

Funcionalidades nuevas de la versión 2.16 desde la 2.14

- Se puede elegir entre dos métodos de numeración de compases, en especial para cuando se emplean repeticiones:

The image displays six musical staves, each illustrating a different method of numbering repeated measures. The staves are arranged vertically and are labeled 1 through 6c. Each staff shows a sequence of notes and rests, with a bracket indicating a repeated section. The numbering methods are as follows:

- Staff 1: First ending, labeled '1.'.
- Staff 2: Second ending, labeled '2.'.
- Staff 3: Third ending, labeled '3.'.
- Staff 4: First ending, labeled '1.'.
- Staff 5: Second ending, labeled '2.'.
- Staff 6: Third ending, labeled '3.'.
- Staff 6c: Third ending, labeled '3.'.

- Las expresiones de Scheme dentro de fragmentos de código de LilyPond incrustados (`{...#}`) se ejecutan ahora dentro de la cerradura léxica del código de Scheme circundante. El símbolo `$` ya no es especial dentro del código de LilyPond incrustado. Se puede utilizar de forma incondicional dentro de código de LilyPond para su evaluación inmediata, de forma parecida a la forma en que se utilizaba anteriormente `ly:export`. Se ha suprimido `ly:export`. Como consecuencia, ahora `#` está libre para diferir la evaluación de su argumento hasta que el analizador sintáctico reduzca efectivamente la expresión contenida, reduciendo significativamente el potencial de la evaluación prematura.
- Se ha mejorado el soporte de acordes de tipo jazz: se reconocen los acordes lidios y alterados; ahora se tratan los separadores entre modificadores de acorde de forma independiente de los separadores entre acordes invertidos y sus notas de bajo (y por omisión, la barra inclinada se usa ahora solamente para el último tipo de separador); las notas adicionales ya no van prefijadas por "add" de forma predeterminada; y la "m" en los acordes menores se puede personalizar. Consulte [Sección "Nombres de acorde personalizados" in Referencia de la Notación](#) para más información.
- Se ha cambiado el nombre de la instrucción `\markuplines` por `\markuplist` para conseguir una mejor correspondencia con su semántica y con la nomenclatura generarl de LilyPond.

- Se ha simplificado considerablemente la interfaz para especificar afinaciones en las tablat-
uras.
- Las barras ahora pueden preservar la inclinación por encima de los saltos de línea.



Para hacerlo, se han hecho obsoletas varias funciones de "callback".

- `ly:beam::calc-least-squares-positions`
- `ly:beam::slope-damping`
- `ly:beam::shift-region-to-valid`

Además, `ly:beam::quanting` ahora acepta un argumento adicional para ayudar a los cálculos sobre los cambios de línea. Todas estas funciones se llaman automáticamente cuando se ajusta el parámetro `positions`.

- En los argumentos de función, la música, los elementos de marcado y las expresiones de Scheme (así como algunas otras entidades sintácticas) se han hecho mayormente intercambiables y se diferencian solamente mediante la evaluación del predicado respectivo.
- Ahora se pueden definir las funciones musicales (y sus parientes cercanos) con argumentos opcionales.
- Para definir instrucciones que se ejecutan solamente por sus efectos secundarios, ahora está disponible `define-void-function`.
- Hay una instrucción nueva `define-event-function` en analogía con `define-music-function` que se puede usar para definir funciones musicales que actúan como eventos sin que se requiera un especificador de dirección como `(-, ^` o `_)` antes de ellos.

```
dyn=#(define-event-function (parser location arg) (markup?)
      (make-dynamic-script arg))
\relative c' { c\dyn pfsss }
```



- Se puede incluir una lista de alias en ASCII para caracteres especiales.

```
\paper {
  #(include-special-characters)
}
\markup "&bull; &dagger; &copyright; &OE; &ss; &para;"
```

• † © Œ ß ¶

- Hay una instrucción nueva `define-scheme-function` en analogía con `define-music-function` que puede usarse para definir funciones que se evalúan a expresiones de Scheme pero aceptan argumentos en la sintaxis de LilyPond.
- Ahora se puede utilizar la construcción `#{ ... #}` no solo para crear listas secuenciales de música, sino también para eventos musicales únicos, expresiones musicales vacías, post-eventos, elementos de marcado (sobre todo para liberar a los usuarios de la necesidad de usar la macro `markup`), listas de marcado, expresiones numéricas, definiciones y modificaciones de contextos y algunas otras cosas. Si no contiene nada o contiene un único evento musical, ya no devuelve una lista secuencial de música, sino una expresión musical vacía o simplemente el propio evento musical, respectivamente.
- Nueva opción de la línea de órdenes `--loglevel=level` para controlar el volumen de datos que LilyPond produce en la salida. Los valores posibles son ERROR (errores), WARN (advertencias), BASIC_PROGRESS (progreso básico), PROGRESS (progreso) y DEBUG (depuración).
- `\set \once` ahora reinicia correctamente el valor de la propiedad al valor previo.



- La alineación de los elementos de matiz dinámico extensos (reguladores, crescendi textuales, etc.) se divide automáticamente si se da explícitamente una dirección distinta.



- Ahora las apoyaturas y mordentes funcionan también dentro de una ligadura de expresión, y no solo dentro de una ligadura de fraseo. Asimismo, se ha añadido la función `\slashedGrace` que no imprime ninguna ligadura partiendo de la nota del mordente.



- Para suprimir a línea en un elemento de crescendo extenso (y otros elementos extensos similares), LilyPond contempla ahora de forma plena la propiedad `#'style = #'none`.



- LilyPond.app está disponible ahora para MacOS X 10.7. ¡Gracias, Christian Hitz!
- Los glissandos pueden abarcar varias líneas.