the spacings should be equivalent.

`staff-tabstaff-spacing.ly`
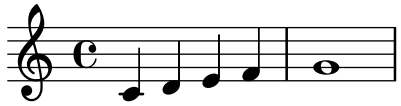
Title 1



Title 2
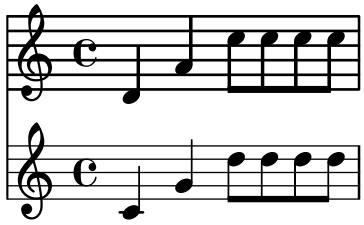


Title 3
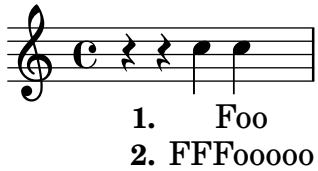


The staff is a grob (graphical object) which may be adjusted as well, for example, to have 6 thick lines and a slightly large `staff-space`. However, beams remain correctly quantized.

`staff-tweak.ly`



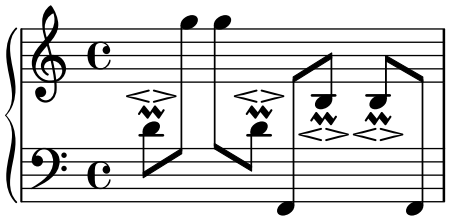
Stanza numbers are put left of their lyric. They are aligned in a column.

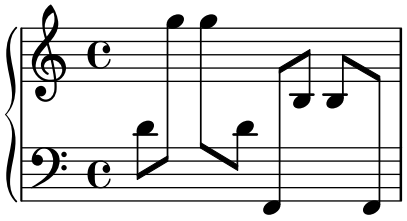
`stanza-number.ly`



Cross-staff stems avoid articulations. Articulations that don't get in the way of stems do not cause unwanted horizontal space.
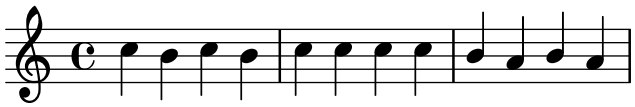
`stem-cross-staff-articulation.ly`
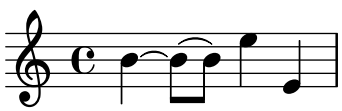




Stem directions for notes on the middle staff line are determined by the directions of their neighbors.

`stem-direction-context.ly`
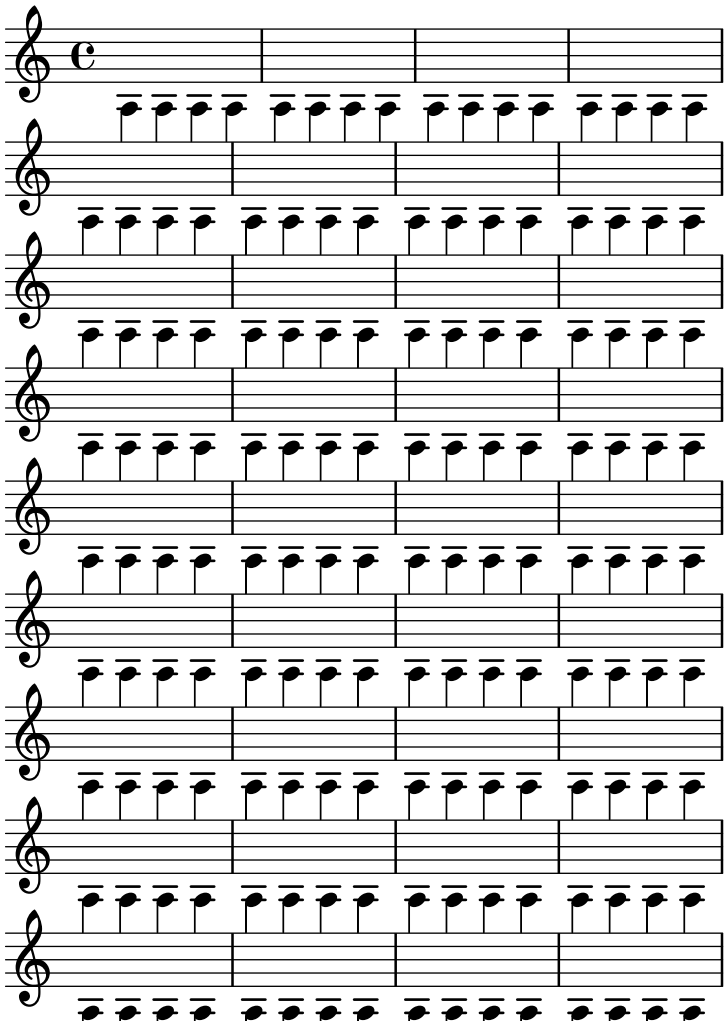


Stems, beams, ties and slurs should behave similarly, when placed on the middle staff line. Of course stem-direction is down for high notes, and up for low notes.

`stem-direction.ly`
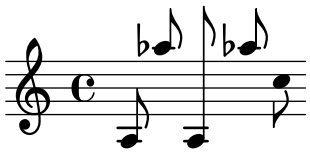


Stems with overridden 'Y-extent should not confuse height estimation. This example should fit snugly on one page.

`stem-length-estimation.ly`



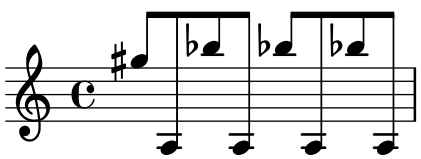
Stem length and stem-begin-position can be set manually.

`stem-length.ly`



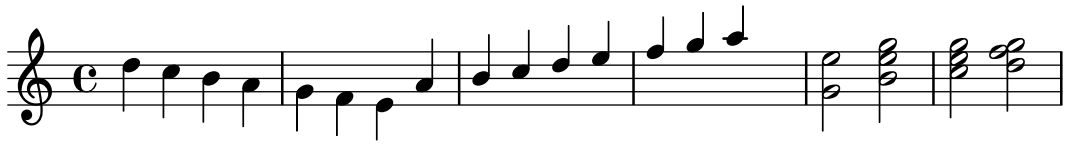
Lilypond gets beamed stem pure heights correct to avoid outside staff collisions.

`stem-pure-height-beamed.ly`
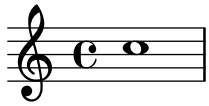


If note head is ‘over’ the center line, the stem is shortened. This happens with forced stem directions, and with some chord configurations.

`stem-shorten.ly`
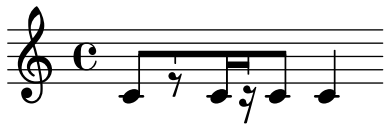


Stemlets don't cause stems on whole notes.

`stem-stemlet-whole.ly`
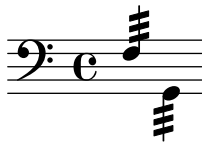


Stemlets are small stems under beams over rests. Their length can be set with `stemlet-length`.

`stem-stemlet.ly`



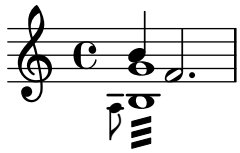
Tremolo works even when a stem is forced in a particular direction.

`stem-tremolo-forced-dir.ly`
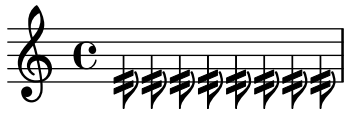


Tremolos should avoid other notes in the staff as best as possible and issue a warning otherwise.

`stem-tremolo-note-collision.ly`
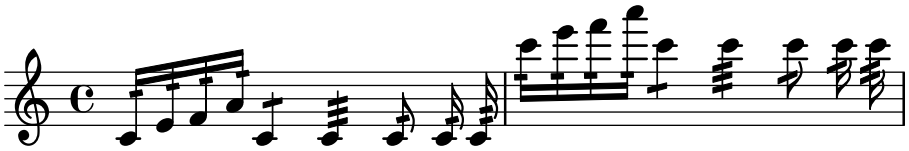


Stem tremolos count in a note column's horizontal skyline.

`stem-tremolo-note-column.ly`



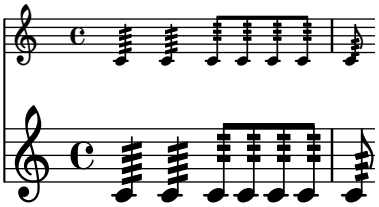
Tremolos are positioned a fixed distance from the end of the beam. Tremolo flags are shortened and made rectangular on beamed notes or on stem-up notes with a flag. Tremolo flags are tilted extra on stem-down notes with a flag.

`stem-tremolo-position.ly`



stem tremolo vertical distance also obeys staff-space settings.
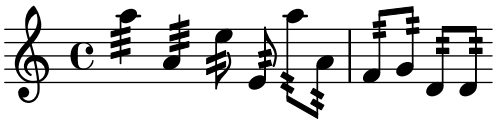
`stem-tremolo-staff-space.ly`
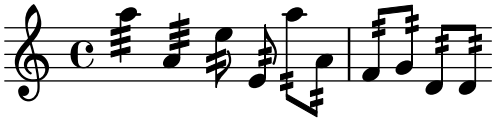


Controlling the appearance of tremolo slashes. Property `slope` is self-explanatory. Property `shape` determines whether slashes look like rectangles or like very small beams. Setting these properties directly cause all slashes behave in the specified way. However, one usually wants the slashes to behave differently depending on whether the note has flags, beams or only a plain stem. That's what the `style` property is used for: it sets shape and slope depending on the situation. There are two styles defined: `default` and `constant`.

`stem-tremolo-style.ly`
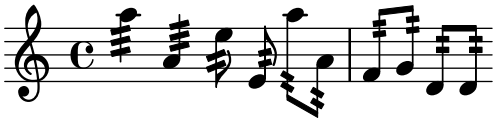
default. First three notes should have beam-like slashes. Slash of the third note should be more sloped than first two notes. Slashes on beamed notes should be rectangular and parallel to the beams.
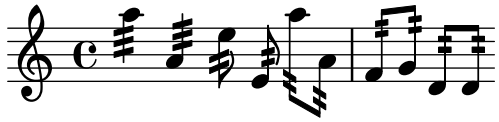


style=constant. All slashes should be rectangular. All slashes should have the same slope except for downstem flagged notes.
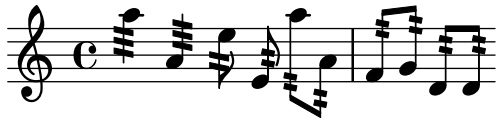


shape=rectangle. All slashes should be rectangular. Slope like in default.



shape=beam-like. All slashes should be beam-like. Slope like in default.



slope=-0.2 All slashes should have the same downward slope. Shape like in default.
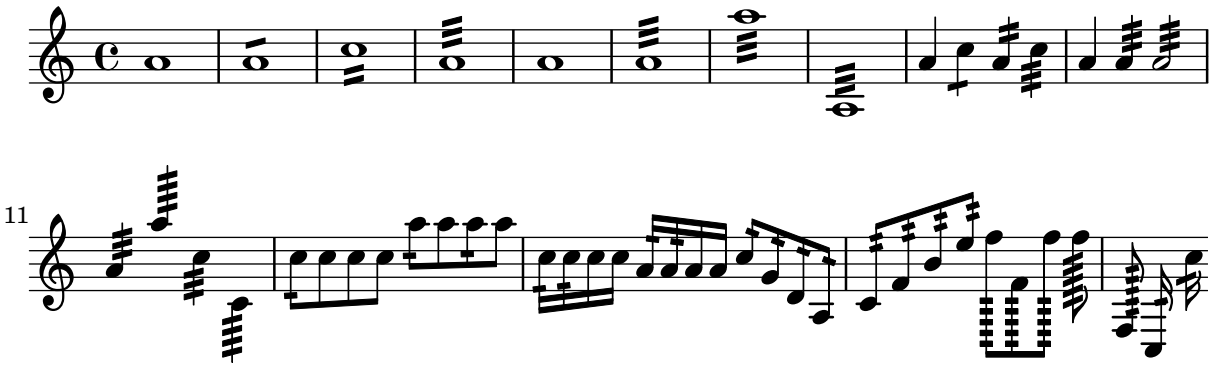


Stem tremolos or rolls are tremolo signs that look like beam segments crossing stems. If the stem is in a beam, the tremolo must be parallel to the beam. If the stem is invisible (e.g. on a whole note), the tremolo must be centered on the note. If the note has a flag (eg. an unbeamed 8th note), the tremolo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

If an impossible tremolo duration (e.g. :4) is given, a warning is printed.

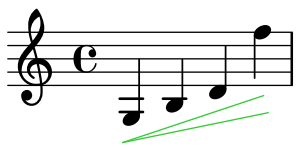
`stem-tremolo.ly`

:4 :8 :16 :32 x :



Combinations of rotation and color do work.
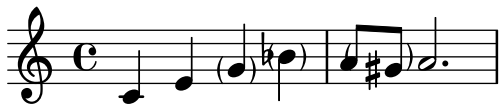
`stencil-color-rotation.ly`



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

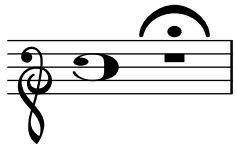
The parenthesized beam is less successful due to implementation of the Beam. The note head is also rather naive, since the extent of the parens are also not seen by accidentals.

`stencil-hacking.ly`



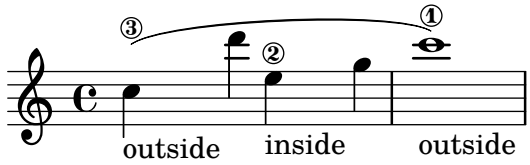
Stencils can be scaled using `ly:stencil-scale`. Negative values will flip or mirror the stencil without changing its origin; this may result in collisions unless the scaled stencil is realigned (e.g., the time signature in this test).

`stencil-scale.ly`
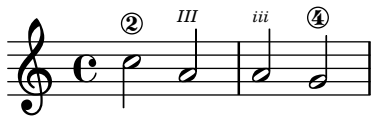


String numbers should only be moved outside slurs when there is a collision.

`string-number-around-slur.ly`



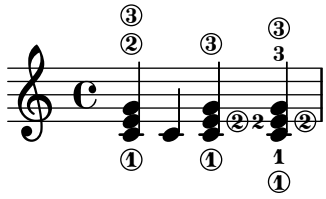
Different styles may be used for string number indications. Predefined options are arabic (used by default) and roman numerals.

`string-number-styles.ly`



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

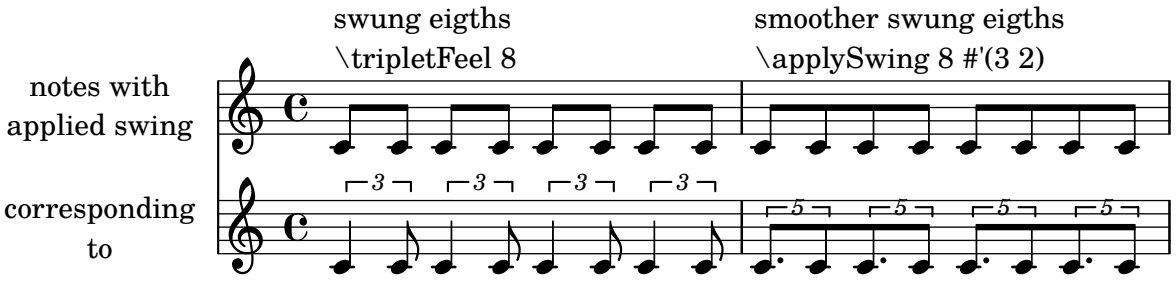
`string-number.ly`

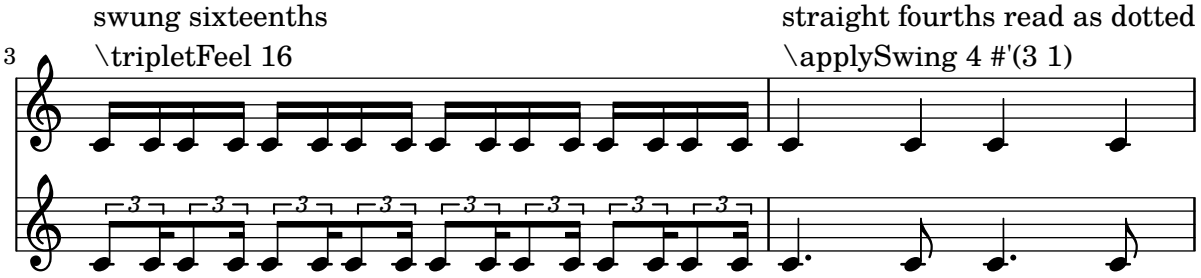


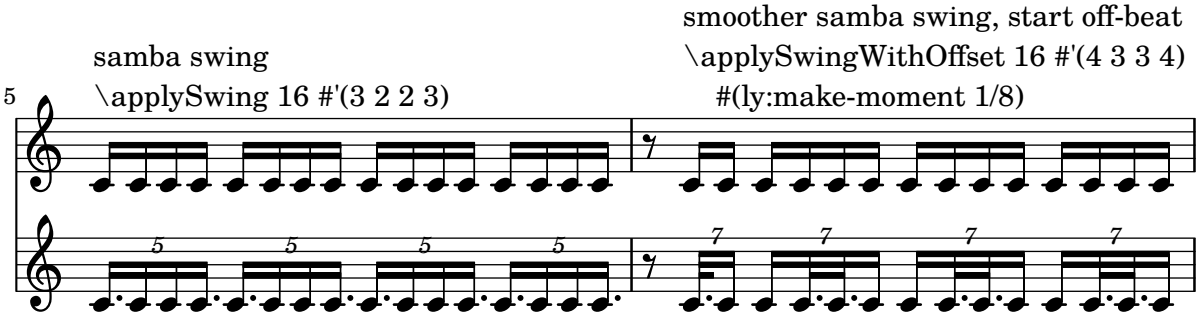
Tests for swing.ly

`swing-test.ly`

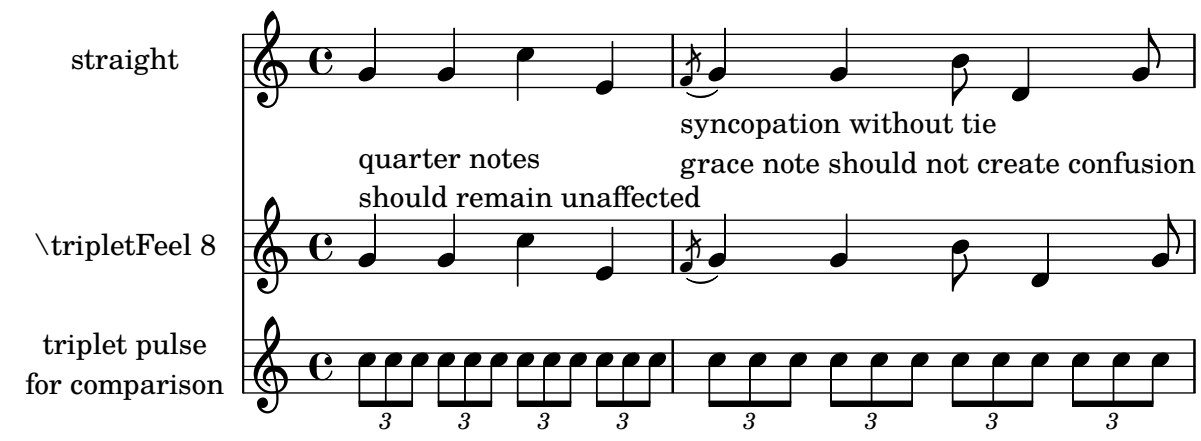
1. Swing type demos

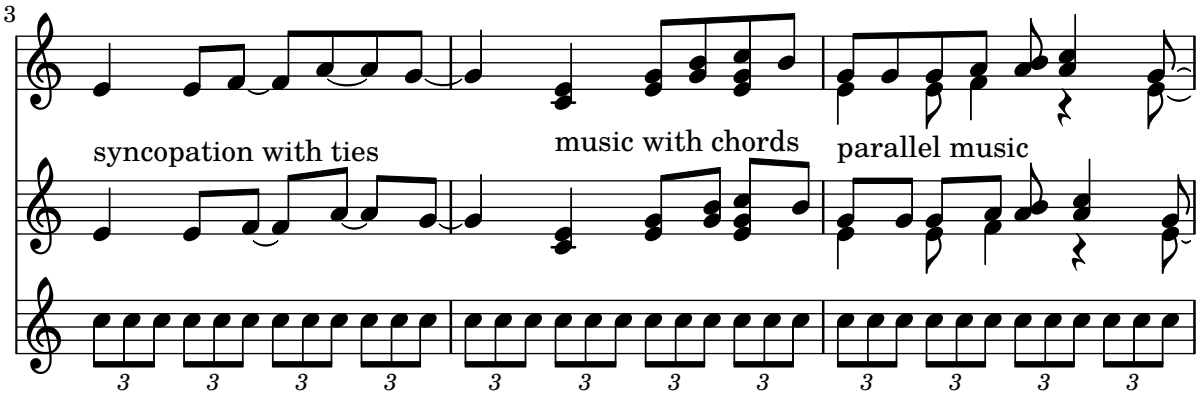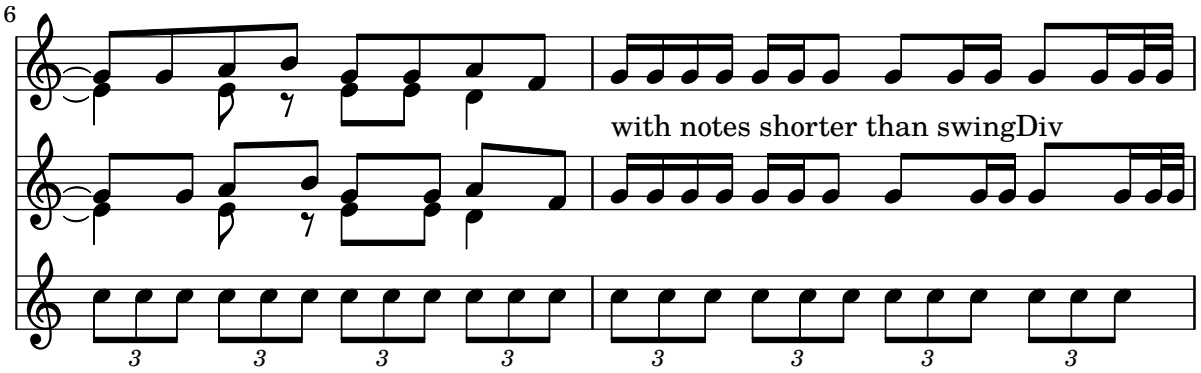






2. Triplet feel in various situations

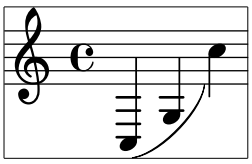






The size of every system is correctly determined; this includes postscript constructs such as slurs.

`system-extents.ly`



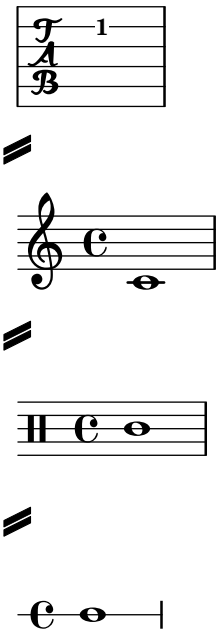
By setting the padding between systems to a negative value, it is possible to eliminate the anti-collision constraints.

system-overstrike.ly



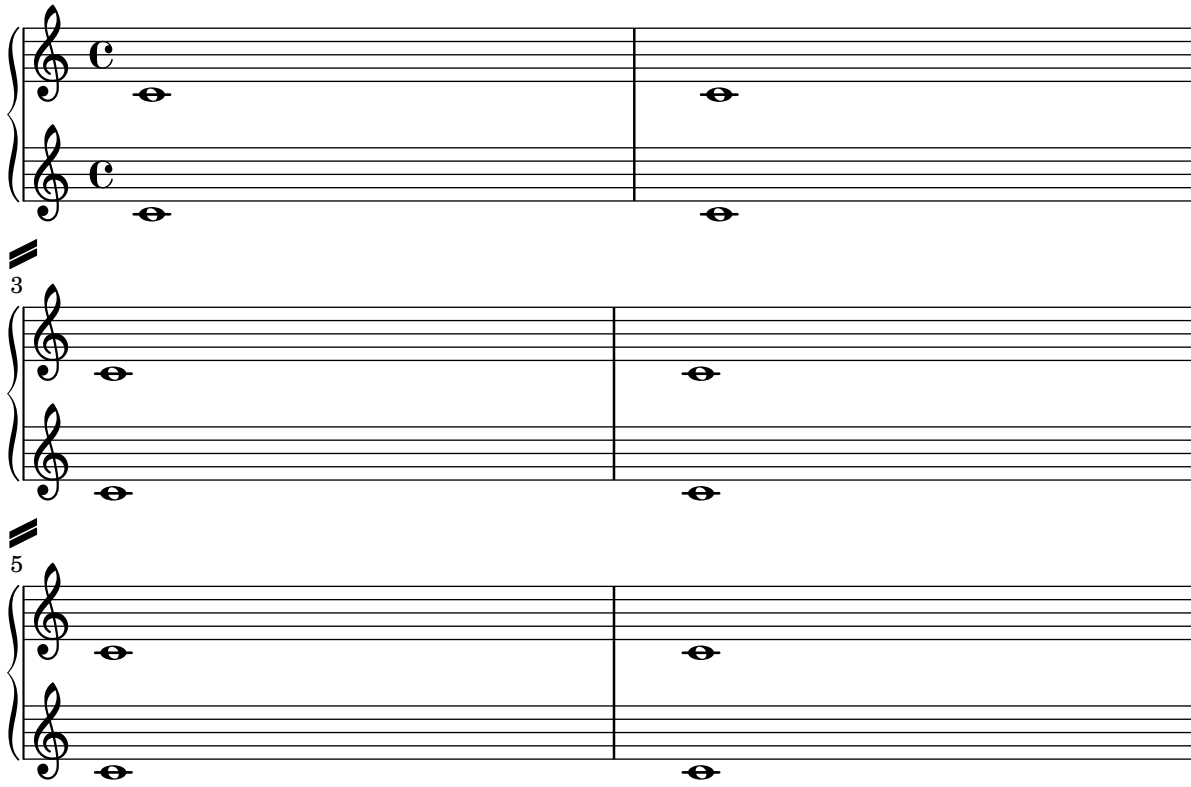
System separator positioning works with all spaceable staff contexts.

system-separator-spaceable-staves.ly



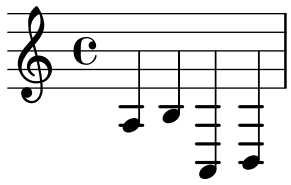
System separators may be defined as markups in the `system-separator-markup` field of the paper block. They are centered between the boundary staves of each system.

system-separator.ly



When the staff-space is increased, the system-start delimiter should still be collapsed (i.e. the collapse-height should not give an absolute length, but a multiple of staff-spaces).

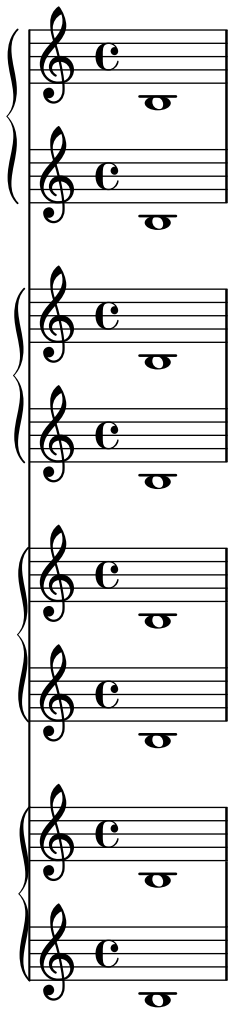
`system-start-bar-collapse-staffspace.ly`



Disregarding the value of `systemStartDelimiter`, setting `SystemStart`*`Grob`* style of `StaffGroup` to `'brace` always prints a `SystemStartBrace`.

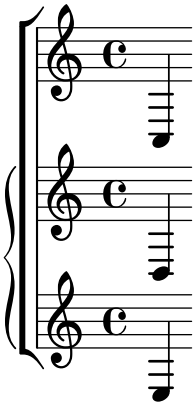
Every `StaffGroup` should start with a `SystemStartBrace`.

`system-start-brace-style.ly`
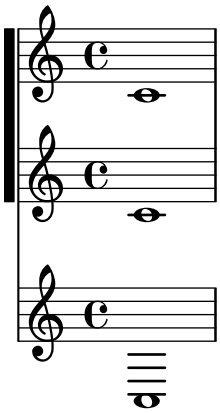


A piano context included within a staff group should cause the piano brace to be drawn to the left of the staff angle bracket.

`system-start-bracket.ly`



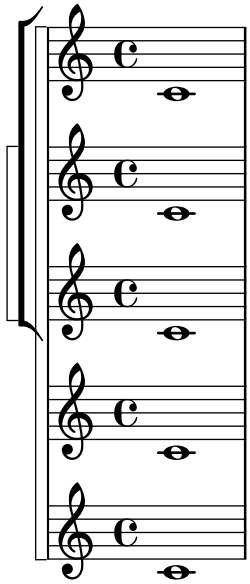
A heavy-bar system start delimiter may be created by tuning the `SystemStartBar` grob.

`system-start-heavy-bar.ly`



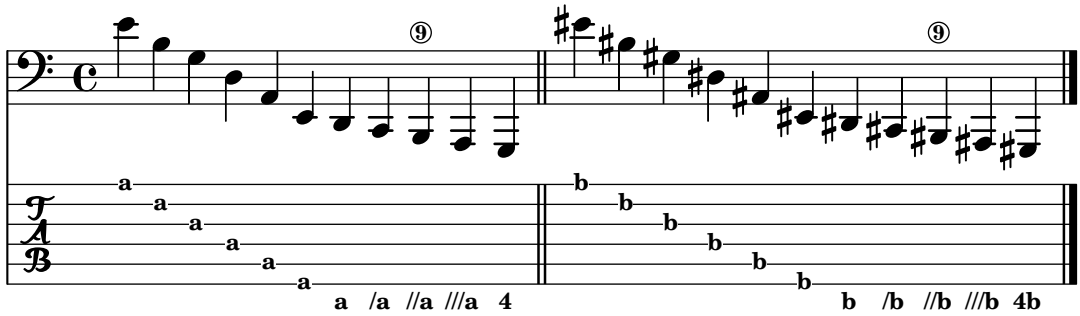
Deeply nested system braces, brackets, etc., may be created with the `systemStartDelimiterHierarchy` property.

`system-start-nesting.ly`



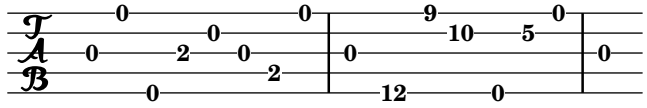
Additional bass strings (for baroque lute, etc.) are supported in TabStaff. They are printed below lowest line as: a, /a, //a, ///a, 4, 5 etc. `additionalBassStrings` needs to be set accordingly.

`tablature-additional-bass-strings.ly`



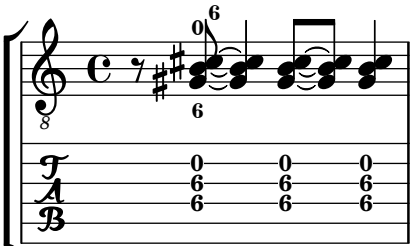
Tablature may also be tuned for banjo.

`tablature-banjo.ly`



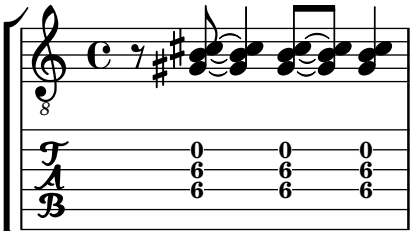
In a TabStaff, the chord repetition function needs to retain string and fingering information. Using `\tabChordRepeats` achieves that, in contrast to the music on the main staff.

`tablature-chord-repetition-finger.ly`



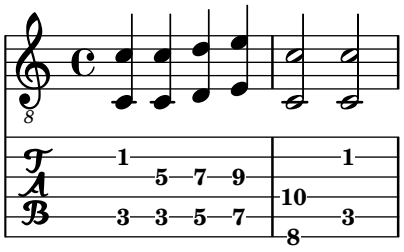
In a TabStaff, the chord repetition function needs to save the string information. The obsolete function `\tabChordRepetition` establishes this setting score-wide. Nowadays, you would rather use just `\tabChordRepeat` on the music in the tabstaff, not affecting other contexts.

`tablature-chord-repetition.ly`



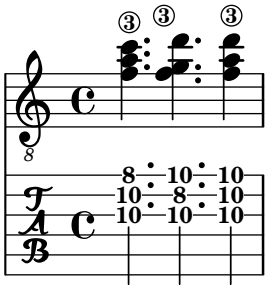
Context property `defaultStrings` defines desired strings for fret calculations if no strings are defined explicitly.

`tablature-default-strings.ly`



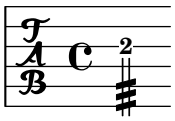
With full notation, the dots on the tablature heads should respect two-digit fret numbers.

`tablature-dot-placement.ly`



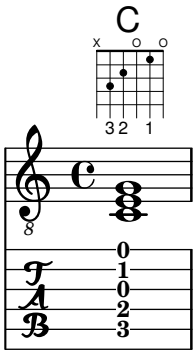
Tremoli applied to double stems in a `TabVoice` should be centered on the double stem.

`tablature-double-stem-tremolo.ly`



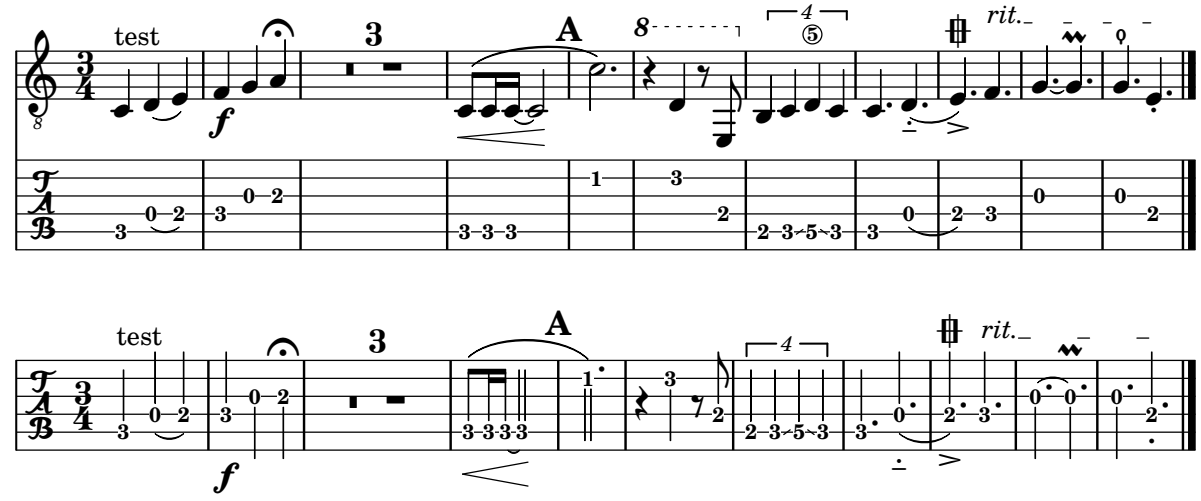
Tablatures derived from stored fretboard diagrams display open strings as fret 0 in the tablature. The tablature and fretboard should match.

`tablature-fretboard-open-string.ly`



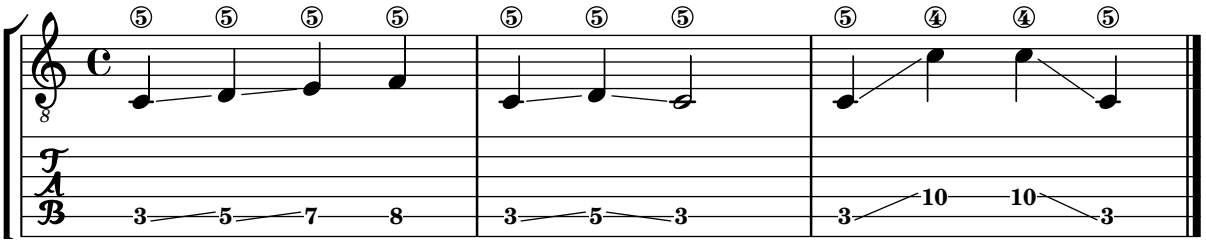
As default, tablature staves show only the fret numbers, because in most situations, they are combined with normal staves. When used without standard notation, `tabFullNotation` can be used.

`tablature-full-notation.ly`



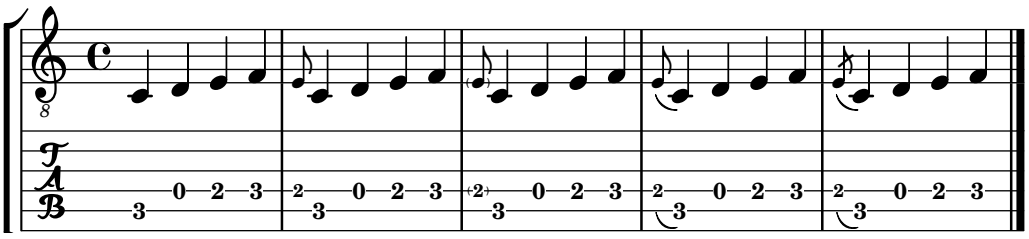
Glissando lines in tablature have the right slope.

`tablature-glissando.ly`



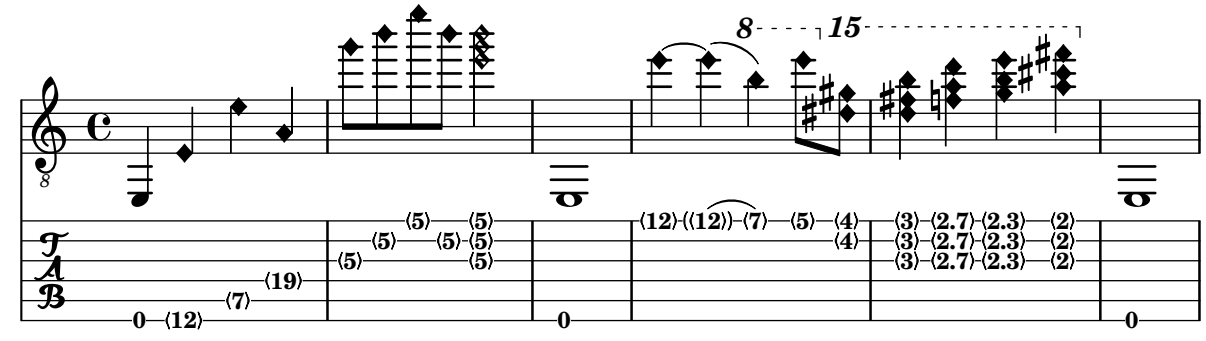
Fret numbers belonging to grace notes are smaller.

`tablature-grace-notes.ly`



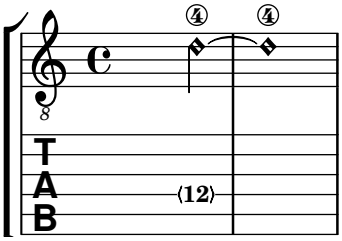
Harmonics can be specified either by ratio or by fret number.

`tablature-harmonic-functions.ly`



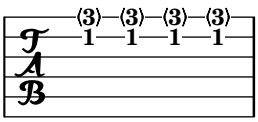
When a harmonic note is tied in tablature, neither the fret number nor the harmonic brackets for the second note appear in the tablature.

`tablature-harmonic-tie.ly`



Harmonics get angled brackets in tablature. Harmonics in chords should retain their proper position, regardless of whether or not strings are specified. In this example, the harmonics should always be on string 1.

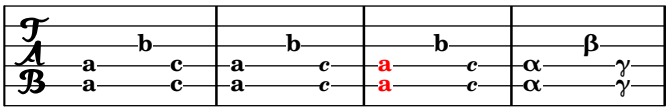
`tablature-harmonic.ly`



A sample tablature with lettered tab, using fretLabels to modify the fret letters.

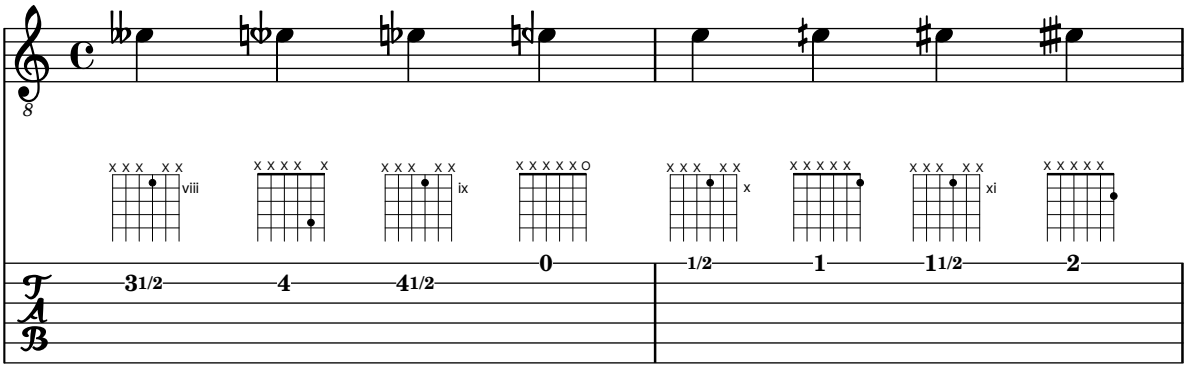
By default, letters are drawn sequentially from the alphabet, but if the context property fretLabels is defined, these are substituted. If specified, the length of fretLabels must be sufficient to label all the frets used. A warning is issued if the length is too short.

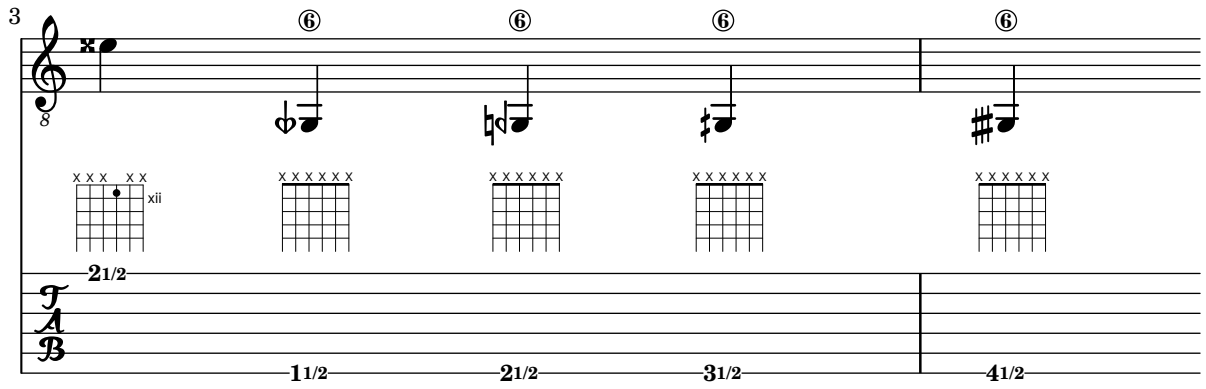
`tablature-letter.ly`



The `TabStaff` will print micro-tones as mixed numbers of fret-number and a fraction. The context-property `supportNonIntegerFret` needs to be set `#t` in `Score`-context. `FretBoards` will print those micro-tones only if they can be found in the chosen settings for `stringTunings`, otherwise a warning (surpressed here) will be printed and an empty `FretBoard` returned. Which should be the case for the last four of the examples pitches. Micro-tones assigned to strings work nicely.

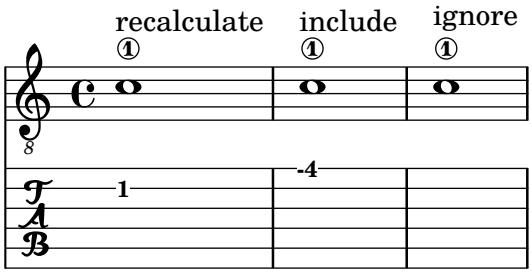
`tablature-micro-tone.ly`





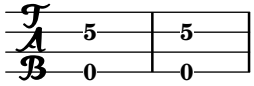
Negative fret numbers calculated due to assigning a string number can be displayed, ignored, or recalculated. Here we should have all three cases demonstrated.

`tablature-negative-fret.ly`



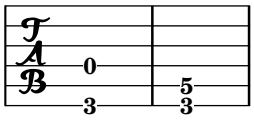
Open strings can always be part of a chord in tablature, even when frets above 4 have been used in the chord. In this case, both chords should show an open fourth string.

`tablature-open-string-chord.ly`



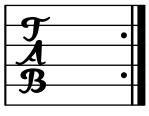
Open strings are part of a chord in tablature, even when `minimumFret` is set. This can be changed via `restrainOpenStrings`.

`tablature-open-string-handling.ly`



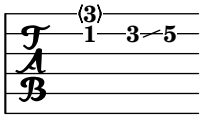
How a repeat sign looks in tablature.

`tablature-repeat.ly`



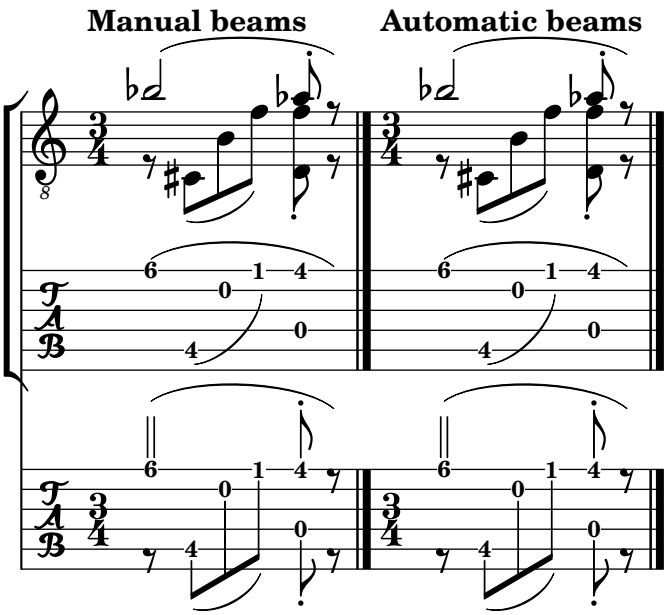
Tab supports slides.

`tablature-slide.ly`



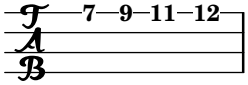
Slur placement in complementary tablatures should not be affected by either automatic or manual beaming.

`tablature-slurs-with-beams.ly`



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

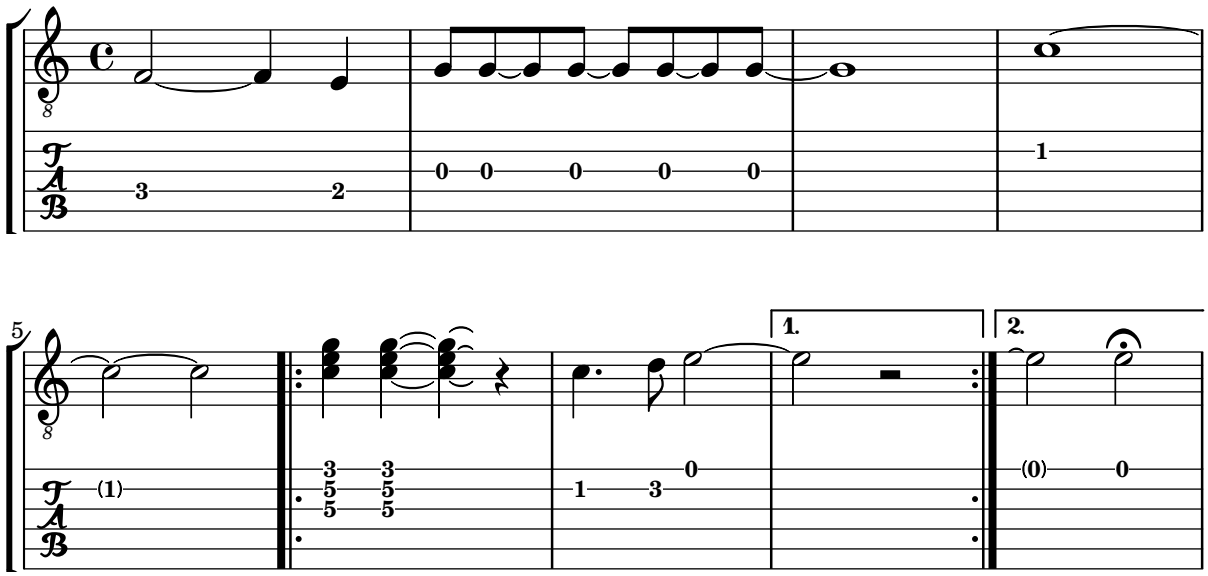
`tablature-string-tunings.ly`

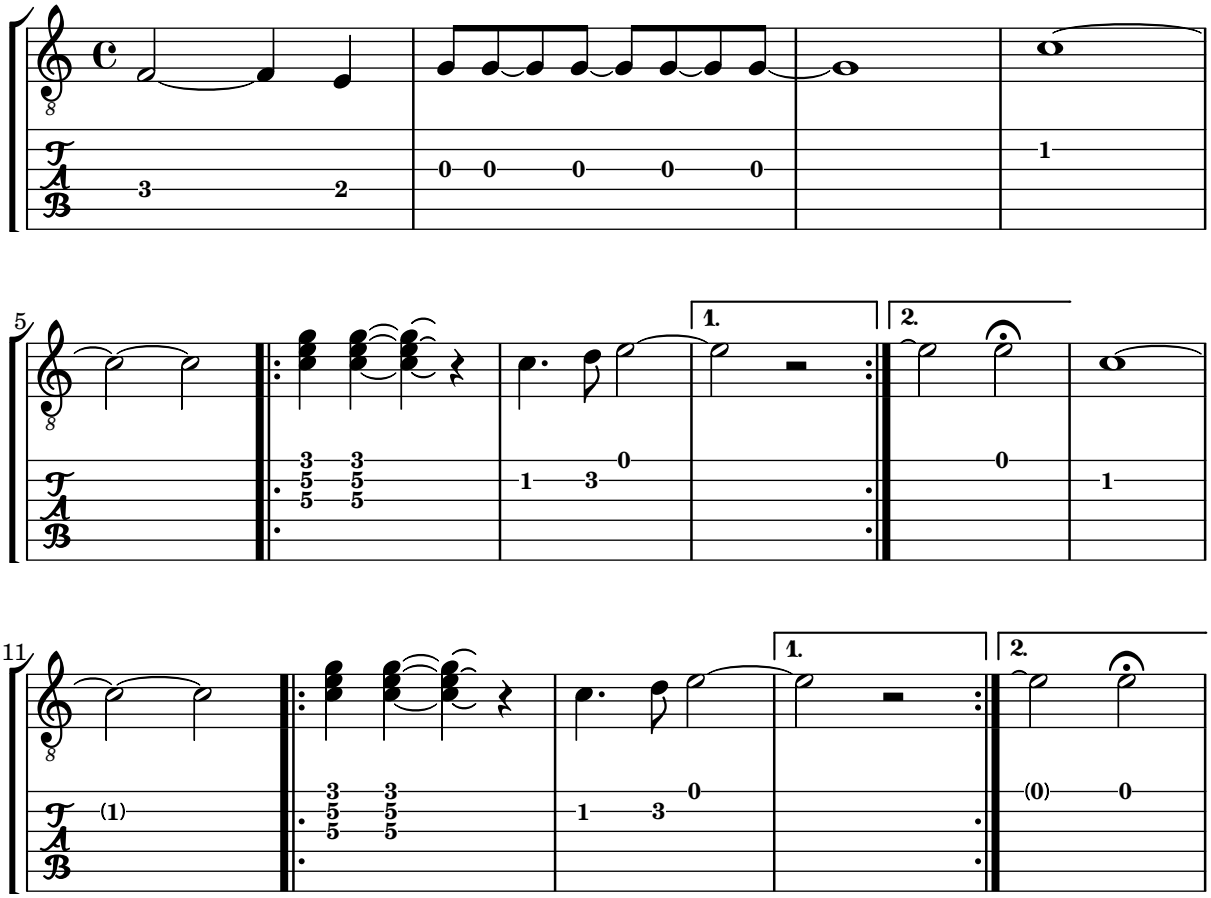


In tablature, notes that are tied to are invisible except after a line break or within a second volta; here, the fret number is displayed in parentheses.

As an option, the notes that are tied to may become invisible completely, even after line breaks.

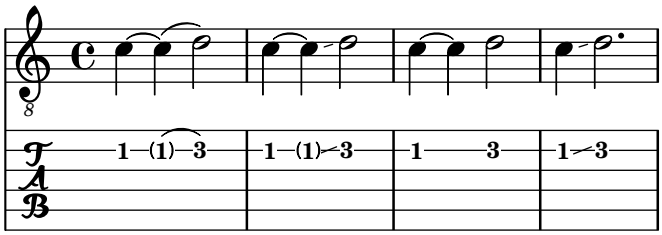
`tablature-tie-behaviour.ly`





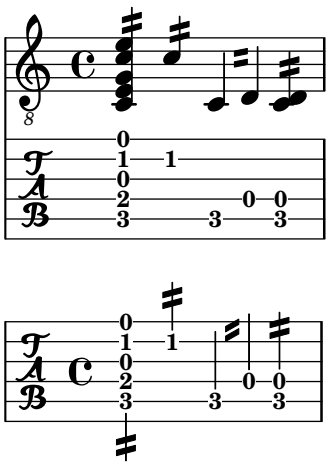
If a slur or a glissando follows a tie, the corresponding fret number is displayed in parentheses.

`tablature-tie-spanner.ly`



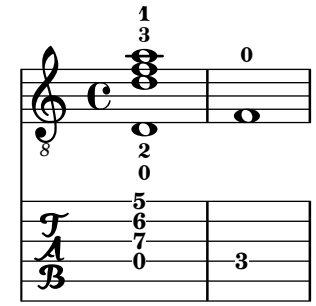
Tremolos will appear on tablature staffs only if `\tabFullNotation` is active. Otherwise, no tremolo indications are displayed on the TabStaff. Also, tablature beams are the same thickness on TabStaff and Staff.

`tablature-tremolo.ly`



A fingering indication of zero counts as an open string for fret calculations. An inappropriate request for an open string will generate a warning message and set the requested pitch in the tablature.

`tablature-zero-finger.ly`

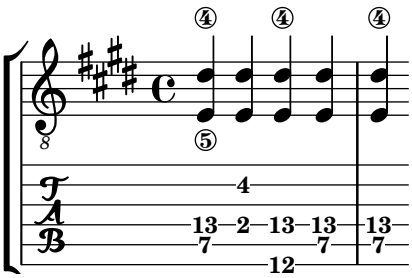


A sample tablature, with both normal staff and tab.

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.

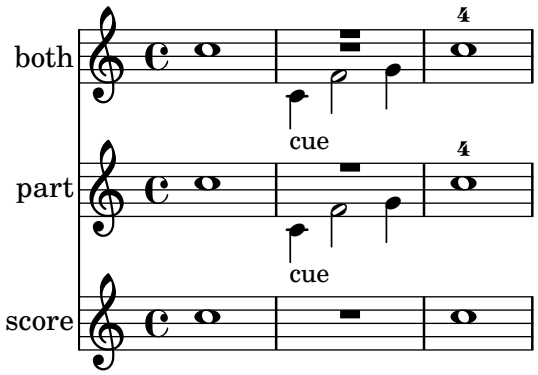
String numbers can be entered as note articulations (inside a chord) and chord articulations (outside a chord)

`tablature.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.

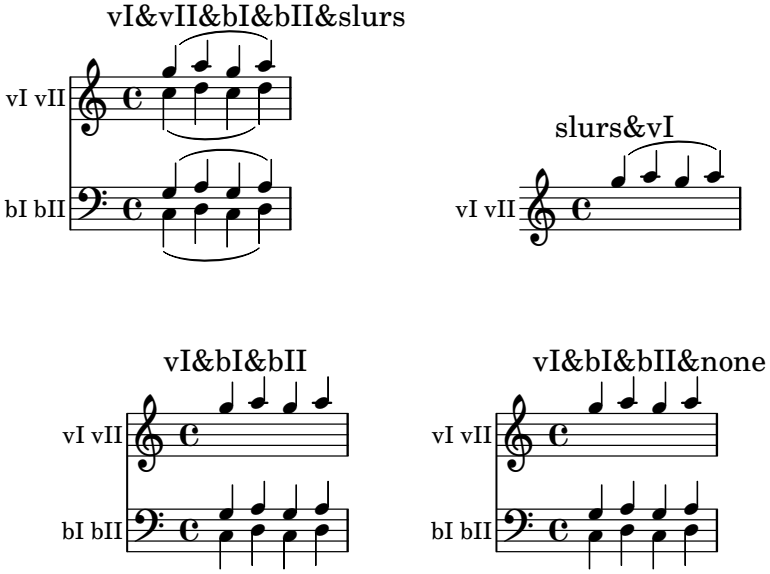
`tag-filter.ly`



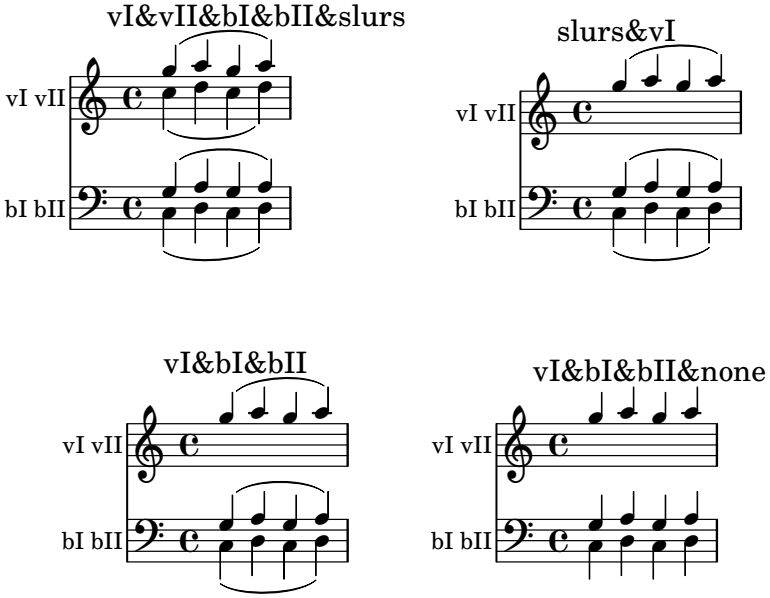
The operation of `\keepWithTag` can be made more flexible by using `\tagGroup`.

`tag-group.ly`

\keepWithTag



\tagGroup vI.vII
\tagGroup bI.bII

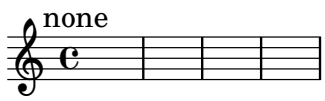


The `\removeWithTag` and `\keepWithTag` commands can name multiple tags to remove or to keep.
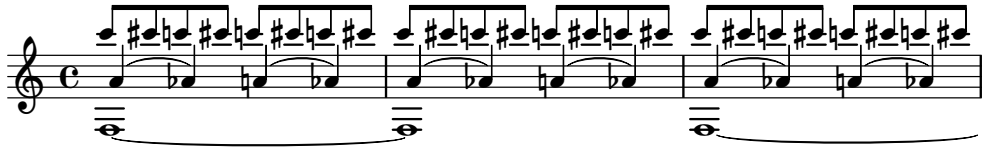
tag-multiple.ly
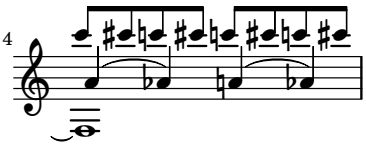
\keepWithTag

none

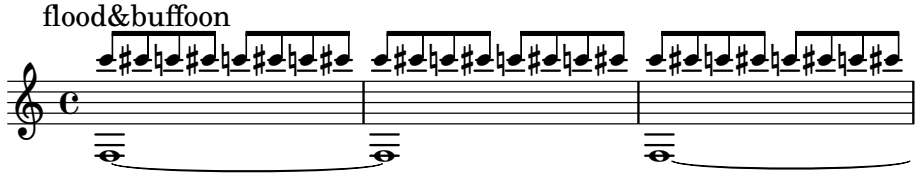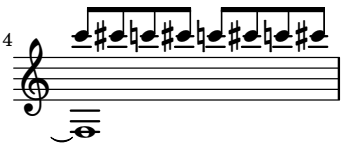


flood&highball&buffoon



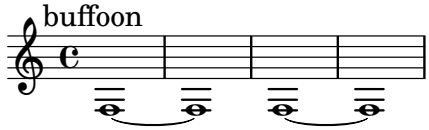


flood&buffoon



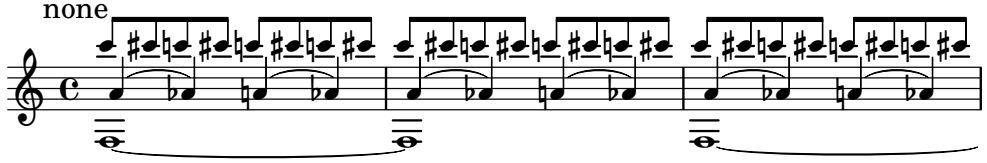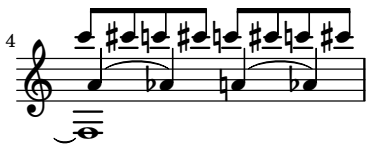


buffoon
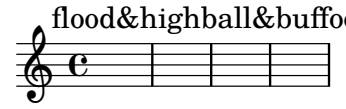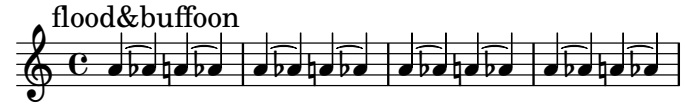


\removeWithTag
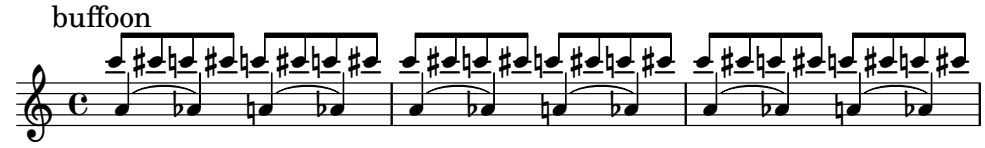
none
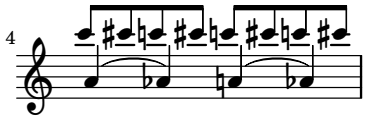






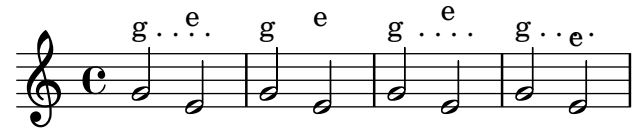






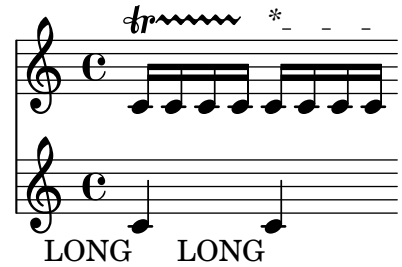
`TextScript`s are spaced closely, following outlines of the stencil. When markup commands like `pad-around` and `with-dimensions` change the extent of a stencil, these changed extents have effect in the stencil outline used to place the resulting `TextScript`.

`text-script-vertical-skylines.ly`



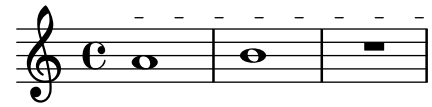
Text and trill spanners are attached to note columns, so attachments in other staves have no effect on them.

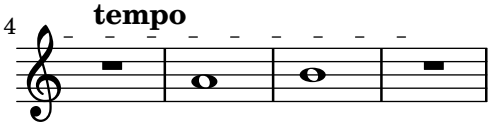
`text-spanner-attachment-alignment.ly`



Text spanners ending on, or broken across, full-measure rests extend to the rests, or over the rests, as appropriate.

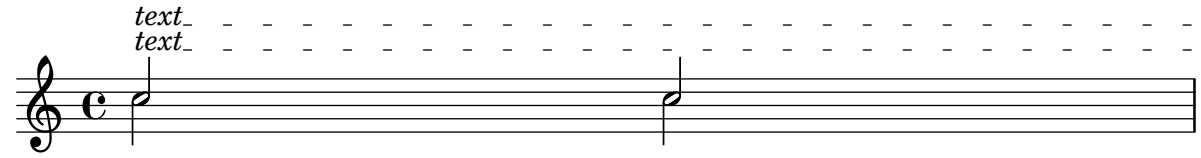
`text-spanner-full-rest.ly`

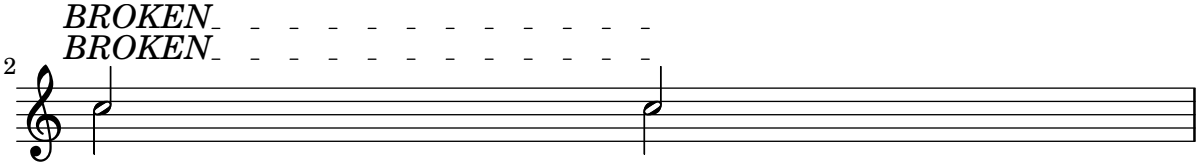




The order of setting nested properties does not influence text spanner layout.

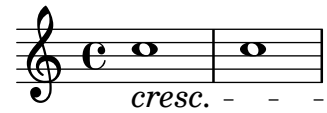
`text-spanner-override-order.ly`

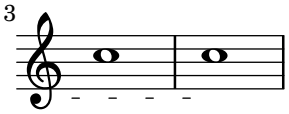




Text spanners should not repeat start text when broken.

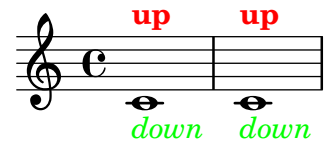
`text-spanner.ly`





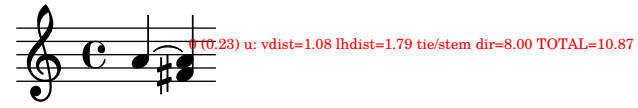
`\etc` can be used for constructing event functions for '`TextScript`' events with sequences starting with '`-`', '`^`', or '`_`'. This example should have notes all adorned in the same manner.

`textetc.ly`
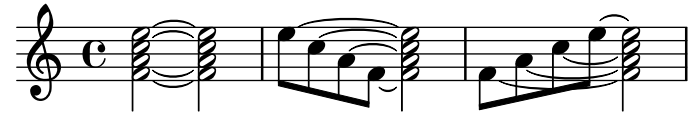


lilypond should flip the tie's direction to avoid a collision with the sharp.

`tie-accidental.ly`
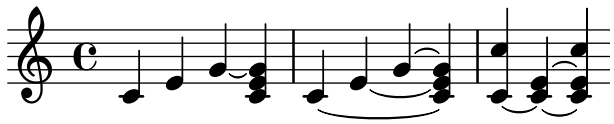


Advanced tie chord formatting also works with arpegiated ties. Due to arpeggios, tie directions may be changed relative to the unarpegiated case.

`tie-arpeggio-collision.ly`
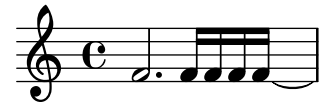


when `tieWaitForNote` is set, the right-tied note does not have to follow the lef-tied note directly. When `tieWaitForNote` is set to false, any tie will erase all pending ties.
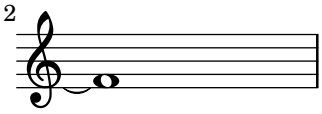
`tie-arpeggio.ly`



Broken ties honor `minimum-length` also. This tie has a `minimum-length` of 5.

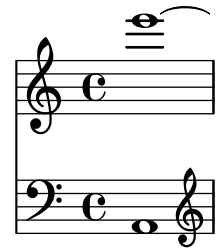
`tie-broken-minimum-length.ly`
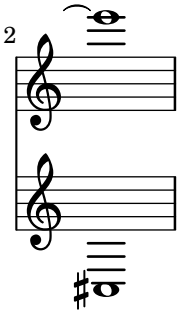




Broken tie lengths are not affected by clefs in other staves.

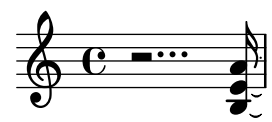
`tie-broken-other-staff.ly`
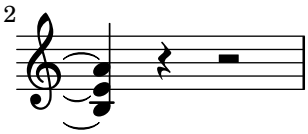




Ties behave properly at line breaks.

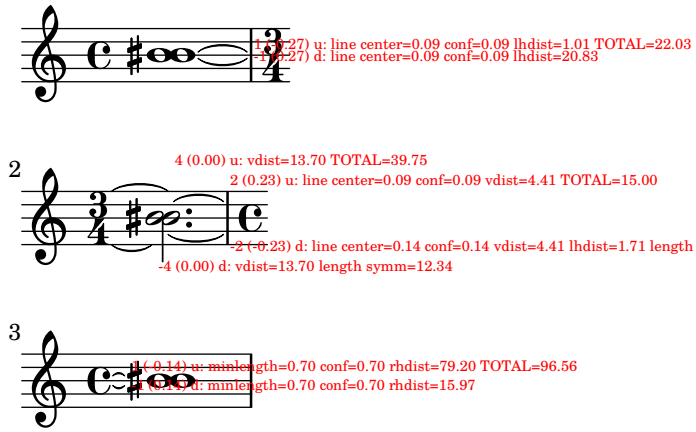
`tie-broken.ly`





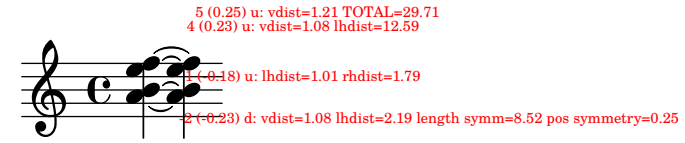
Tie detail property multi-tie-region-size controls how many variations are tried for the extremal ties in a chord.

`tie-chord-broken-extremal.ly`



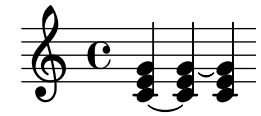
Switching on debug-tie-scoring annotates the tie scoring decisions made.

`tie-chord-debug.ly`



Individual chord notes can also be tied
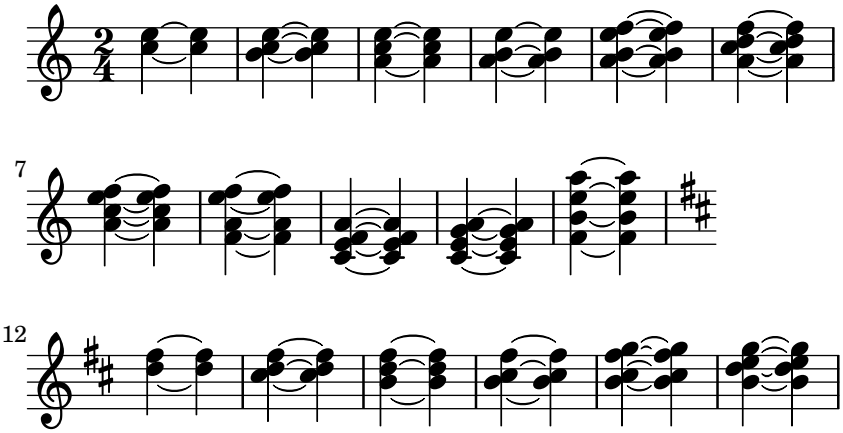
`tie-chord-partial.ly`
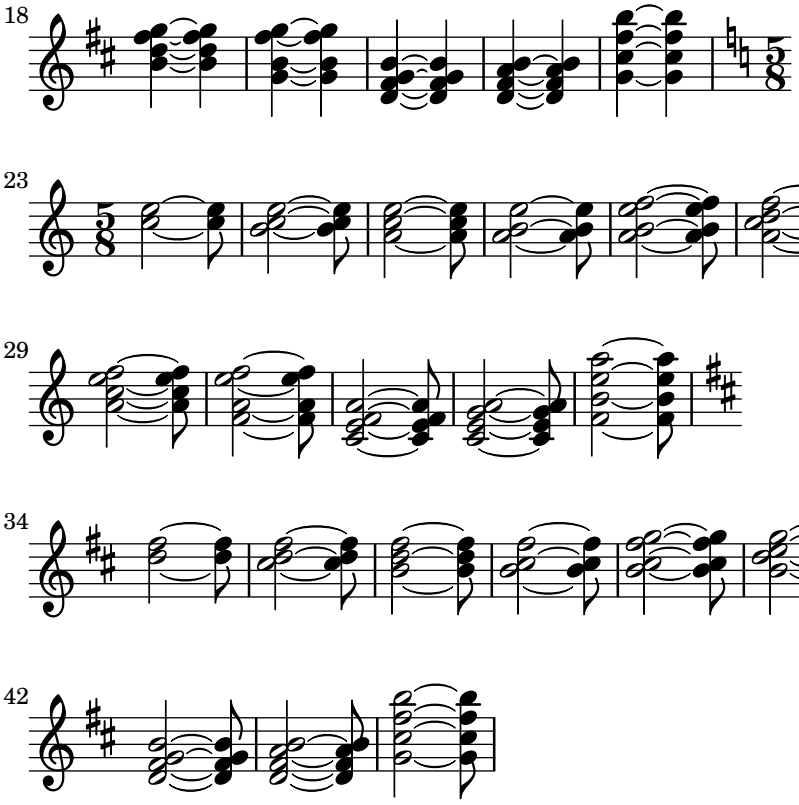


In chords, ties keep closer to the note head vertically, but never collide with heads or stems. Seconds are formatted up/down; the rest of the ties are positioned according to their vertical position.

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.
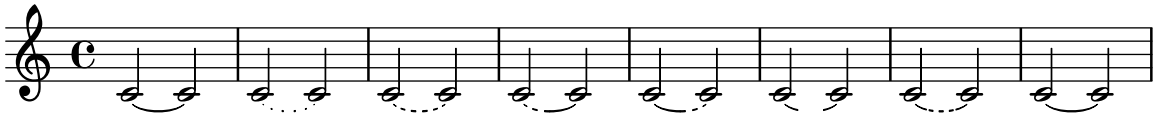
`tie-chord.ly`





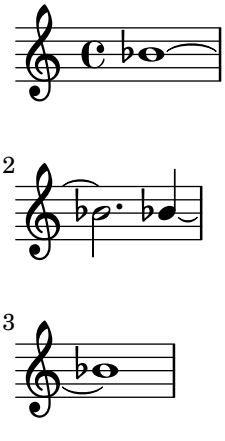
The appearance of ties may be changed from solid to dotted or dashed.

`tie-dash.ly`
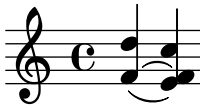


In the single tie case, broken ties peek across line boundaries to determine which direction to take.

`tie-direction-broken.ly`
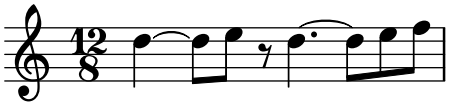


Tie directions can be set with `_` and `^`. This makes correction in complex chords easier.

`tie-direction-manual.ly`
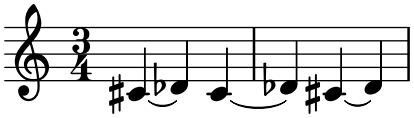


Ties avoid collisions with dots.

`tie-dot.ly`
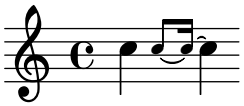


LilyPond should accept a tie between notes which are enharmonically identical.

`tie-enharmonic.ly`
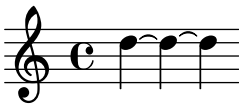


Tying a grace to a following grace or main note works.

`tie-grace.ly`



If using exact values (this is, either integers or rational values like '`(/ 4 5)`'), `staff-position` is used to vertically tune the tie for staff line avoidance. If using an inexact value like a floating point number, it is taken as the vertical location.

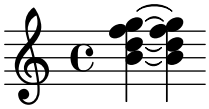
`tie-manual-vertical-tune.ly`



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

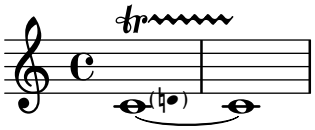
You can leave a Tie alone by introducing a non-pair value (eg. `#t`) in the `tie-configuration` list.

`tie-manual.ly`
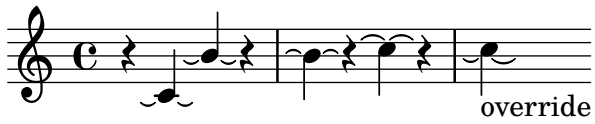


The pitch of a pitched trill should not trigger a warning for unterminated ties.

`tie-pitched-trill.ly`



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

`tie-semi-single.ly`
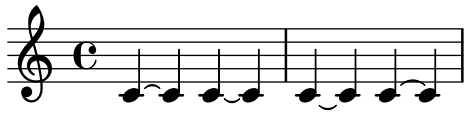


Tie directions are also scored. In hairy configurations, the default rule for tie directions is overruled.

`tie-single-chord.ly`



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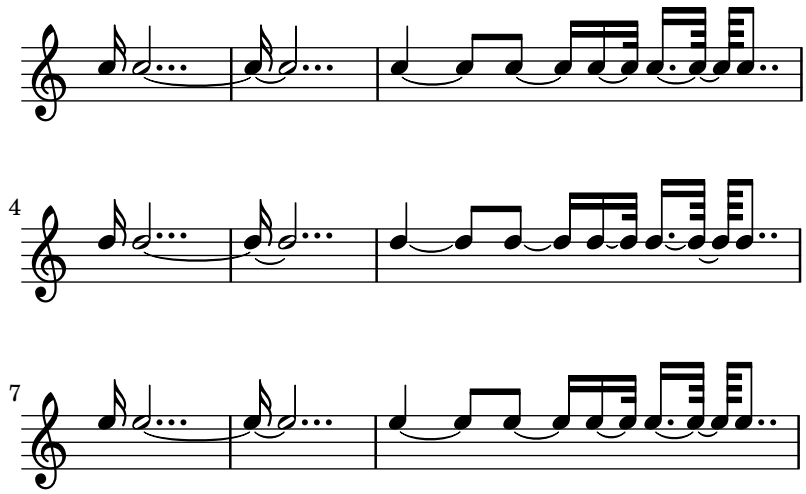
`tie-single-manual.ly`



Formatting for isolated ties.

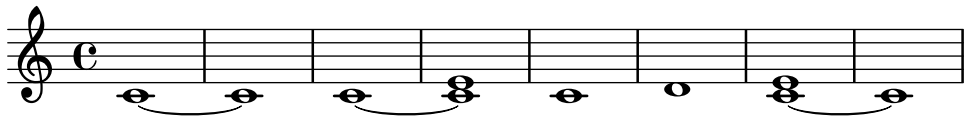
- short ties are in spaces
- long ties cross staff lines
- ties avoid flags of left stems.
- ties avoid dots of left notes.
- short ties are vertically centered in the space, as well those that otherwise don't fit in a space
- extremely short ties are put over the noteheads, instead of between.

`tie-single.ly`



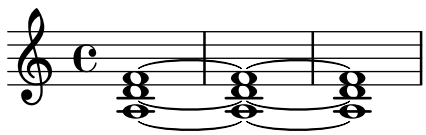
When a tie is followed only by unmatching notes and the tie cannot be created, lilypond prints out a warning unless `tieWaitForNote` is set.

`tie-unterminated.ly`



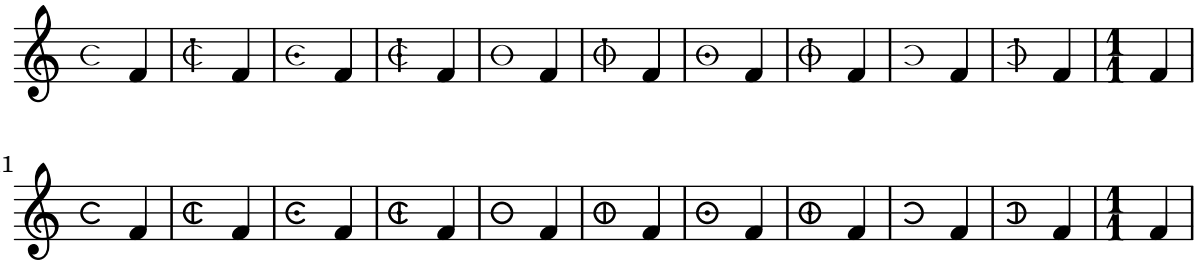
For whole notes, the inside ties do not cross the center of the note head, horizontally.

`tie-whole.ly`
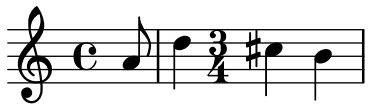


This test covers the mensural and neomensural time signature styles.

`time-signature-mensural.ly`



Mid-measure time signature changes not accompanied by \partial generate warnings.

`time-signature-midmeasure-warning.ly`
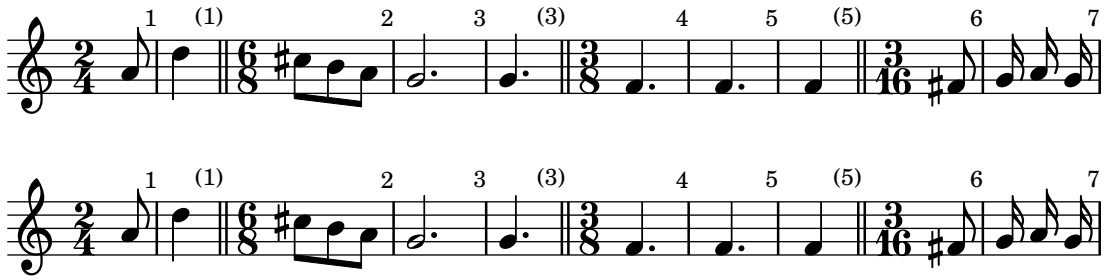


Mid-measure time signature changes must be accompanied by \partial.

In this example, no bar numbers should be omitted or repeated, and all double bar lines should have parenthesized bar numbers consistent with the single bar lines. Both scores should look identical.
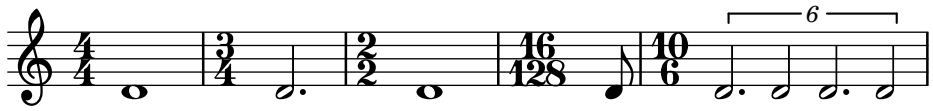
- \time 2/4 occurs at a negative position
- \time 6/8 occurs at a position less than the new measure length
- \time 3/8 occurs at a position equal to the new measure length
- \time 3/16 occurs at a position greater than the new measure length

`time-signature-midmeasure.ly`



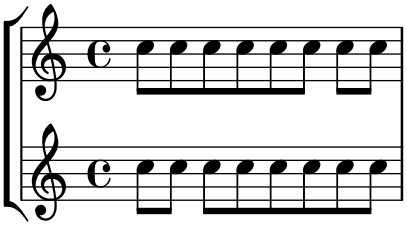
The numbered time signature style prints a fraction.

`time-signature-numbered.ly`
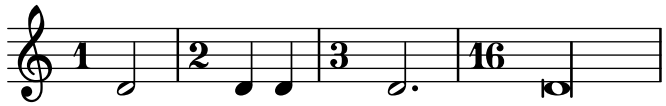


Default values for time signature settings can vary by staff if the `Timing_translator` and `Default_bar_line_engraver` are moved from `Score` to `Staff`. In this case, the upper staff should be beamed 3/4, 1/4. The lower staff should be beamed 1/4, 3/4.

`time-signature-settings-by-staff.ly`



The single-digit time signature style prints the numerator only.

`time-signature-single-digit.ly`



The input representation is generic, and may be translated to XML.

`to-xml.ly`



A table of contents is included using `\markuplist \table-of-contents`. The toc items are added with the `\tocItem` command. In the PDF backend, the toc items are linked to the corresponding pages.

`toc.ly`

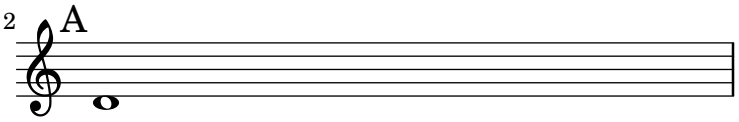
Table of Contents

The first score 2
Mark A 3
The second score 4

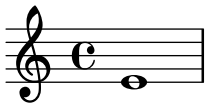
2



3



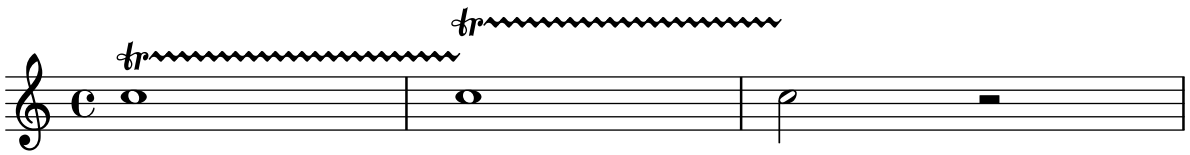
4
Second score



Music engraving by LilyPond 2.21.1—www.lilypond.org

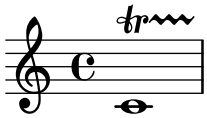
Consecutive trill spans work without explicit `\stopTrillSpan` commands, since successive trill spanners will automatically become the right bound of the previous trill.

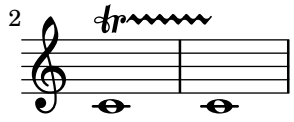
`trill-spanner-auto-stop.ly`



A TrillSpanner crossing a line break should restart exactly above the first note on the new line.

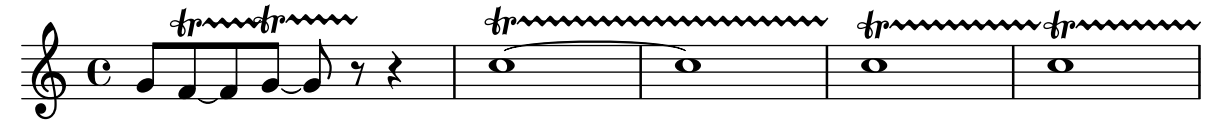
`trill-spanner-broken.ly`
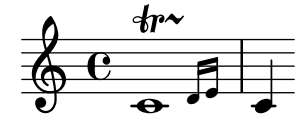




Chained trills end at the next trill or barline. Collisions can be prevented by overriding `bound-details`.

`trill-spanner-chained.ly`
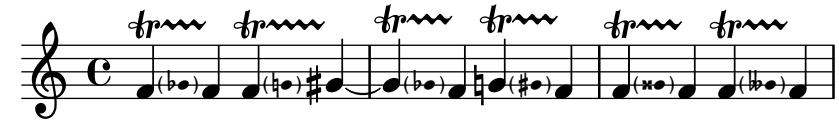


Trill spanner can end on a grace note

`trill-spanner-grace.ly`



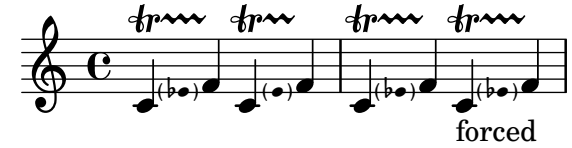
Pitched trills on consecutive notes with the same name and octave should not lose accidentals; in the following example, accidentals should be visible for all trill-pitches.

`trill-spanner-pitched-consecutive.ly`



Pitched trill accidentals can be forced.

`trill-spanner-pitched-forced.ly`



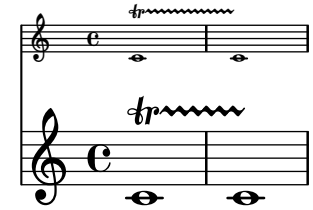
Pitched trills are denoted by a small note head in parentheses following the main note. This note head is properly ledgered, and parentheses include the accidental.

`trill-spanner-pitched.ly`



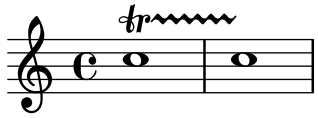
The horizontal position of the beginning of a trill spanner is positioned correctly relative to the note head it is attached to, even if scaled to a smaller size.

`trill-spanner-scaled.ly`



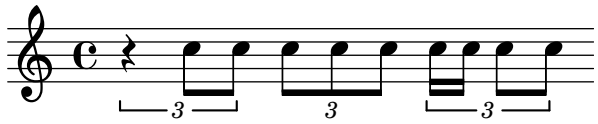
The trill symbol and the wavy line are neatly aligned: the wavy line should appear to come from the crook of the r

`trill-spanner.ly`



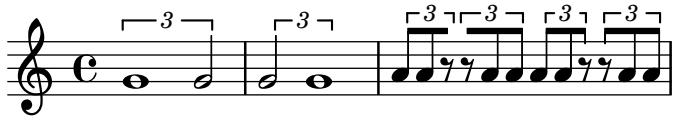
In combination with a beam, the bracket of the tuplet bracket is removed. This only happens if there is one beam, as long as the bracket.

`tuplet-beam.ly`
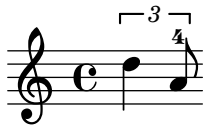


Tuplet brackets should align to visible or transparent stems only. For stemless notes or rests they should span the whole note width.

`tuplet-bracket-X-positions.ly`



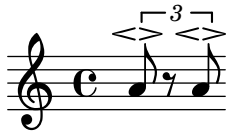
`TupletBracket` grobs avoid `Fingering` grobs.

`tuplet-bracket-avoid-fingering.ly`



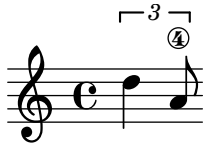
Tuplet brackets avoid scripts by default.

`tuplet-bracket-avoid-scripts.ly`



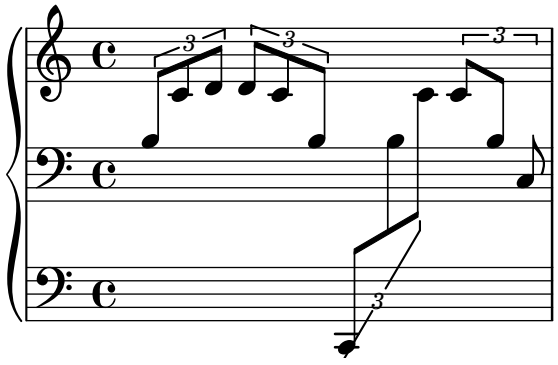
`TupletBracket` grobs avoid `StringNumber` grobs.

`tuplet-bracket-avoid-string-number.ly`



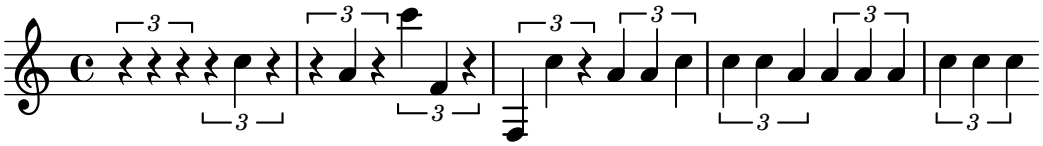
Cross-staff tuplets are drawn correctly, even across multiple staves.

`tuplet-bracket-cross-staff.ly`



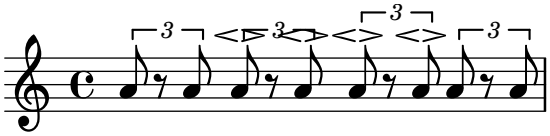
The direction of tuplet brackets is the direction of the majority of the stems under the bracket, with ties going to UP.

`tuplet-bracket-direction.ly`



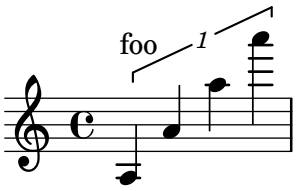
Tuplet brackets' outside staff priority can be set. Brackets, by default, carry their numbers with them.

`tuplet-bracket-outside-staff-priority.ly`



Tuplet brackets do not push objects with outside-staff-priority too high.

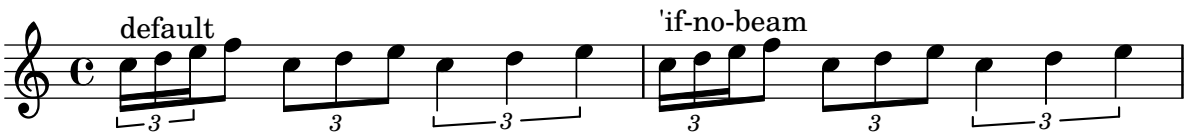
`tuplet-bracket-vertical-skylines.ly`
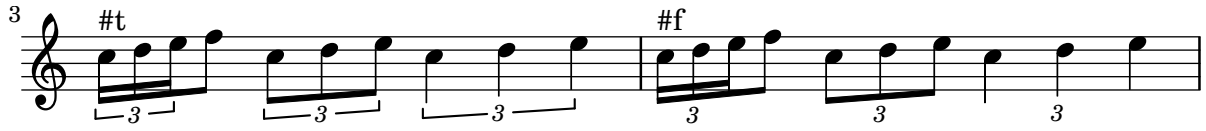


The default behavior of tuplet-bracket visibility is to print a bracket unless there is a beam of the same length as the tuplet. Overriding `'bracket-visibility` changes the bracket visibility as follows:

- `#t` (always print a bracket)
- `#f` (never print a bracket)
- `'if-no-beam` (only print a bracket if there is no beam)

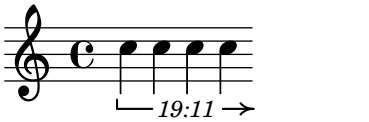
`tuplet-bracket-visibility.ly`
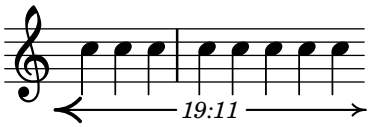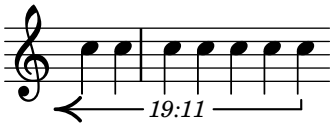




Broken tuplets are adorned with little arrows. The arrows come from the `edge-text` property, and thus be replaced with larger glyphs or other text.

`tuplet-broken.ly`

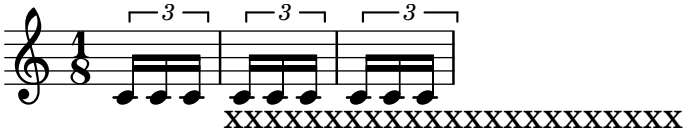






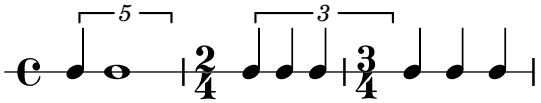
With `full-length-to-extent`, the extent of the attaching column for a full-length tuplet bracket can be ignored.

`tuplet-full-length-extent.ly`
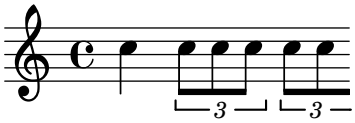


tuplet can be made to run to prefatory matter or the next note, by setting `tupletFullLengthNote`.
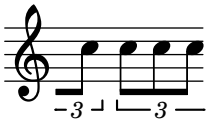
`tuplet-full-length-note.ly`
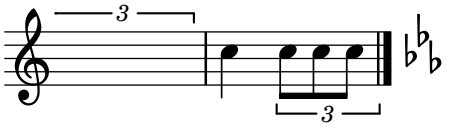


If `tupletFullLength` is set, tuplets end at the start of the next non-tuplet note.

`tuplet-full-length.ly`
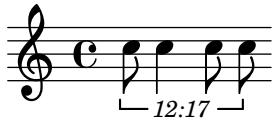






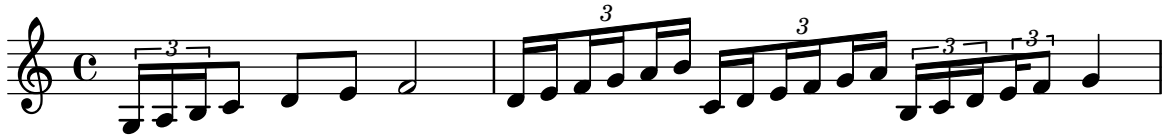
The size of the tuplet bracket gap is adjusted to the width of the text.

`tuplet-gap.ly`
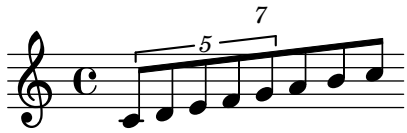


Overlong tuplet span specifications are reduced to the actual length.

`tuplet-long-spanner.ly`
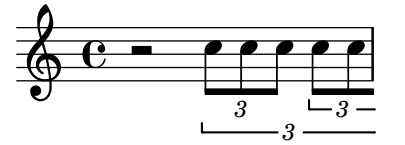


Nested tuplets do collision resolution, also when they span beams.

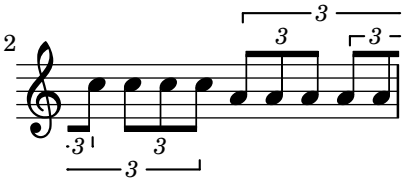
`tuplet-nest-beam.ly`

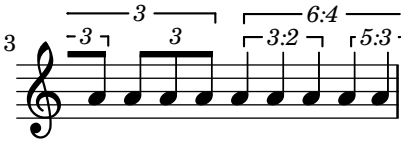


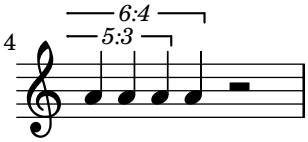
Broken nested tuplets avoid each other correctly.

`tuplet-nest-broken.ly`



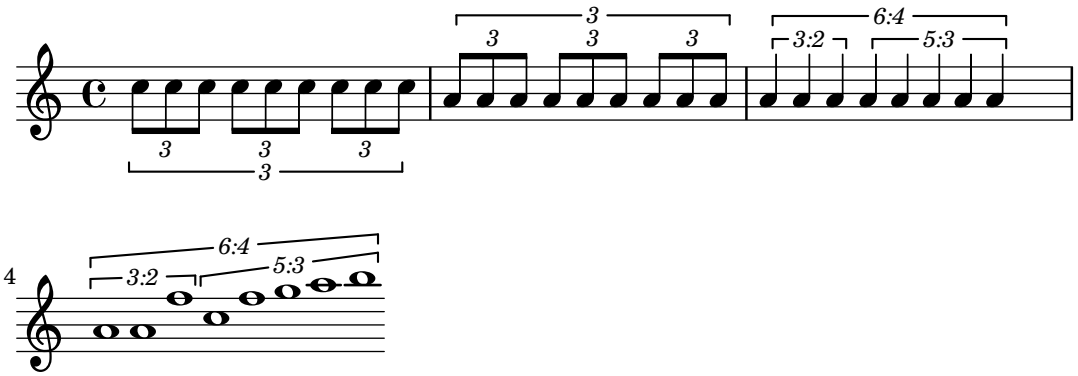






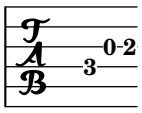
Tuplets may be nested.

tuplet-nest.ly



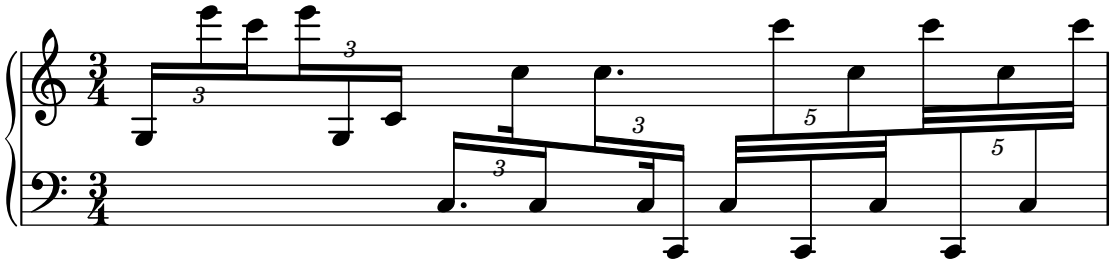
Removing Stem_engraver doesn't cause crashes.

tuplet-no-stems.ly
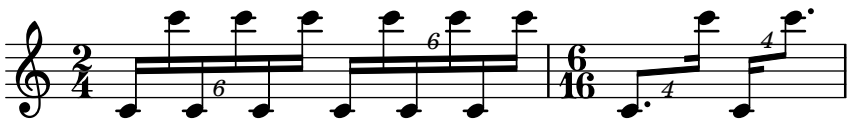


Tuplet numbers are positioned correctly on kneed French-style beams.

tuplet-number-french-kneed-beams.ly



In tuplets with an even number of stems, the number may be placed on either side of the beam when the central stems point in different directions. The exception to this is when there is a fractional beam on one of the central stems, in which case the number is placed opposite the partial beam.
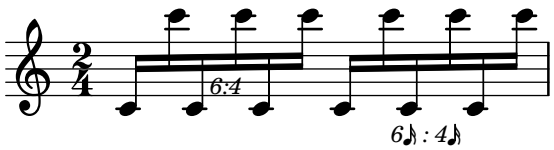
tuplet-number-kneed-beam-even-stem-count.ly



Tuplet numbers are placed next to the beam unless there is insufficient horizontal space for them, in which case bracket-based positioning is used and a programming error is issued.

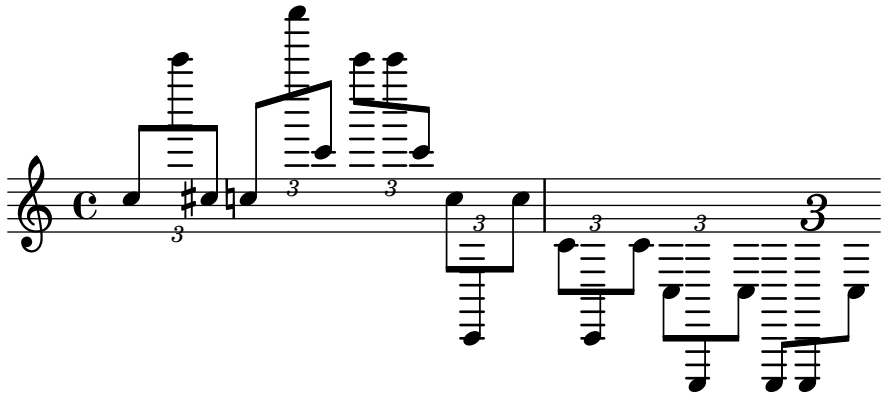
The first tuplet number should be between stems; the second should be below the noteheads.

tuplet-number-kneed-beam-horizontal-fit.ly



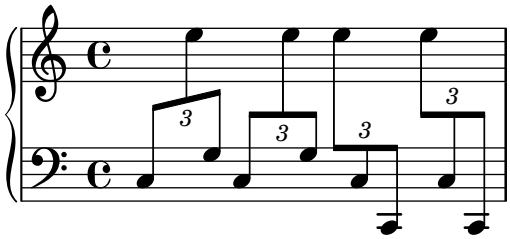
A tuplet number associated with a kneed beam is not placed between beam and staff where it may collide with ledger lines.

`tuplet-number-kneed-beam-ledger-lines.ly`



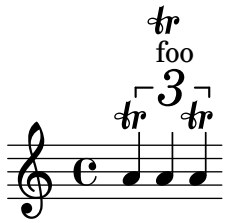
Tuplet numbers are placed next to kneed beams when `Beam.positions` is overridden.

`tuplet-number-kneed-beam-positions.ly`



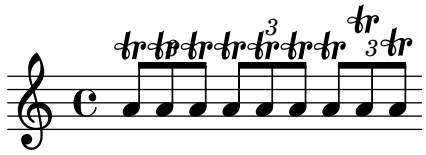
Grobs whose parents have `outside-staff-priority` set should figure into the vertical skyline of the `VerticalAxisGroup` such that grobs with a higher `outside-staff-priority` are correctly positioned above them.

`tuplet-number-outside-staff-positioning.ly`



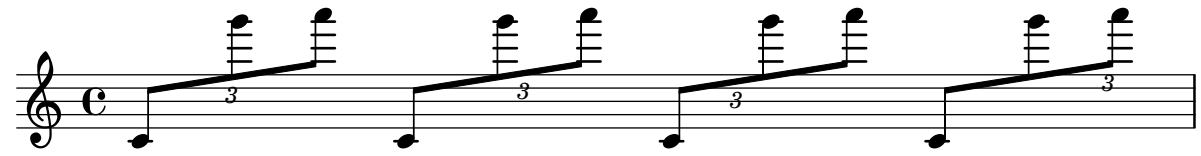
Tuplet numbers' outside staff priority can be set.

`tuplet-number-outside-staff-priority.ly`



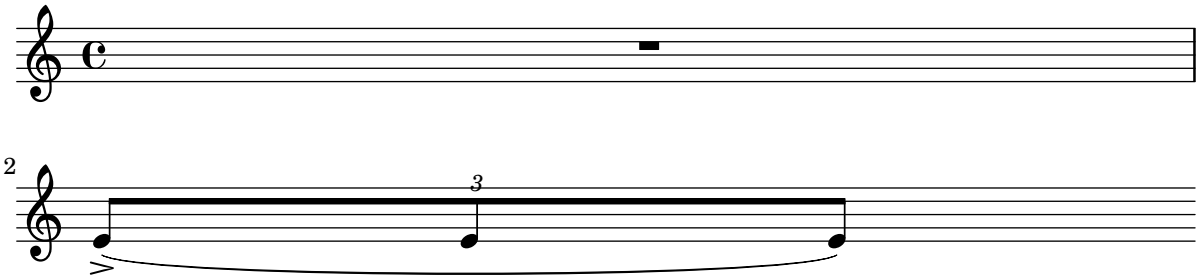
Tuplet numbers will maintain a constant distance from kneed beams when offset horizontally.

`tuplet-number-shift-along-kneed-beam.ly`



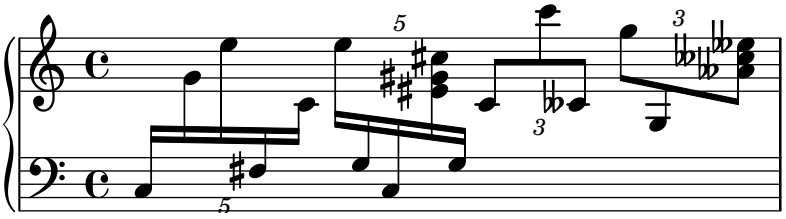
Tuplet number position is correct when slurs and scripts are present.

`tuplet-number-slur-script.ly`



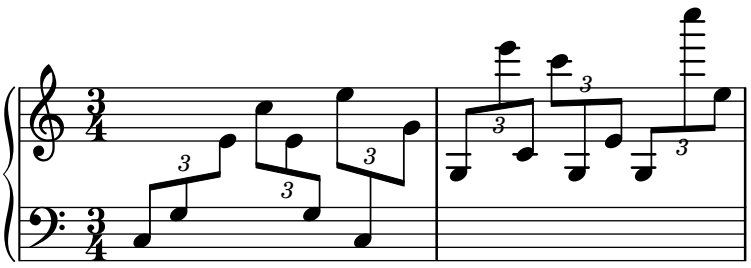
Tuplet numbers associated with kneed beams will avoid accidentals.

`tuplet-numbers-kneed-beams-accidentals.ly`



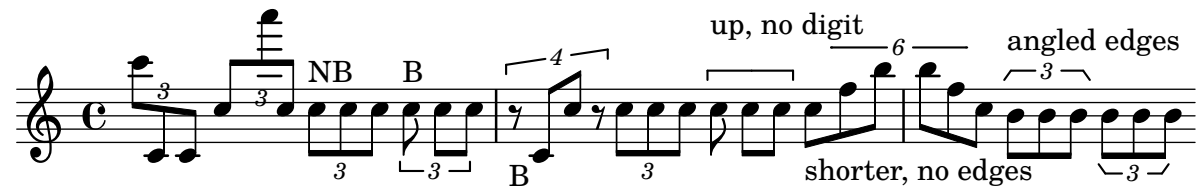
Tuplet numbers are positioned next to kneed beams.

`tuplet-numbers-kneed-beams.ly`



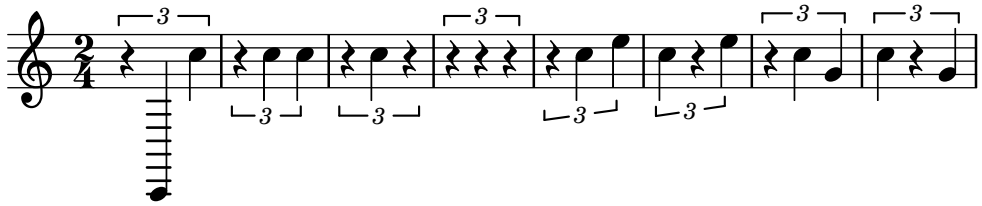
Tuplet bracket formatting supports numerous options, for instance, bracketed (B) and non-bracketed (NB).

`tuplet-properties.ly`



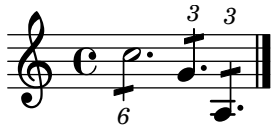
Tuplets may contain rests.

`tuplet-rest.ly`



Show tuplet numbers also on single-note tuplets (otherwise the timing would look messed up!), but don't show a bracket. Make sure that tuplets without any notes don't show any number, either.

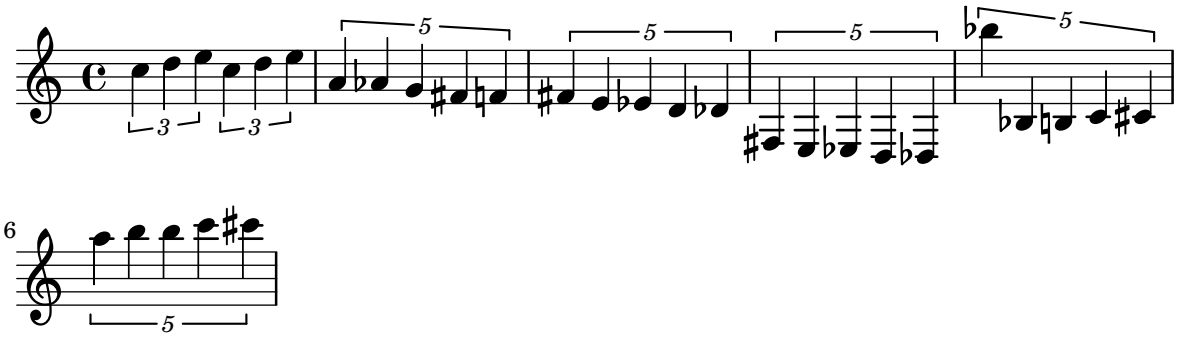
`tuplet-single-note.ly`



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
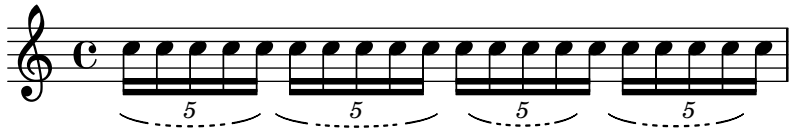
The bracket direction is determined by the dominating stem direction.

`tuplet-slope.ly`



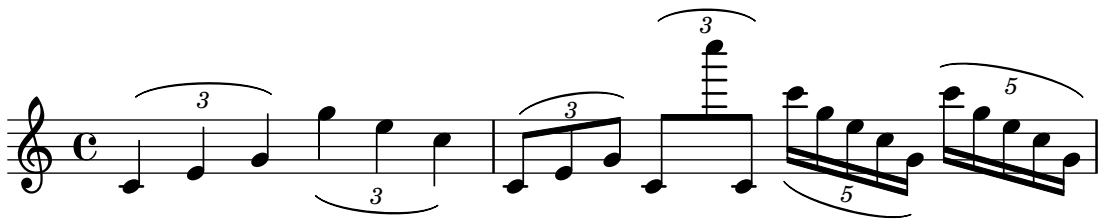
Tuplet slurs may be tweaked through the `shorten-pair` and `dash-definition` properties.

`tuplet-slur-tweaks.ly`



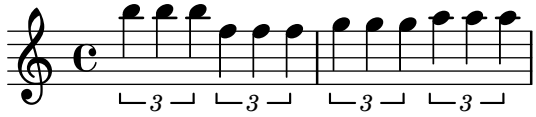
Slurs may be used instead of brackets for tuplets through the `tuplet-slur` property of `TupletBracket`. Rules for visibility are the same as for regular brackets, so `bracket-visibility` should be set to `#t` if the slur is desired for beamed groups.

`tuplet-slurs.ly`



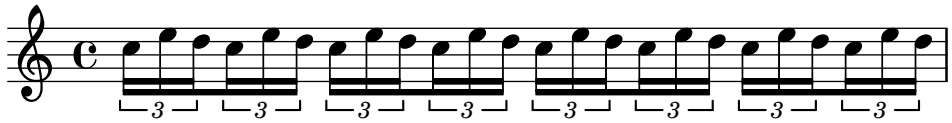
Horizontal tuplet brackets are shifted vertically to avoid staff line collisions.

`tuplet-staffline-collision.ly`
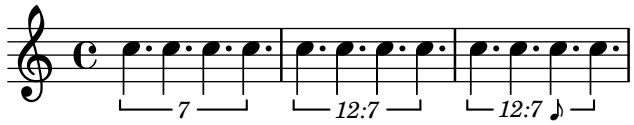


Subdivision works properly for tuplets.

`tuplet-subdivision.ly`
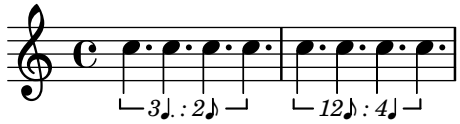


Non-standard tuplet texts: Printing other tuplet fractions than the ones actually assigned.

`tuplet-text-different-numbers.ly`
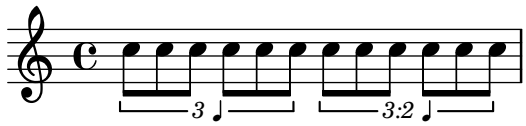


Non-standard tuplet texts: Printing a tuplet fraction with note durations assigned to both the denominator and the numerator.

`tuplet-text-fraction-with-notes.ly`



Non-standard tuplet texts: Appending a note value to the normal text and to the fraction text.
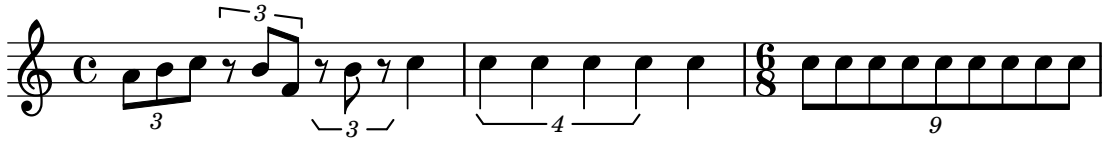
`tuplet-text-note-appended.ly`



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.

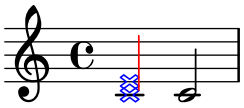
The bracket stops at the end of the stems, if the stems have the same direction as the bracket. The endings can be adjusted with `bracket-flare`.

`tuplets.ly`
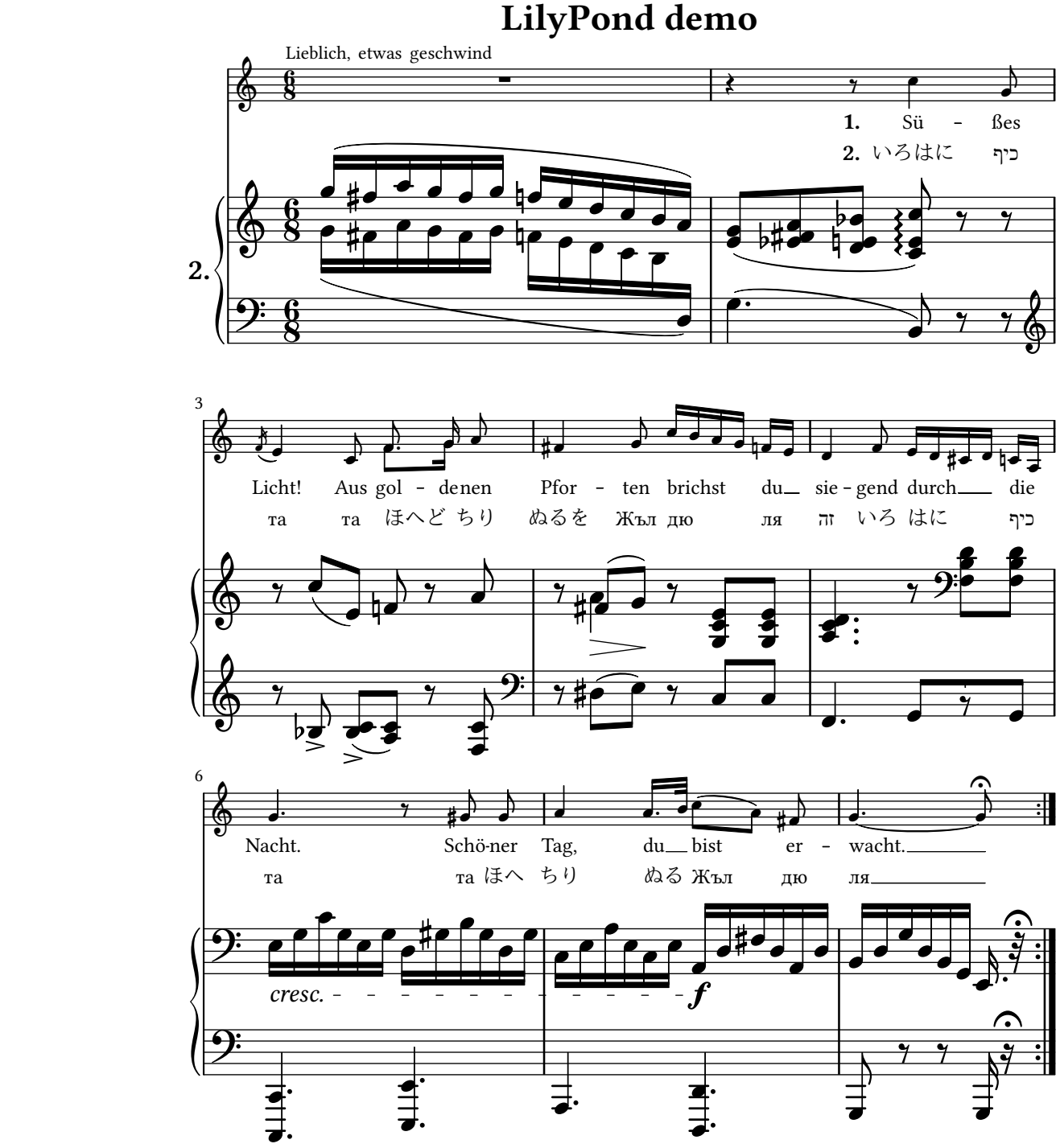


Overrides can be the target of a `\propertyTweak`, with the tweaks accumulating as override. The main application is for stacking commands implemented in terms of `\propertyTweak`. This example should show the starting chord with blue, cross-styled note heads and a red stem.

`tweaks-as-overrides.ly`



heavily mutilated Edition Peters Morgenlied by Schubert

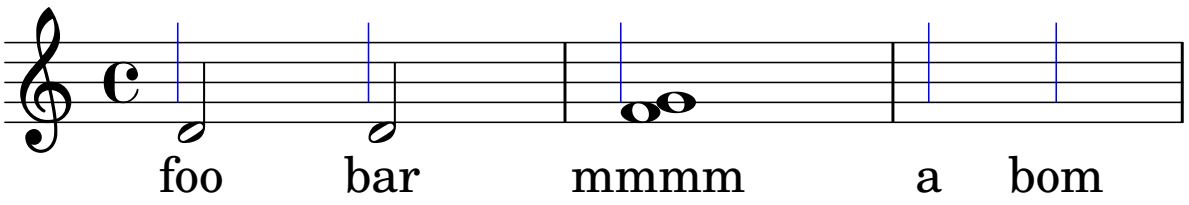
typography-demo.ly



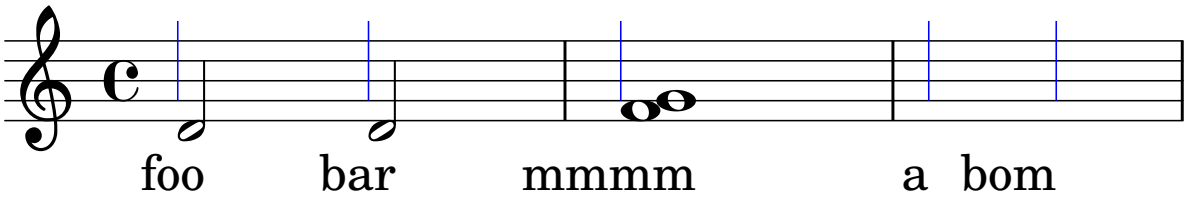
Lyrics without an `associatedVoice` should align properly. If there are notes in the `PaperColumn`, they should align to them, and when there are no notes, they should align relative to the `PaperColumn` itself (represented with blue `GridLines` here)

`unassociated-lyrics-alignment.ly`

default (centered):
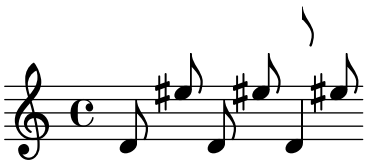


right-aligned:
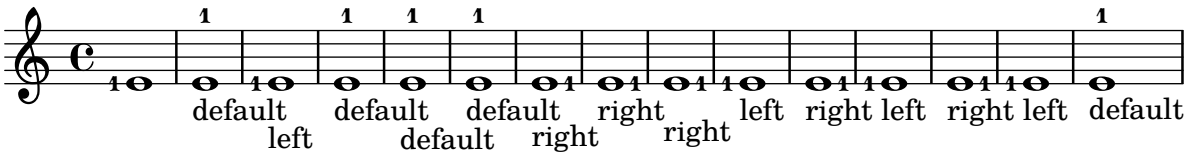


unpure-pure containers take two arguments: an unpure property and a pure property. The pure property is evaluated (and cached) for all pure calculations, and the unpure is evaluated for all unpure calculations. In this regtest, there are three groups of two eighth notes. In the first group, the second note should move to accommodate the flag, whereas it should not in the second group because it registers the flag as being higher. The flag, however, remains at the Y-offset dictated by ly:flag::calc-y-offset. In the third set of two 8th notes, the flag should be pushed up to a Y-offset of 8.

`unpure-pure-container.ly`



`\once \unset` should change a context property value for just one timestep and then return to the previous value.

`unset-once.ly`
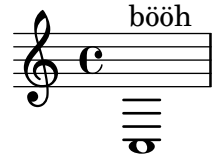


words in mixed font in a single string are separated by spaces as in the input string. Here a Russian word followed by a roman word.

`utf-8-mixed-text.ly`

Здравствуйте Hallo
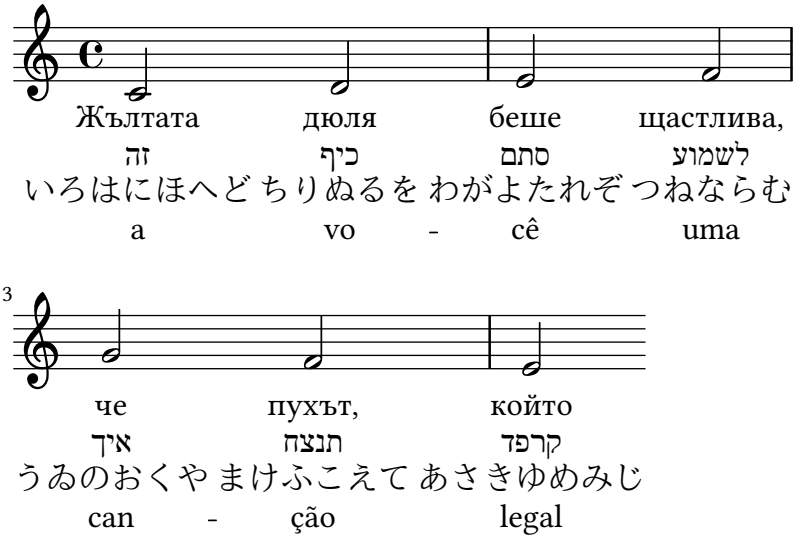
In GUILE v2, embedded Scheme can contain UTF-8 strings and identifiers. Here, identifer bääh contains music with the text "bööh"

`utf-8-scheme.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.

`utf-8.ly`

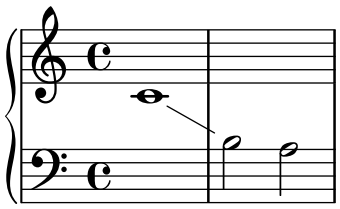


This does not produce typeset output but checks that `\version` statements in included files do not inhibit the warning in the main file when a `\version` statement is missing there.

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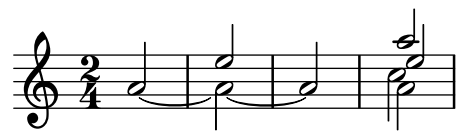
Whenever a voice switches to another staff a line connecting the notes can be printed automatically. This is enabled if the property `followVoice` is set to true.

`voice-follower.ly`



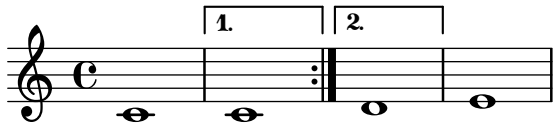
The `\voices` command can be used for continuing voices and changing the order of `\voiceOne`...`\voiceFour` style overrides.

`voices-command.ly`



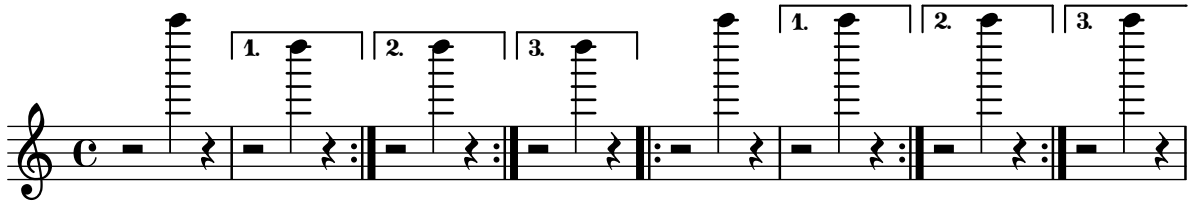
Volta bracket end hooks can be added for other bar line types.

`volta-bracket-add-volta-hook.ly`



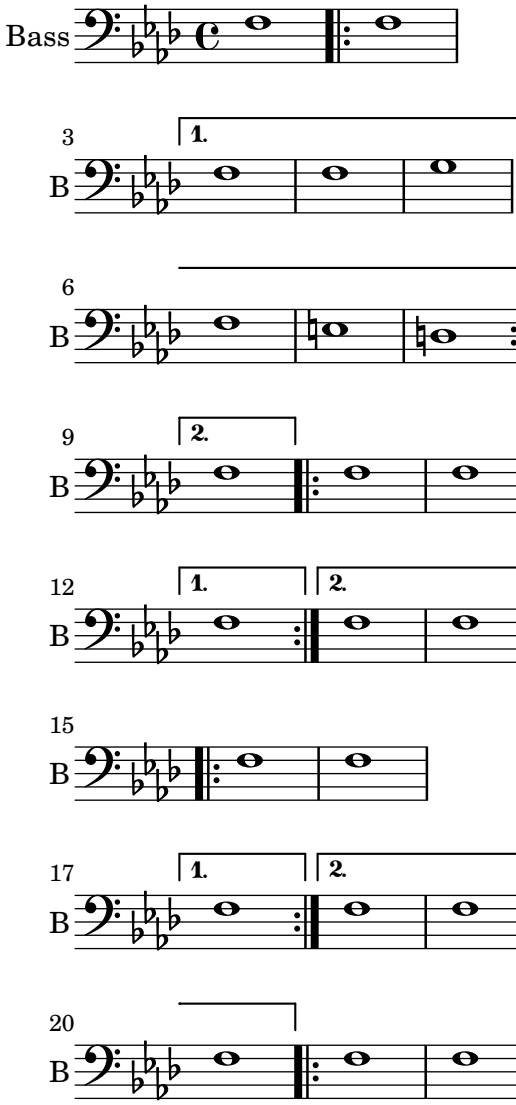
Volta brackets are vertically fit to objects below them.

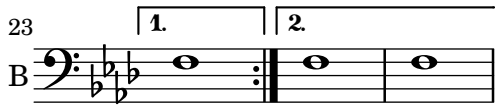
`volta-bracket-vertical-skylines.ly`



Broken volta spanners behave correctly at their left edge in all cases.
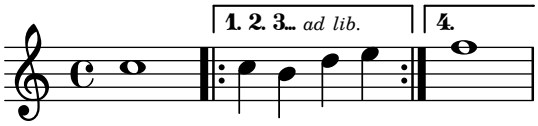
`volta-broken-left-edge.ly`





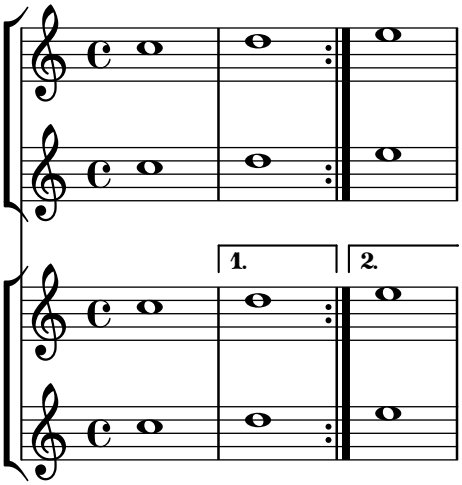
Volte using `repeatCommands` can have markup text.

`volta-markup-text.ly`



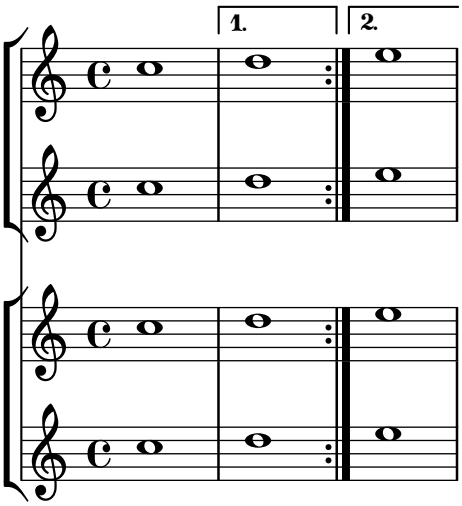
By putting Volta_engraver in a staff context, one can get volta brackets on staves other than the topmost one.

`volta-multi-staff-inner-staff.ly`



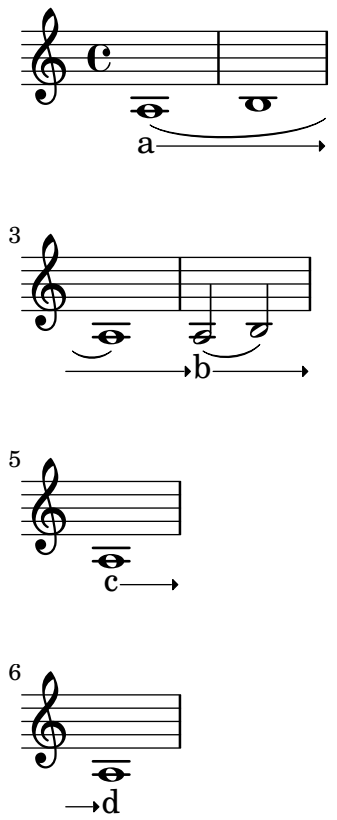
By default, the volta brackets appear only in the topmost staff.

`volta-multi-staff.ly`
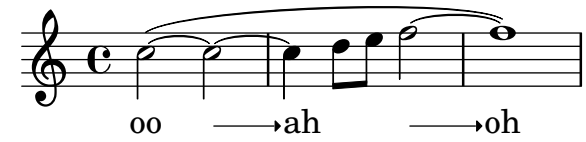


A vowel transition runs to the end of the line if it continues on the next line, or if the next lyric syllable is at the first note on the next line. Transition arrows are printed at the beginning of the line only when they go past the first note, or when property `after-line-breaking` is `#t`.

vowel-transition-broken.ly



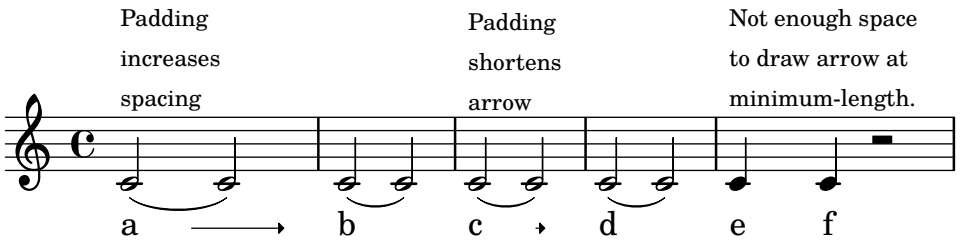
The length of the transition between one syllable and the next is indicated by the length of the arrow, which may not start immediately after a new syllable.

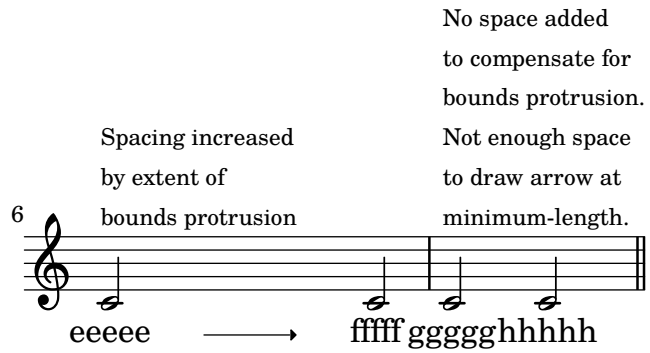
vowel-transition-delayed-start.ly

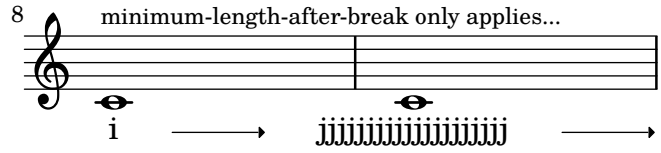


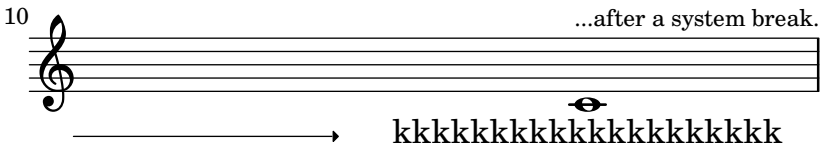
For vowel transitions, `minimum-length` refers to the drawn length of the arrow. The protrusion of the syllables and padding is in effect added to `minimum-length` for spacing. This default behaviour can be changed by overriding `springs-and-rods`, which may cause the transition arrow not to be drawn if there is insufficient space (rather than adding the space necessary to draw it at `minimum-length`). `minimum-length-after-break` controls the minimum length of the segment following a system break.

vowel-transition-minimum-length.ly
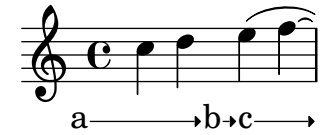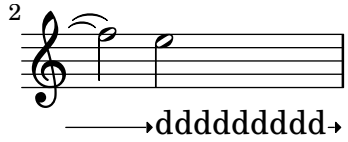




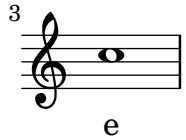




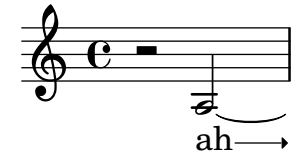
When a syllable is horizontally offset, the arrow should adjust accordingly.
`vowel-transition-offset-syllable.ly`
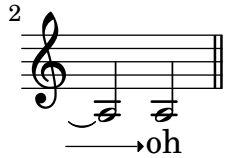


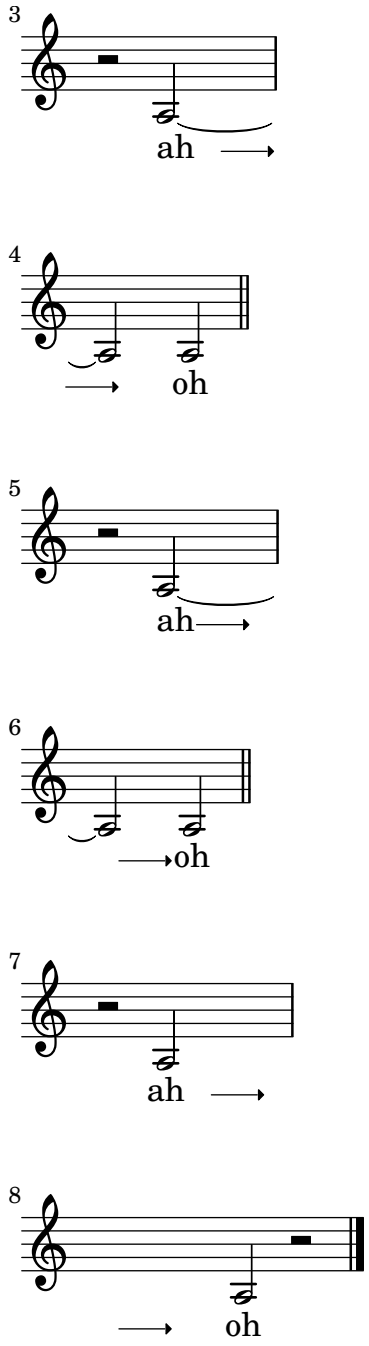




For vowel transitions, left/right padding are independent of left-broken/right-broken padding.
`vowel-transition-padding-broken.ly`

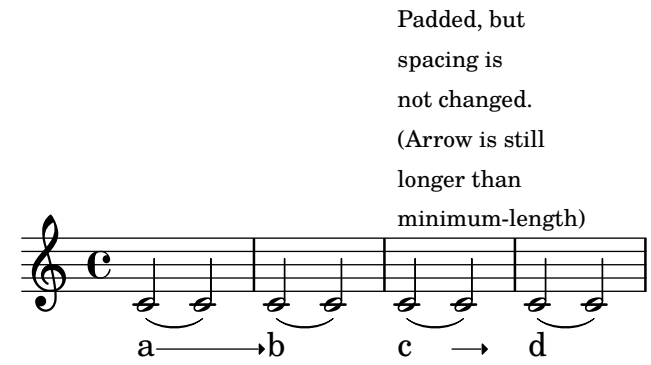


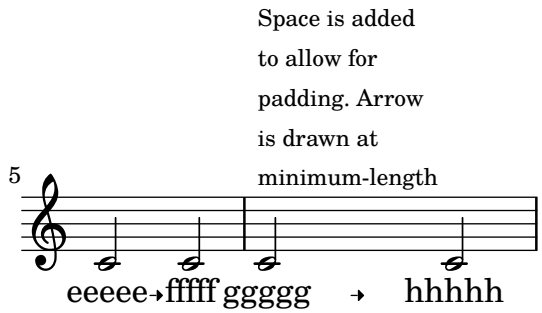




Padding does not cause `VowelTransitions` to become shorter than `minimum-length`. Instead, space is added if necessary leaving the arrow at `minimum-length`.

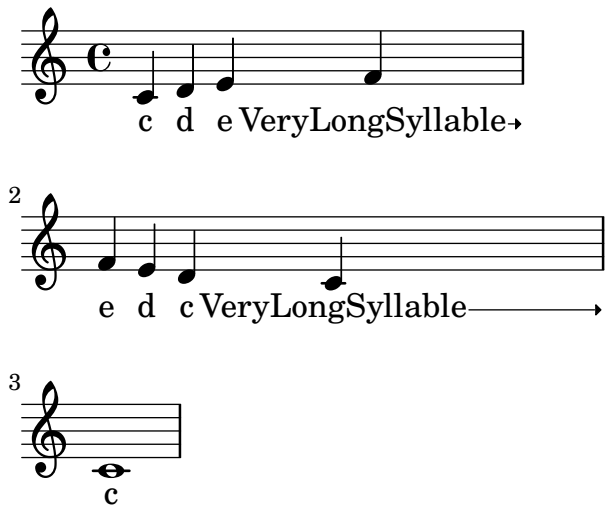
`vowel-transition-padding.ly`





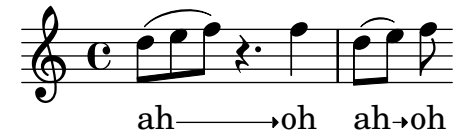
Vowel transition arrows are always drawn, but they do not protrude into the margin. Instead, space is added so that the arrow can be drawn at `minimum-length`.

`vowel-transition-right-margin.ly`



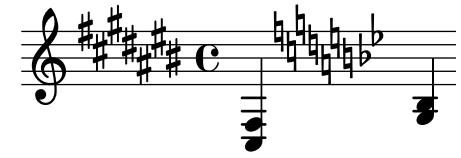
A vowel transition arrow may span several notes. The arrow may extend past a rest, but not past the next lyric syllable.

`vowel-transition.ly`
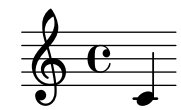


If you specify two different key signatures at one point, a warning is printed.

`warn-conflicting-key-signatures.ly`



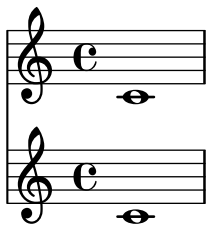
If a warning is expected, but not triggered, print out a warning about this fact. This will be used to detect missing warnings in our regtests.

`warn-expected-warning-missing.ly`
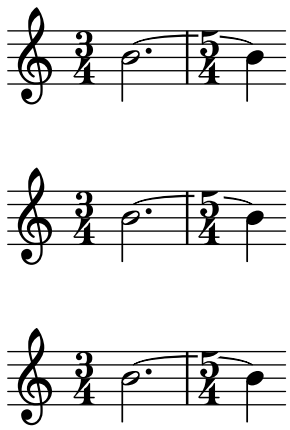


A warning is printed if a dynamic spanner is unterminated.

warn-unterminated-span-dynamic.ly
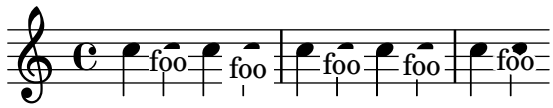


If the 'whiteout property of a grob is set to a number or #t, that part of all objects in lower layers which falls under the extent of the grob's whiteout area is whited out. Here the TimeSignature whites out the Tie but not the StaffSymbol.

whiteout-lower-layers.ly



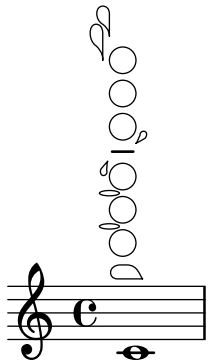
The whiteout command underlays a white background under a markup. The shape is determined by `whiteout-style`. The default is `box` which produces a rectangle. `rounded-box` produces a rounded rectangle. `outline` approximates the outline of the markup.

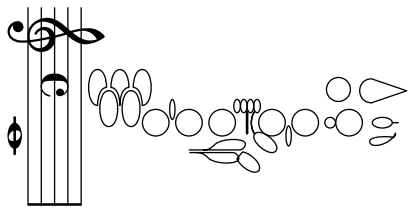
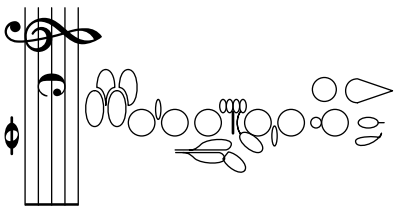
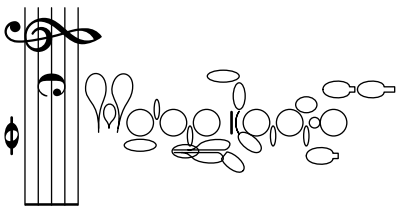
whiteout.ly

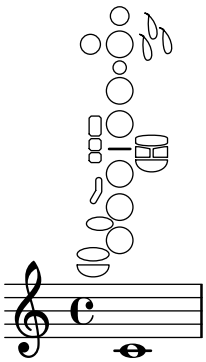


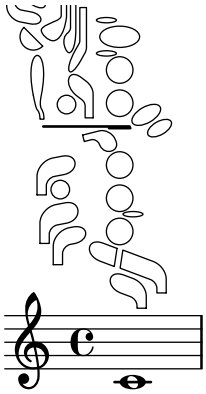
Empty woodwind diagrams for all instruments in woodwind-diagrams.scm.

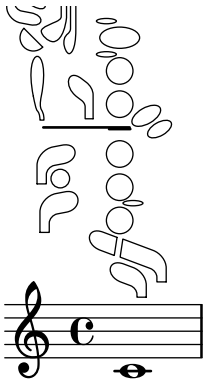
woodwind-diagrams-empty.ly







Lists all possible keys for all instruments in woodwind-diagrams.scm
`woodwind-diagrams-key-lists.ly`
Setting staff-space to 0 does not cause a segmentation fault.
`zero-staff-space.ly`

