Neuerungen von Version 2.20 gegenüber 2.18

Neuigkeiten in der Eingabe

Tonhöhen

- Ausgeschriebene Tonhöhen in der englischen Eingabesprache mit `sharp` oder `flat` im Namen benötigen jetzt einen Bindestrich.

  ```
  \key a-flat \major
  ```

  statt:

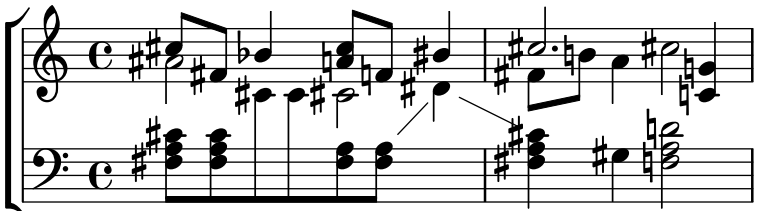
  ```
  \key aflat \major
  ```

  Tonhöhen mit doppelten Vorzeichen werden hierbei nur mit einem Bindestrich versehen. Für die englische Variante des deutschen oder niederländischen `cisis` etwa:

  ```
  \key c-sharpsharp \major
  ```

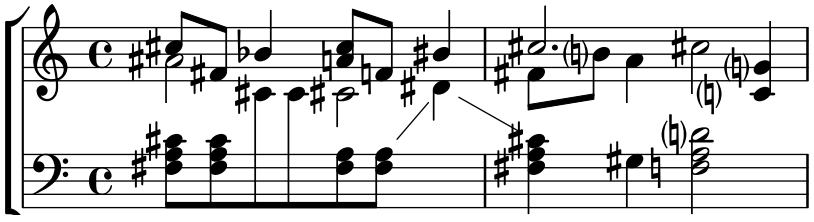
- Vorzeichenregeln können jetzt auch über einen `ChoirStaff`-Kontext übergreifend definiert werden.
- Zwei neue Vorzeichenregeln kombinieren die Eigenschaften von `modern-voice`, `piano` und ihren Äquivalenten:

  `choral`

  Die Vorzeichenregel `choral` ist jetzt Standardvorgabe für `ChoirStaff`.
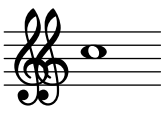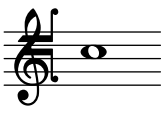
  `choral-cautionary`

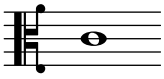
  `choral-cautionary` entspricht `choral`, setzt aber unterstützende Vorzeichen in Behelfsform.

  Vgl. Abschnitt “Automatische Versetzungszeichen” in *Notationsreferenz*.

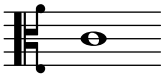
- Vier neue Notenschlüssel stehen zur Verfügung; ‚GG‘ (Doppel-G), ‚Tenor G‘, ‚varC‘ und Verwandte, sowie ‚Varpercussion‘:

Example	**Output**	**Example**	**Output**
`\clef GG`		`\clef tenorG`	
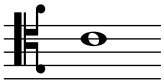
`\clef varC`



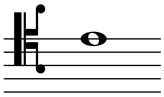
`\clef altovarC`
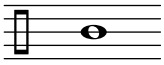


`\clef tenorvarC`



`\clef baritonevarC`
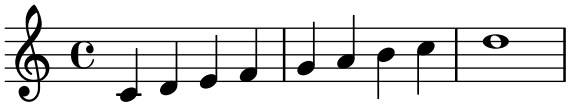


`\clef varpercussion`



Vgl. Abschnitt “Notenschlüssel-Glyphen” in *Notationsreferenz*.
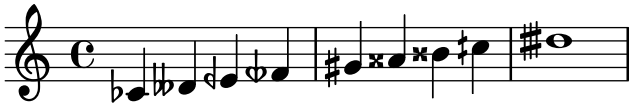
- Französische Notennamen sind nun als eigenständige Notensprache statt als Variante der italienischen verfügbar. Das deutsche *d* kann hierbei wahlweise `re` oder `ré` geschrieben werden.

```
\language "français"
do ré mi fa | sol la si do | ré1
```



Doppelkreuze werden durch Anhängen von `x` eingegeben:

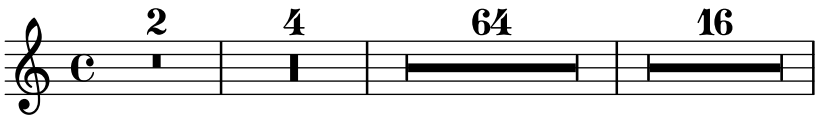
```
\language "français"
dob, rebb misb fabsb | sold ladd six dosd | rédsd1
```



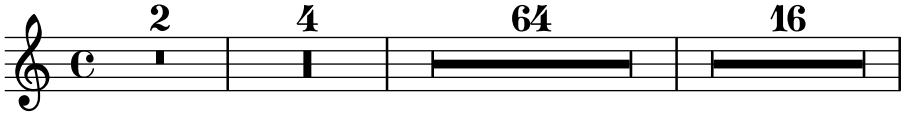
Neuheiten bei Tonlängen

- Mehrtaktpausen werden mit einer zu ihrer Dauer passenden Breite gesetzt, wobei `MultiMeasureRest.space-increment` das Verhältnis bestimmt. Die Standardvorgabe dieses Wertes beträgt `2.0`.

```
\compressEmptyMeasures
R1*2 R1*4 R1*64 R1*16
```
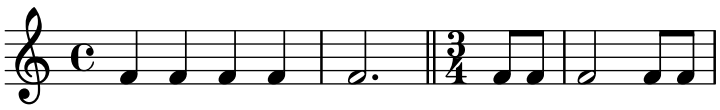



```
\compressEmptyMeasures
\override Staff.MultiMeasureRest.space-increment = 2.5
R1*2 R1*4 R1*64 R1*16
```



- Das Verhalten von Auftakten mit `\partial` in parallelen Stimmen oder anderen Kontexten ist deutlich verbessert worden.

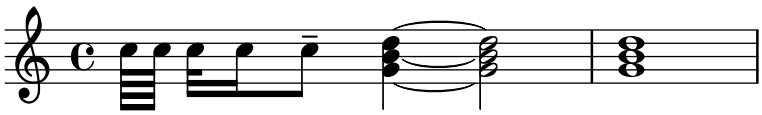
- Es ist nunmehr möglich, das Taktmaß im laufenden Takt durch kombinierte Verwendung von `\time` und `\partial` zu verändern.

```
f f f f | f2. \bar "||"
\time 3/4 \partial 4
f8 8 | f2 f8 f |
```

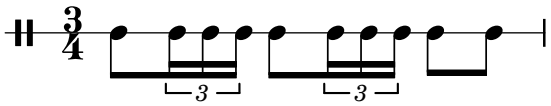


- Alleinstehende Notenlängen in der Eingabe stehen nunmehr für tonhöhenlose Noten. Die zur Verwendung kommende Tonhöhe wird dabei von der letzten Note oder dem letzten Akkord übernommen. Das ist insbesondere von Nutzen, um für Musik- und Schemefunktionen Rhythmen vorzugeben und kann die Lesbarkeit der Eingabe in einigen Fällen deutlich verbessern.

```
c64[ 64] 32 16 8^- <g b d>4~ 2 | 1
```



```
\new DrumStaff \with { \override StaffSymbol.line-count = 1 }
\drummode {
  \time 3/4
  tambourine 8 \tuplet 3/2 { 16 16 16 }
             8 \tuplet 3/2 { 16 16 16 } 8 8 |
}
```



- Bebalkungsausnahmen können nunmehr mit der Schemefunktion `\beamExceptions` angegeben werden. Zuvor war die folgende Eingabe erforderlich:
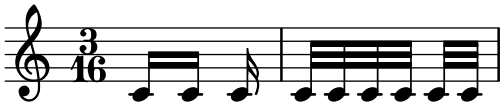
```
\set Timing.beamExceptions =
#'(                          ;Start der Aliste
   (end .                    ;Eintrag für Balkenende
    (                        ;Start der Aliste von Endpunkten
     ((1 . 32) . (2 2 2))    ;Regel für 1/32 Balken -- jedes 1/16
    )))

\time #'(2 1) 3/16
c16 c c
\repeat unfold 6 { c32 }
```

Nunmehr ist folgende Variante möglich:

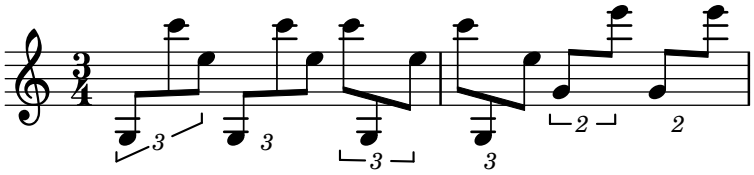
```
\set Timing.beamExceptions =
  \beamExceptions { 32[ 32] 32[ 32] 32[ 32] }

\time 2,1 3/16
c16 c c |
\repeat unfold 6 { c32 } |
```

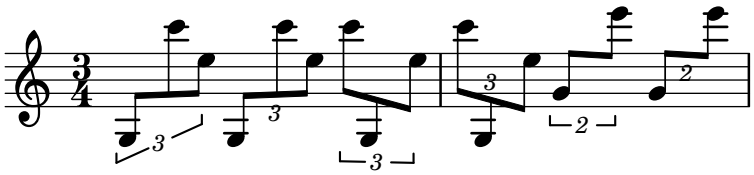


wobei zusätzliche Ausnahmeregeln mit Taktstrichen `|` in der Eingabe abzutrennen sind. Die Ausnahmeregeln können wie hier ohne Tonhöhen geschrieben werden; zwingend ist dies aber nicht.

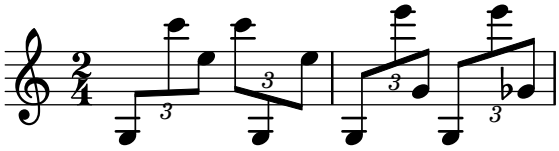
- Die Positionierung der Bezifferung von N-tolen mit Wendehalsbalken berücksichtigt eine Klammer nur dann, wenn diese auch erscheint. Vorher konnte dies solchermaßen aussehen:



Eine fehlende Klammer wird nunmehr berücksichtigt:
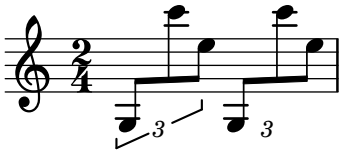


- Im Fall von Wendehalsbalken werden die N-tolen-Ziffern in die Kollisionsvermeidung eingebunden.



Das ursprüngliche Verhalten kann mit der `knee-to-beam`-Eigenschaft des `TupletNumber` Layout-Objekts abgerufen werden.

```
\time 2/4
\override Beam.auto-knee-gap = 3
\override TupletNumber.knee-to-beam = ##f
\override TupletBracket.bracket-visibility = ##t
\tuplet 3/2 4 { g8 c'' e, }
\once \override TupletBracket.bracket-visibility = ##f
\tuplet 3/2 4 { g,,8 c'' e, }
```



Neues bei Ausdrucksbezeichnungen
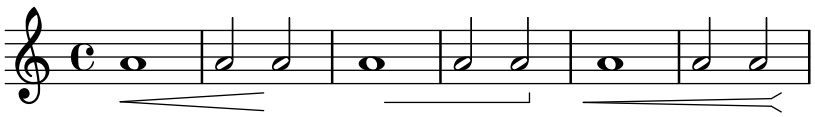
- Die Enden von Crescendo-Klammern können mit der Layout-Objekt-Eigenschaft `shorten-pair` verschoben werden. Dies war zuvor nur für Text-Klammern (mithin `TupletBracket` und `OttavaBracket`) möglich.

  Positive Werte verschieben hierbei nach rechts, negative nach links.

```
\once \override Hairpin.shorten-pair = #'(0 . 2)
a1\< | a2 a\!
```

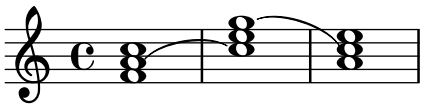
```
\once \override Hairpin.shorten-pair = #'(2 . 0)
\once \override Hairpin.stencil = #constante-hairpin
a1\< | a2 a\!

\once \override Hairpin.shorten-pair = #'(-1 . -1)
\once \override Hairpin.stencil = #flared-hairpin
a1\< | a2 a\!
```

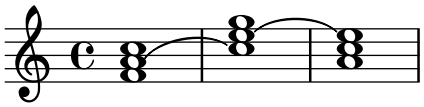


- Binde- und Phrasierungsbögen können nunmehr an einer Einzelnote innerhalb eines Akkordes verankert werden.

```
<f a( c>1 | <c') e g(> | <a c) e>
```



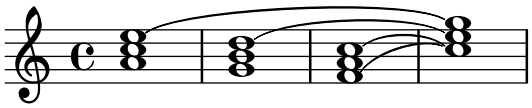
```
<f( a\( c>1 | <c'\) e\( g> | <a c e\)>
```



- Um hierbei mehr als einen Bogen gleichzeitig führen zu können, können Bogenanfänge und -enden mit `\=X` gekennzeichnet werden, so daß man zueinandergehörende Bögen mit der gleichen Ziffer oder dem gleichen Symbol markieren kann.

  Sinnvoll zu verwenden ist dies bei überlappenden oder verschachtelten Bögen.

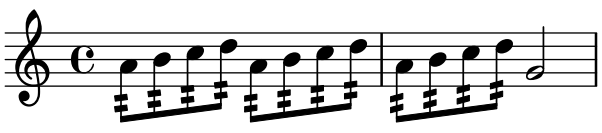
```
<a c e\=7\(>1 | <g b d\=£(> |
<f\=A( a c\="foo"(> | <c'\="foo")\=A) e\=£) g\=7\)> |
```



  Vgl. Abschnitt “Ausdrucksbezeichnungen als Bögen” in *Notationsreferenz*.

Neues bei Wiederholungsbezeichnungen

- Das Aussehen von Tremolostrichen (Form, Stil und Neigung) ist besser konfigurierbar.



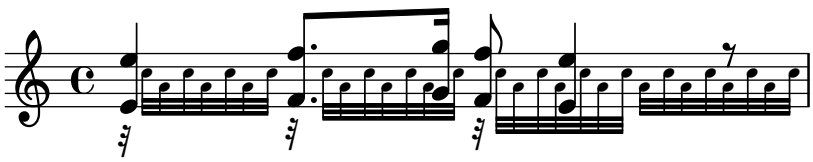
- Die Musikfunktion `\unfoldRepeats` kann optional eine Liste der auszuschreibenden Wiederholungstypen enthalten. Möglich sind hier `percent`, `tremolo` und `volta`.

  Ohne eine solche Liste wird `repeated-music` angenommen, was für alle Arten von Wiederholung gilt.

Neuerungen bei Notensystemen

- Das neue Kommando `\magnifyStaff` skaliert Systemgröße, Notenlinien, Taktstriche, Balkenabschnitte und die horizontale Platzierung innerhalb eines Notensystems. Die Notenlinien können dabei nicht gegenüber der Normalgröße ausgedünnt werden, weil die Dicken von Notenhälsen, Bögen und anderen grafischen Elementen von deren Dicke abhängen.
- Das neue Kommando `\magnifyMusic` beläßt die Größe des Notensystemes, skaliert aber Elemente wie Notenhälse, Balken, und horizontale Abstände.

```
\new Staff <<
  \new Voice \relative {
    \voiceOne
    <e' e'>4 <f f'>8. <g g'>16 <f f'>8 <e e'>4 r8
  }
  \new Voice \relative {
    \voiceTwo
    \magnifyMusic 0.63 {
      \override Score.SpacingSpanner.spacing-increment = #(* 1.2 0.63)
      r32 c'' a c a c a c r c a c a c a c
      r c a c a c a c a c a c a c a c
    }
  }
>>
```



- Die Kontextmodifikation `\RemoveAllEmptyStaves` hat dieselbe Wirkung wie zuvor schon `\RemoveEmptyStaves`, entfernt leere Notensysteme aber auch in der Anfangszeile eines Stückes.
- Ein neuer Textbeschriftungsbefehl `justify-line` setzt Wörter im Gegensatz zu `fill-line` mit konstantem *Zwischenraum* so, daß sie eine Zeile füllen.

```
\markup \fill-line {oooooo oooooo oooooo oooooo}
\markup \fill-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

```
\markup \justify-line {oooooo oooooo oooooo oooooo}
\markup \justify-line {ooooooooo ooooooooo oo ooo}
```

oooooo oooooo oooooo oooooo

ooooooooo oooooooo oo ooo

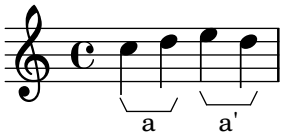
Neue Herausgeberanmerkungen

- Analyseklammern können jetzt mithilfe eines `HorizontalBracketText`-Objektes beschriftet werden.

```
\layout {
```

```
  \context {
    \Voice
    \consists "Horizontal_bracket_engraver"
  }
}

{
  \once \override HorizontalBracketText.text = "a"
  c''\startGroup d''\stopGroup
  e''-\tweak HorizontalBracketText.text "a'" \startGroup d''\stopGroup
}
```



Unterstützung weiterer Schriftarten

- Die Nutzung anderer Musikzeichensätze als LilyPonds Standardfont ‚Emmentaler' ist jetzt möglich. Mehr Information dazu bei Abschnitt "Die Notationsschriftart verändern" in *Notationsreferenz*.
- Die Standardtextzeichensätze sind nicht mehr `Century Schoolbook L`, `sans-serif`, und `monospace`.

  In der `svg`-Ausgabe:

Familie	**Standardzeichensatz**
roman	`serif`
sans	`sans-serif`
typewriter	`monospace`

  `serif`, `sans-serif`, und `monospace` werden der `generic-family` in SVG- und CSS-Spezifikation entlehnt.

  In anderen Ausgabeformaten:

Familie	**Standardzeichensatz**	**Aliasliste**
roman	`LilyPond Serif`	TeX Gyre Schola, C059, Century SchoolBook URW, Century Schoolbook L, DejaVu Serif, ..., serif
sans	`LilyPond Sans Serif`	TeX Gyre Heros, Nimbus Sans, Nimbus Sans L, DejaVu Sans, ..., sans-serif
typewriter	`LilyPond Monospace`	TeX Gyre Cursor, Nimbus Mono PS, Nimbus Mono, Nimbus Mono L, DejaVu Sans Mono, ..., monospace

  `LilyPond Serif`, `LilyPond Sans Serif`, and `LilyPond Monospace` sind Schriftschnittaliase, die in der LilyPond zugeordneten FontConfig-Konfigurationsdatei `00-lilypond-fonts.conf` definiert sind.

  When ein Zeichen nicht im ersten angegebenen Schriftschnitt aufgeführt ist, wird der nächste Schnitt genutzt.

  Eine Beschreibung der Alternativnamen ist in `00-lilypond-fonts.conf` im Installationsverzeichnis zu finden.

- Beim Einsatz von OpenType-Schriften können Font Features genutzt werden. Allerdings stellen alle OpenType-Schriften den vollständigen Satz an Funktionen bereit.

```
% Echte Kapitälchen
\markup { Normal Style: Hello HELLO }
\markup { \caps { Small Caps: Hello } }
\markup { \override #'(font-features . ("smcp"))
          { True Small Caps: Hello } }

% Zifferntypen
\markup { Normal Number Style: 0123456789 }
\markup { \override #'(font-features . ("onum"))
          { Old Number Style: 0123456789 } }

% Stilistik-Alternanten
\markup { \override #'(font-features . ("salt 0"))
          { Stylistic Alternates 0: εφπρθ } }
\markup { \override #'(font-features . ("salt 1"))
          { Stylistic Alternates 1: εφπρθ } }

% Mehrfach-Features
\markup { \override #'(font-features . ("onum" "smcp" "salt 1"))
          { Multiple features: Hello 0123456789 εφπρθ } }
```

Normal Style: Hello HELLO

Small Caps: Hello

True Small Caps: Hello

Normal Number Style: 0123456789

Old Number Style: 0123456789

Stylistic Alternates 0: εφπρθ

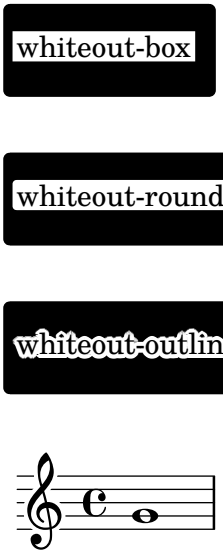
Stylistic Alternates 1: ϵϕϖϱϑ

Multiple features: Hello 0123456789 ϵϕϖϱϑ

- Zwei neue Arten von Weißumrandung werden unterstützt. Der Stil `outline` nutzt mehrfach versetzte Kopien des Schriftbild, um die äußeren Konturen in Weiß vorzuzeichnen, während der Stil `rounded-box` ein weißes Oval mit den Gesamtabmessungen vorzeichnet. Wie auch beim Standardstil `box` kann die Dicke der Umrandung über das Attribut `thickness` angegeben werden.

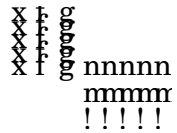
```
\markup {
  \combine
    \filled-box #'(-1 . 15) #'(-3 . 4) #1
    \override #'(thickness . 3)
    \whiteout whiteout-box
```

```
}
\markup {
  \combine
    \filled-box #'(-1 . 24) #'(-3 . 4) #1
    \override #'(style . rounded-box)
    \override #'(thickness . 3)
    \whiteout whiteout-rounded-box
}
\markup {
  \combine
    \filled-box #'(-1 . 18) #'(-3 . 4) #1
    \override #'(style . outline)
    \override #'(thickness . 3)
    \whiteout whiteout-outline
}
\relative {
  \override Staff.Clef.whiteout-style = #'outline
  \override Staff.Clef.whiteout = 3
  g'1
}
```



Ein neuer Textbefehl `\with-dimensions-from` erfüllt denselben Zweck wie `\with-dimensions`, übernimmt aber die Abmessungen des ersten angegebenen Textobjektes für das zweite.

```
\markup {
  \pattern #5 #Y #0 "x"
  \pattern #5 #Y #0 \with-dimensions-from "x" "f"
  \pattern #5 #Y #0 \with-dimensions-from "x" "g"
  \override #'(baseline-skip . 2)
  \column {
    \pattern #5 #X #0 "n"
    \pattern #5 #X #0 \with-dimensions-from "n" "m"
    \pattern #5 #X #0 \with-dimensions-from "n" "!"
  }
}
```



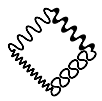
- Der Textbefehl `\draw-squiggle-line` malt eine geschlängelte Linie, wobei die Attribute `thickness`, `angularity`, `height` und `orientation` zur Anpassung genutzt werden können.

```
\markup
  \overlay {
    \draw-squiggle-line #0.5 #'(3 . 3) ##t

    \translate #'(3 . 3)
    \override #'(thickness . 4)
    \draw-squiggle-line #0.5 #'(3 . -3) ##t

    \translate #'(6 . 0)
    \override #'(angularity . -5)
    \draw-squiggle-line #0.5 #'(-3 . -3) ##t

    \translate #'(3 . -3)
    \override #'(angularity . 2)
    \override #'(height . 0.3)
    \override #'(orientation . -1)
    \draw-squiggle-line #0.2 #'(-3 . 3) ##t
  }
```

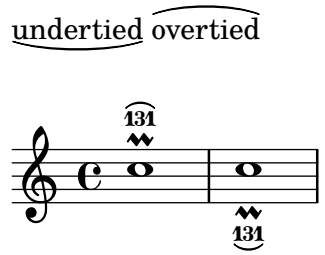


- Für Textbögen stehen die Textbefehle `\undertie` und `\overtie` sowie das beidseitig verwendbare Kommando `\tie` zur Verfügung.

```
\markup {
  \undertie "undertied"
  \overtie "overtied"
}

m = {
  c''1 \prall -\tweak text \markup \tie "131" -1
}

{ \voiceOne \m \voiceTwo \m }
```



Neuerungen bei diversen Musiksparten

Bei Gesangsmusik

- Neu verfügbar ist eine vielseitig verwendbare Vorlage für diverse Chormusik. Sie ist einsetzbar für Satz in zwei oder vier Systemen, wahlweise mit Klavierbegleitung. Im Gegensatz

zu anderen Vorlagen ist diese in LilyPond integriert und wird mit `\include` eingebunden. Mehr dazu bei Abschnitt “Eingebaute Vorlagen” in *Handbuch zum Lernen*.

- `\addlyrics` kann nunmehr mit beliebigen Kontexten genutzt werden, etwa mit `Staff`.
- `\lyricsto` und `\addlyrics` sind in ihrem Verhalten angeglichen worden. Für beide ist wie auch bei `\lyrics` und `\chords` ein mit geschweiften Klammern umschlossenes Argument nötig. Aus Gründen der Rückwärtskompatibilität kann alternativ eine Musikvariable (etwa `\mus`) verwendet werden.

  Eine Konvertierungsregel in `convert-ly` entfernt in diesem Zusammenhang redundante Einsätze von `\lyricmode`.

Neues bei Saiteninstrumenten mit und ohne Bünde

- In Tabulatur kann ein neuer Notenkopfstil Verwendung finden – `TabNoteHead.style = #'slash`.
- In Akkorddiagrammen sind die Abstände zwischen Bünden und zwischen Saiten nun separat einstellbar. Hierzu dienen innerhalb von `fret-diagram-details` die Unterattribute `fret-distance` und `string-distance`.

```
fretMrkp = \markup { \fret-diagram-terse "x;x;o;2;3;2;" }

\markuplist
\override #'(padding . 2)
\table #'(0 -1) {
  "default"

  \fretMrkp

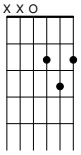
  "fret-distance"

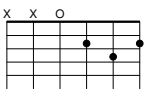
  \override #'(fret-diagram-details . ((fret-distance . 2)))
  \fretMrkp

  "string-distance"

  \override #'(fret-diagram-details . ((string-distance . 2)))
  \fretMrkp
}
```

default 

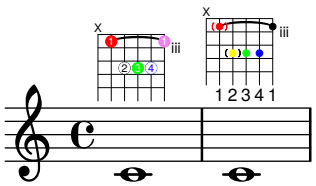
fret-distance 

string-distance 

- Kreise und Klammern in Akkorddiagrammen können bei Verwendung des Textbefehls `\fret-diagram-verbose` einzeln eingefärbt werden.

```
\new Voice {
```

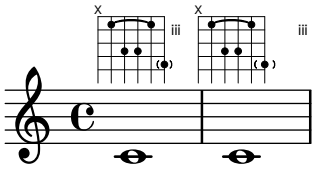
```
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . in-dot))) {
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1 red)
                               (place-fret 4 5 2 inverted)
                               (place-fret 3 5 3 green)
                               (place-fret 2 5 4 blue inverted)
                               (place-fret 1 3 1 violet)
                               (barre 5 1 3 ))
    }
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (finger-code . below-string))) {
      \fret-diagram-verbose #'((mute 6)
                              (place-fret 5 3 1 red parenthesized)
                              (place-fret 4 5 2 yellow
                                          default-paren-color
                                          parenthesized)
                              (place-fret 3 5 3 green)
                              (place-fret 2 5 4 blue )
                              (place-fret 1 3 1)
                              (barre 5 1 3))
    }
  }
}
```



- Als weitere Unterattribute in `fret-diagram-details` sind bei Verwendung des `\fret-diagram-verbose`-Textbefehls `fret-label-horizontal-offset` mit Einfluß auf `fret-label-indication` und `paren-padding` für den Zwischenraum zwischen Kreisen und umgebenden Klammern verfügbar.

```
\new Voice {
  c1^\markup {
    \fret-diagram-verbose #'((mute 6)
                             (place-fret 5 3 1)
                             (place-fret 4 5 2)
                             (place-fret 3 5 3)
                             (place-fret 1 6 4 parenthesized)
                             (place-fret 2 3 1)
                             (barre 5 2 3))
  }
  c1^\markup {
    \override #'(fret-diagram-details . (
                 (fret-label-horizontal-offset . 2)
                 (paren-padding . 0.25))) {
```

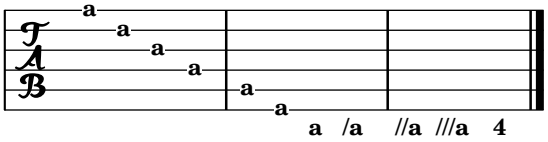
```
      \fret-diagram-verbose #'((mute 6)
                               (place-fret 5 3 1)
                               (place-fret 4 5 2)
                               (place-fret 3 5 3)
                               (place-fret 1 6 4 parenthesized)
                               (place-fret 2 3 1)
                               (barre 5 2 3))
    }
  }
}
```



- In Lautentabulaturen werden zusätzliche Baßsaiten unterstützt.
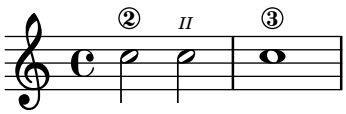
```
m = { f'4 d' a f d a, g, fis, e, d, c,  \bar "|." }

\score {
  \new TabStaff \m
  \layout {
    \context {
      \Score
      tablatureFormat = #fret-letter-tablature-format
    }
    \context {
      \TabStaff
      stringTunings = \stringTuning <a, d f a d' f'>
      additionalBassStrings = \stringTuning <c, d, e, fis, g,>
      fretLabels = #'("a" "b" "r" "d" "e" "f" "g" "h" "i" "k")
    }
  }
}
```



- Saiten können nunmehr auch mit römischen Ziffern bezeichnet werden, wie etwa bei Streichinstrumenten üblich.

```
c2\2
\romanStringNumbers
c\2
\arabicStringNumbers
c1\3
```

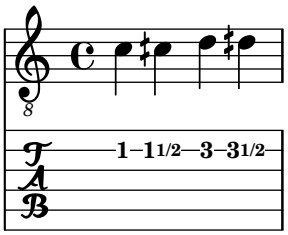


- In `TabStaff` können auch nicht ganzzahlige Bundnummern Verwendung finden, um Mikrotonalität (etwa durch Querzug) zu bezeichnen.

```
\layout {
  \context {
    \Score
    supportNonIntegerFret = ##t
  }
}

mus = \relative { c'4 cih d dih }

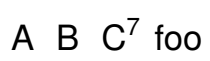
<<
  \new Staff << \clef "G_8" \mus >>
  \new TabStaff \mus
>>
```



Neues bei Akkordbezeichnungen

- Innerhalb von `\chordmode` können jetzt auch `< >` und `<< >>` für explizite Notenpassagen Einsatz finden.
- Akkordnamen können explizit durch Überschreiben ihres `text`-Attributes gestaltet werden.

```
<<
\new ChordNames \chordmode {
  a' b c:7
  \once \override ChordName.text = "foo"
  d
}
>>
```



Neues in Ein- und Ausgabe

Neues bei der Eingabestruktur

- `\header`-Blöcke können in Variablen gespeichert werden und als Parameter von Musik- und Schemefunktionen sowie als Inhalt von `#{...#}`-Konstrukten verwendet werden. Ihre Schemerepräsentation erfolgt dabei durch Guile-Module.

  Konstrukte wie `\book`, `\bookpart`, `\score`, `\with`, `\layout`, `\midi`, `\paper` können in ähnlicher Weise genutzt und durchgereicht werden; die dazu eingesetzten Datentypen sind aber andere.

Neues bei Überschriften und Kopfzeilen

- Die Seitennumerierung kann durch Anpassung der Blattvariable `pate-number-type` auch in römischen Ziffern erfolgen.

Neues beim Eingabeformat

- Ein neuer Befehl `\tagGroup` ergänzt die bestehenden Befehle `\keepWithTag` und `\removeWithTag`.

  Als Beispiel deklariert

  ```
  \tagGroup #'(violinI violinII viola cello)
  ```

  eine Gruppe von Notenmarkierungen. Verwendet man in Folge einen Befehl

  ```
  \keepWithTag #'violinI
  ```

  so sind davon nur Notenmarkierungen aus derselben Gruppe wie ‚violinI' betroffen: Musik, die mit einer von *violinI* verschiedenen Markierung aus dieser Gruppe versehen ist, wird dabei entfernt. Gruppenfremde Markierungen sind davon nicht betroffen.

Neuerungen in der Ausgabe

- Die Quelltexte von LilyPond-Dokumenten können nunmehr in die erzeugte PDF-Datei eingebunden werden. Diese Option ist standardmäßig deaktiviert und kann wegen des nicht sichtbaren Inhaltes als Sicherheitsrisiko aufgefaßt werden. Praktisch alle PDF-Betrachter unterstützen eingebettete Dokumente; anderenfalls erscheint die Ausgabe aber wie gewohnt. Die Option ist nur bei PDF-Ausgabe verfügbar.
- Die Prozedur `output-classic-framework` und die Option `-dclip-systems` sind nun auch für `SVG`-Ausgabe verfügbar.
- Die Option `-dcrop` kann zur randlosen und umbruchfreien Formatierung von `SVG`- und `PDF`-Dokumenten verwendet werden.
- Das Grafikobjektattribut `output-attributes` kann bei der SVG-Ausgabe anstelle von `id` eingesetzt werden. Es erlaubt die Definition diverser Attribute als Assoziationsliste. Beispielsweise würde der Wert `#'((id . 123) (class . foo) (data-whatever . ,,bar\))` die folgende Gruppenmarke in einer SVG-Datei erzeugen: `<g id=,,123\ class=,,foo\ data-whatever=,,bar\> ... </g>`.
- PostScript's Strichstärkenangleichung wird nicht mehr automatisch aktiviert sondern auf der Standardeinstellung der Ausgabeeinheit belassen (bei der Erzeugung von Rasterdateien nutzt Ghostscript die Strichstärkenangleichung bei Auflösungen bis zu 150dpi). Ist die Angleichung aktiviert, kommt ein komplexerer Zeichenalgorithmus zum Einsatz, der gerade bei Notenhälsen und Taktstrichen eine einheitlichere Strichstärke erzeugt.

  Die Angleichung kann mit LilyPonds Kommandozeilenoption '`-dstrokeadjust`' auch erzwungen werden. Bei der Erzeugung von PDF-Dateien wird die Dokumentvorschau dabei typischerweise deutlich besser ausfallen, was aber auf Kosten einer erheblich größeren Ausgabedatei geschieht. Das Druckbild bei hohen Auflösungen bleibt dabei unverändert.
- Die neue Funktion `make-path-stencil` erlaubt die Verwendung aller absolut und relativ positionierten `path`-Befehle, mithin:

  `lineto`, `rlineto`, `curveto`, `rcurveto`, `moveto`, `rmoveto`, `closepath`. Ebenfalls unterstützt sind die aus SVG-Pfaden bekannten entsprechenden Einbuchstabenversionen:

  `L`, `l`, `C`, `c`, `M`, `m`, `Z` und `z`. Die neue Funktion ist mit der ursprünglichen `make-connected-path-stencil`-Funktion kompatibel. Vgl. auch `scm/stencil.scm`.

Neuerungen bei MIDI

- Die meisten Phrasierungen werden nun in der MIDI-Ausgabe berücksichtigt. Akzent und Marcato erhöhen die Lautstärke von Noten, staccato, staccatissimo und portato verkürzen sie. Atemzeichen verkürzen die vorangehende Note.

  Anpassungen sind möglich durch Veränderungen der Attribute `midiLength` und `midiExtraVelocity` auf einem `ArticulationEvent`. Beispiele etwa in `ly/script-init.ly`.

- Die Länge von Atempausen nach gehaltenen Noten wird nur vom letzten Notenwert in der mit Haltebögen gebildeten Note bestimmt, was der typischen Ausführung entspricht und die Synchronisation der Atempausen verschiedener Stimmen erleichtert.
- Der ‚expression level' von MIDI-Kanälen kann durch das Kontextattribut `Staff.midiExpression` bestimmt werden, um die Lautstärke während einer Note abzuändern, wobei Werte zwischen `0.0` und `1.0` zulässig sind.

```
\score {
  \new Staff \with {
    midiExpression = #0.6
    midiInstrument = "clarinet"
  }
  <<
    { a'1~ a'1 }
    {
      \set Staff.midiExpression = #0.7 s4\f\<
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.9 s4
      \set Staff.midiExpression = #1.0 s4

      \set Staff.midiExpression = #0.9 s4\>
      \set Staff.midiExpression = #0.8 s4
      \set Staff.midiExpression = #0.7 s4
      \set Staff.midiExpression = #0.6 s4\!
    }
  >>
  \midi { }
}
```

- Bei der Midiausgabe wird der Midi-Sequenzname nun durch den Eintrag `title` im `\header`-Block (je nach Vorhandensein, auf Ebene von `\score`, `\bookpart`, `\book` oder dokumentweit) definiert. Ist dies nicht erwünscht, kann der Sequenzname stattdessen durch den `\header`-Eintrag `miditittle` vorgegeben werden.

Verbesserungen beim Extrahieren von Musik

- `\displayLilyMusic` und die zugrundeliegenden Schemefunktionen lassen redundante Notenlängenangaben nicht mehr weg. Dadurch wird es leichter, alleinstehende Längenangaben in Ausdrücken wie

```
{ c4 d4 8 }
```

  zuverlässig zu erkennen und zu formatieren.

Neues bei Abständen und Positionen

Verbesserungen beim Seitenumbruch

- Es gibt zwei neue Funktionen für den Seitenumbruch. `ly:one-page-breaking` passt die Höhe der Seite automatisch an, so dass alles auf eine einzige Seite passt. `ly:one-line-auto-height-breaking` positioniert die Musik auf eine einzige Zeile (genau wie `ly:one-line-breaking`), passt dabei aber nicht nur die Seitenbreite, sondern auch die Seitenhöhe automatisch auf das Format der Musik an.

Verbesserungen bei der vertikalen und horizontalen Platzierung

- Systeme lassen sich nun relativ zu ihrer derzeiten Position verschieben mittels der `extra-offset`-Untereigenschaft von `NonMusicalPaperColumn.line-break-system-details`. Dabei sind sowohl vertikale als auch horizontale Verschiebungen möglich. Das ist besonders

hilfreich, um kleinere Korrekturen an der automatischen vertikalen Positionierung einzelner Systeme vorzunehmen; siehe Abschnitt “Explizite Positionierung von Systemen” in *Notationsreferenz* für Einzelheiten.
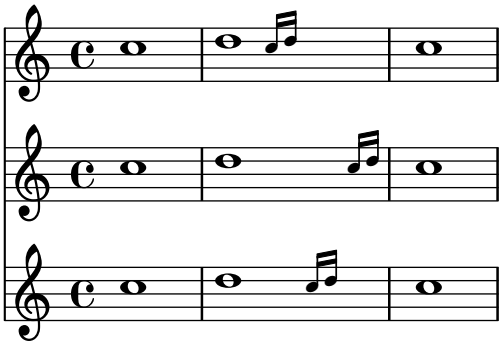
- Der optische Ausgleich der `Mi`-Notenköpfe (klein und normal) in Funk- und Walter-Notation wurde verbessert, so dass sie nun genauso breit sind wie andere Notenköpfe ihrer Familien. Auch die `So`-Notenköpfe wurden optisch in den gewöhnlichen Aiken- und Sacred-Harp-Familien verbessert, ebenso in ihren dünnen Varianten.
- `LeftEdge` hat jetzt eine definierbare vertikale Ausdehnung (`Y-extent`), siehe Abschnitt “LeftEdge” in *Referenz der Interna*.
- Grobs und ihre ‚Eltern‘ können jetzt unabhängig voneinander ausgerichtet werden, wodurch die Grob-Platzierung flexibler wird. Beispielsweise kann die linke Kante (‚left‘) eines Grobs an der Mitte (‚center‘) seines übergeordneten Objekts ausgerichtet werden.
- Die horizontale Ausrichtung von `TextScript`, `DynamicText` und `LyricText` wurde verbessert.

Neue Möglichkeiten, Voreinstellungen zu verändern

Das Kommando `\afterGrace` hat jetzt ein optionales Argument.

`\afterGrace` hat jetzt ein optionales Argument, um die horizontale Positionierung seiner Noten durch die Angabe eines Längen-Bruchteils festzulegen.

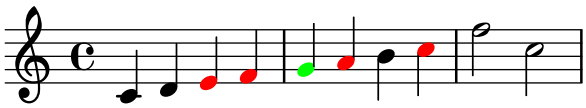
```
<<
\new Staff \relative {
  % Der Standardwert (3/4)
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Den Standardwert wie bisher manuell verändern (15/16)
  #(define afterGraceFraction (cons 15 16))
  c''1 \afterGrace d1 { c16[ d] } c1
}
\new Staff \relative {
  % Einfacher: Mit dem neuen optionalen Argument (5/6)
  c''1 \afterGrace 5/6 d1 { c16[ d] } c1
}
>>
```



- Die Kommandos `\override`, `\revert`, `\set` und `\unset` funktionieren jetzt alle auch mit dem Präfix `\once`, das sie nur einmalig wirksam werden lässt.

```
\relative {
  c'4 d
  \override NoteHead.color = #red
```

```
  e4 f |
  \once \override NoteHead.color = #green
  g4 a
  \once \revert NoteHead.color
  b c |
  \revert NoteHead.color
  f2 c |
}
```
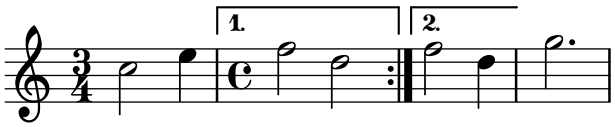


Neu bei Funktionen und Schnittstellen zu Programminterna

- Die Musik- und Grobeigenschaft `spanner-id`, die gleichzeitige Legato- und Phrasierungsbögen mit dem neuen Kommando \=X unterscheidbar macht, muss keine Zeichenkette (*string*) mehr sein, sondern ist jetzt ein Schlüssel (*key*), der entweder eine natürliche Zahl oder ein Symbol sein kann.
- Kontext-Eigenschaften, die in der Eigenschaft '`alternativeRestores`' aufgeführt werden, werden bei aufeinanderfolgenden Volta-Klammern auf ihren Wert am Beginn der *ersten* Klammer zurückgesetzt.

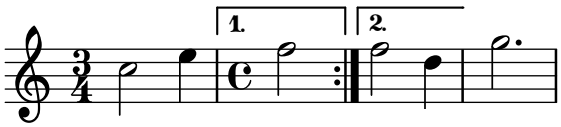
  Die derzeitige Voreinstellung stellt erstens die ‚Taktart' wieder her:

```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4 f2 d | }
  { f2 d4 | }
}
g2. |
```



  Zweitens die ‚Position im Takt':

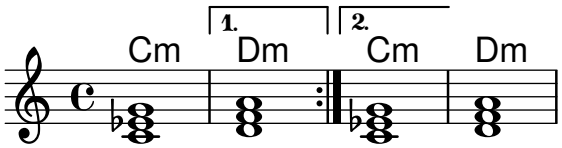
```
\time 3/4
\repeat volta 2 { c2 e4 | }
\alternative {
  { \time 4/4
    \set Timing.measurePosition = #(ly:make-moment -1/2)
    f2 | }
  { f2 d4 | }
}
g2. |
```



  Drittens ‚Akkordwechsel':

```
<<
```

```
  \new ChordNames {
    \set chordChanges = ##t
    \chordmode { c1:m d:m c:m d:m }
  }
  \new Staff {
    \repeat volta 2 { \chordmode { c1:m } }
    \alternative {
      { \chordmode { d:m } }
      { \chordmode { c:m } }
    }
  \chordmode { d:m }
}
>>
```



- LilyPond-Funktionen, die mit `define-music-function`, `define-event-function`, `define-scheme-function` und `define-void-function` definiert sind, können nun direkt von Scheme aus aufgerufen werden, als wären sie genuine Scheme-Prozeduren. Die Typ-Überprüfung und -zuordnung von Argumenten wird dabei genauso durchgeführt wie bei Aufruf der Funktion durch eine LilyPond-Eingabe, einschließlich dem Einsetzen von Default-Werten für optionale Argumente, die die geforderten Typen-Prädikate nicht erfüllen. Wenn eine Folge optionaler Argumente explizit übersprungen werden soll, kann statt `\default` auch `*unspecified*` verwendet werden.
- Aktuelle Eingabe-Position und Parser sind jetzt in GUILE-Fluids abgelegt und können durch die Funktionsaufrufe `(*location*)` and `(*parser*)` abgerufen werden. Das bedeutet, dass viele Funktionen, die bislang ein explizites `parser`-Argument benötigten, dieses nun nicht mehr brauchen.

  Funktionen, die mit `define-music-function`, `define-event-function`, `define-scheme-function` und `define-void-function` definiert werden, brauchen keine `parser`- und `location`-Argumente mehr.

  Bei solchen Funktionsdefinitionen wird LilyPond aber versuchen, den bisher üblichen Gebrauch von `parser`- und `location`- Argumenten zu erkennen, um noch für eine Weile die Rückwärtskompatibilität sicherzustellen.
- Scheme-Funktionen und -bezeichner können jetzt als Ausgabe-Definitionen verwendet werden.
- Scheme-Ausdrücke können jetzt als Akkord-Bestandteile verwendet werden.
- Musikalische Funktionen (auch Scheme- und Void-Funktionen) und Markup-Kommandos, die nur die letzten Argumente einer Folge von Overrides und Funktionsaufrufen liefern, können nun in Form eines einzelnen Ausdrucks definiert werden, der mit `\etc` endet:

```
\markup bold-red = \markup \bold \with-color #red \etc
highlight = \tweak font-size 3 \tweak color #red \etc
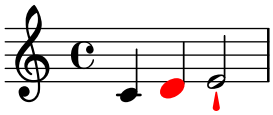
\markup \bold-red "text"
\markuplist \column-lines \bold-red { Eins zwei }

{ c' \highlight d' e'2-\highlight -! }
```

text
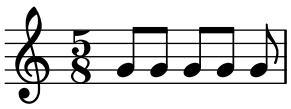
Eins

zwei



- Durch Punkte strukturierte Symbollisten wie `FretBoard.stencil` wurden schon seit Version 2.18 unterstützt. Sie können nun auch ganze Zahlen enthalten und alternativ durch Kommata getrennt werden. Das erlaubt ihre Verwendung der Art

```
{ \time 2,2,1 5/8  g'8 8 8 8 8 }
```
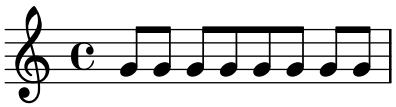


und

```
\tagGroup violine,oboe,fagott
```

- Solche Listen können auch in Zuweisungen, `\set`s und `\override`s verwendet werden. Das erlaubt einen Gebrauch wie:

```
{ \unset Timing.beamExceptions
  \set Timing.beatStructure = 1,2,1
  g'8 8 8 8 8 8 8 8 }
```



- Elementen von Alisten (assoziativen Listen) konnten schon bisher Werte einzeln zugewiesen werden (beispielsweise paper-Variablen wie `system-system-spacing.basic-distance`). Nun kann man auf sie auf die gleiche Art auch verweisen, etwa:

```
\paper {
  \void \displayScheme \system-system-spacing.basic-distance
}
```

Zusammen mit den zuvor beschriebenen Neuerungen erlaubt dies das Definieren von (und Zugreifen auf) Pseudovariablen wie `violin.1`.

- Es gibt jetzt den Beschriftungslistenbefehl `\table` für Tabellen. Jede Spalte kann individuell ausgerichtet werden:

```
\markuplist {
    \override #'(padding . 2)
    \table
      #'(0 1 0 -1)
      {
        \underline { zentriert rechtsbündig zentriert linksbündig }
        ein "1" Tausendstel "0.001"
        elf "11" Hundertstel "0.01"
```

```
        zwanzig "20" Zehntel "0.1"
        tausend "1000" eins "1.0"
      }
}
```

zentriert	rechtsbündig	zentriert	linksbündig
ein	1	Tausendstel	0.001
elf	11	Hundertstel	0.01
zwanzig	20	Zehntel	0.1
tausend	1000	eins	1.0

- `InstrumentName` unterstützt jetzt `text-interface`.
- Die Eigenschaft `thin-kern` des Grobs `BarLine` (Taktstrich) heißt jetzt `segno-kern`.
- `KeyCancellation`-Grobs ignorieren jetzt Notenschlüssel in Stichnoten (wie es `KeySignature`-Grobs auch tun).
- Kommandos der Form `\once \unset` sind jetzt erlaubt.