multicolrule — Decorative rules between columns

Karl Hagen

Released 2019/01/01

Abstract

The multicolrule package lets you customize the appearance of the vertical rule that appears between columns of multicolumn text. It is primarily intended to work with the multicol package, hence its name, but it also supports the twocolumn option and \twocolumn macro provided by the standard classes (and related classes such as the KOMA-Script equivalents), as well as the bidi package (and through it, all RTL scripts loaded with polyglossia).

Contents

1 Introduction 2
   1.1 Bugs and Known Limitations 2
   1.2 License 3

2 Package Options 3
   2.1 Default Operation 3
   2.2 Option ‘tikz’ 3
   2.3 Option ‘twocolumn’ 3

3 The User Interface 4
   3.1 Styles with the ’line-style’ option 5
      3.1.1 Custom Patterns 6
   3.2 Colors 7
   3.3 Width 8
   3.4 Repeated Rules 8
   3.5 Extended Rules 8
   3.6 Rule Patterns 9

4 Implementation 10
   4.1 Preliminaries 10
   4.2 Patching Output Routines 12
   4.3 Creating the Rules 14
      4.3.1 Tikz-only Routines 17
   4.4 Color 19
   4.5 Patterns 19
   4.6 Key-Values 20
   4.7 User Interface 21

*This file describes version v1.2, last revised 2019/01/01.
†latex@polysyllabic.com
In \\	exttt{\LaTeX}, there are two lengths that control the formatting between columns of \texttt{multicolumn} text: \texttt{\textbackslash columnsep} specifies the space between adjacent columns, and \texttt{\textbackslash columnseprule} specifies the width of a solid vertical rule that is placed centered between the columns. The \texttt{multicol} package adds the ability to change the color of the rule, but in both vanilla \texttt{\LaTeX} and \texttt{multicol}, the rule itself is drawn directly inside the routines that output the column boxes, and is therefore difficult for users to alter.

Of course it’s a legitimate question why anyone should want to change this rule, or indeed use one at all, as good typography tends to avoid using large vertical lines.\footnote{See, for example, the remarks in the documentation for the \texttt{booktabs} package} In my own case, I needed to modify the rule because of the requirements of a particular style I was imitating, and having done the hard work of creating the necessary infrastructure for one line style, it was simple to extend the solution to a more general case. I hope someone else will find the options here useful.

The basic line styles that \texttt{multicolrule} makes available are illustrated throughout this guide. The default line-width used is 0.4pt (thin), and the default color is Maroon. You can also find examples of rules created with all available options in the file \texttt{mcrule-example.pdf}.

\section*{New for Version 1.2}

Version 1.2 adds the ability to define patterns, which are aliases for a series of \texttt{\SetMCRule} settings. With patterns, you can change individual separators on the same page. For example, in three-column text, the left separator can differ from the right. You can also alter the appearance of one or more separators anywhere within the environment (see section \ref{patterns}).

\section*{New for Version 1.1}

Version 1.1 supports drawing decorative rules if you have the \texttt{bidi} package loaded, which can occur automatically if you set a right-to-left language with \texttt{polyglossia}. It also provides a mechanism to extend or shrink rules beyond the natural height of the columns, as well as to have the rule fill the available space to the end of the text area (see section \ref{decorative-rules}).

\section*{1.1 Bugs and Known Limitations}

The \texttt{multicolrule} package is written using expl3 syntax, and so requires a less-than-ancient installation of \texttt{\LaTeX}. It requires the packages \texttt{l3keys2e}, \texttt{xparse}, \texttt{xpatch}, \texttt{xcolor}, \texttt{scrlfile}, and depending on the mode of operation may also require \texttt{multicol} and \texttt{tikz}. If you have an up-to-date distribution, these requirements should cause no issues.

I am sure that there are bugs that remain to be uncovered, inefficient methods that could stand improvement, and useful features that still need to be implemented. The development code is maintained on github, and you can file feature requests or bug reports there. Alternatively, you can send an email to latex@polysyllabic.com. I welcome contributions for additional styles, especially to provide more options when running the package without \texttt{tikz}.

The following are the issues I’m currently aware of that aren’t \texttt{multicolrule} errors but which may cause buggy looking behavior:
This package works by patching the output routines of either `multicol` or the \LaTeX\ kernel, depending on the mode of operation. If bidi is loaded, it will also patch that. It will have no effect if you use a class or package that outputs column text via alternate mechanisms. This includes `parcolumns`, and probably other classes and packages designed to typeset parallel-column text as well, although I have not done a survey to determine whether this is the case. If you would like support for one of these, please send me an email or file a feature request on github and I’ll see what I can do.

The line styles that work by repeating elements in a tiled pattern may have significant gaps at the end of columns, particularly for larger patterns. You can mitigate this problem slightly by tweaking the spaces above and below a pattern, but the basic problem is a side-effect of the way these patterns are implemented (with `\cleaders`), which means that only an integer number of copies can be produced. Lines drawn with tikz do not have this problem.

Extending rules beyond their natural column lengths can seriously mess up the output, including, in certain edge cases, causing `multicol` to overprint columns or even put them in the margins. The fact that the extended rule occupies space on the page instead of extending up into the margin was a deliberate design decision and is necessary to support the `extend-fill` and `extend-reserve` options work correctly. A future version may support drawing the rules to a background layer so that the text is not shifted.

### 1.2 License

The `multicolrule` package is copyright 2018–2019 by Karl Hagen. It may be distributed and/or modified under the conditions of the \LaTeX\ Project Public License, either version 1.3c of this license or (at your option) any later version. The latest version of this license is in [http://www.latex-project.org/lppl.txt](http://www.latex-project.org/lppl.txt).

This work has the LPPL maintenance status 'maintained.' The Current Maintainer of this work is Karl Hagen.

### 2 Package Options

#### 2.1 Default Operation

Loading `multicolrule` with its default settings enables multicol support, and that package will be loaded if it hasn’t been already. Note that if you need to pass any parameters to `multicol`, such as `docolaction`, you should load `multicol` with the appropriate settings before you load `multicolrule`, as \LaTeX\ does not support reloading packages with different parameters.

#### 2.2 Option ‘tikz’

You can use more line styles if you also use the tikz package. Some line styles are only available if tikz is enabled, and others look better with it. The default behavior of `multicolrule` depends on the status of the tikz package at the time `multicolrule` is loaded. If `multicolrule` detects that tikz is already loaded, then tikz support will be enabled by default. Otherwise, you need the `tikz` to enable it. This option also accepts explicit boolean values, so you can pass `tikz=false` if you want to explicitly disable tikz support. If tikz support is not enabled (or if it is explicitly disabled), the line styles marked tikz only in section 3.1 will be unavailable and errors will result if you try to use them.

#### 2.3 Option ‘twocolumn’

The `multicolrule` package recognizes the option `twocolumn`, either as a package option
or as a global class option. If you pass this
option to your document class, you do not
need to pass it a second time to the pack-
age. It is only necessary to use the package
option if you plan to have a predominantly
one-column document and use \twocolumn
to switch temporarily into two-column mode.

Because multicol does not work well with
the ordinary two-column mode, multicolrule


\begin{itemize}
\item color
\item color-model
\item custom-line
\item custom-pattern
\item custom-tile
\item double
\item extend-bot
\item extend-fill
\item extend-reserve
\item extend-top
\item line-style
\item pattern-after
\item pattern-for
\end{itemize}

Table 1: \SetMCRule keys

<table>
<thead>
<tr>
<th>Key</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>Set the color of the rule (see sec. 3.2)</td>
</tr>
<tr>
<td>color-model</td>
<td>Set the color model of the rule (see sec. 3.2)</td>
</tr>
<tr>
<td>custom-line</td>
<td>Set a custom tikz line for the rule (tikz only; see sec. 3.1.1)</td>
</tr>
<tr>
<td>custom-pattern</td>
<td>Set a custom individual pattern for the rule (see sec. 3.1.1)</td>
</tr>
<tr>
<td>custom-tile</td>
<td>Set a custom tiling pattern for the rule (see sec. 3.1.1)</td>
</tr>
<tr>
<td>double</td>
<td>Draw two copies of the rule (see sec. 3.4)</td>
</tr>
<tr>
<td>extend-bot</td>
<td>Set an extra amount to extend the rule at the bottom of the column (see sec. 3.5)</td>
</tr>
<tr>
<td>extend-fill</td>
<td>Extend rule to the bottom of the text area (multicol only; see sec. 3.5)</td>
</tr>
<tr>
<td>extend-reserve</td>
<td>Space to reserve at bottom of text area when using extend-fill (multicol only; see sec. 3.5)</td>
</tr>
<tr>
<td>extend-top</td>
<td>Set an extra amount to extend the rule at the top of the column (see sec. 3.5)</td>
</tr>
<tr>
<td>line-style</td>
<td>Select the type of rule printed (default=default; see sec. 3.1)</td>
</tr>
<tr>
<td>pattern-after</td>
<td>Number of separators to delay before beginning to use the specified patterns (default=0; see sec. ??)</td>
</tr>
<tr>
<td>pattern-for</td>
<td>Number times separators to apply the patterns to before returning to default (default=−1; see sec. ??)</td>
</tr>
</tbody>
</table>
Table 1: \SetMCRule keys (cont.)

<table>
<thead>
<tr>
<th>Key</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>patterns</td>
<td>Specify one or more patterns use to draw rules. (default=none; see sec. ??)</td>
</tr>
<tr>
<td>single</td>
<td>Draw a single copy of the rule (default; see sec. 3.4)</td>
</tr>
<tr>
<td>repeat</td>
<td>Set the number of times to draw the rule (see sec. 3.4)</td>
</tr>
<tr>
<td>repeat-distance</td>
<td>Set the horizontal space between adjacent copies of repeated rules (see sec. 3.4)</td>
</tr>
<tr>
<td>triple</td>
<td>Draw three copies of the rule (see sec. 3.4)</td>
</tr>
<tr>
<td>width</td>
<td>Set the width of the rule (see sec. 3.3)</td>
</tr>
</tbody>
</table>

3.1 Styles with the 'line-style' option

You can choose a style for the rule with the line-style key. If the predefined styles are insufficient for your purpose, see section 3.1.1 for different ways to customize the rule in even more radical ways. The width of many line styles scales directly with the setting of \columnseprule, the default \LaTeX length that controls the width of the column rule, but even those that do not, the width must be non-zero for the rule to display (see section 3.3).

Table 2 summarizes the available line styles. Most of the basic patterns come in three versions, differing only in how closely the pattern is spaced: normal, dense, and loose. These settings parallel those found in \tikz and use the same spacing between elements. There are no named settings for double lines and the like because you control that feature separately, with the repeat key. All line styles can be repeated as many times as you like (see section 3.4).

Table 2: Styles available for the line-style key

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>circles</td>
<td>A series of hollow circles (\textit{tikz only})</td>
</tr>
<tr>
<td>dash-dot</td>
<td>A dash followed by a square dot (\textit{tikz only})</td>
</tr>
<tr>
<td>dash-dot-dot</td>
<td>A dash followed by two square dots (\textit{tikz only})</td>
</tr>
<tr>
<td>dashed</td>
<td>A series of dashed lines</td>
</tr>
<tr>
<td>default</td>
<td>A solid rule drawn the same way as the default multicol rule. Does not support extended rules.</td>
</tr>
<tr>
<td>dense-circles</td>
<td>The same as \textit{circles} but more closely spaced (\textit{tikz only})</td>
</tr>
<tr>
<td>dense-dots</td>
<td>The same as \textit{dots} but more closely spaced</td>
</tr>
<tr>
<td>dense-solid-circles</td>
<td>The same as \textit{solid-circles} but more closely spaced (\textit{tikz only})</td>
</tr>
<tr>
<td>densely-dash-dot</td>
<td>The same as \textit{dash-dot} but more closely spaced (\textit{tikz only})</td>
</tr>
<tr>
<td>densely-dash-dot-dot</td>
<td>The same as \textit{dash-dot-dot} but more closely spaced (\textit{tikz only})</td>
</tr>
<tr>
<td>densely-dashed</td>
<td>The same as \textit{dashed} but more closely spaced</td>
</tr>
<tr>
<td>densely-dotted</td>
<td>The same as \textit{dotted} but more closely spaced</td>
</tr>
</tbody>
</table>
### Table 2: Available line-style settings (cont.)

<table>
<thead>
<tr>
<th>Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dots</code></td>
<td>A series of dots drawn with the period (full-stop) of the current font</td>
</tr>
<tr>
<td><code>dotted</code></td>
<td>A series of square dots</td>
</tr>
<tr>
<td><code>loose-dots</code></td>
<td>The same as <code>dots</code> but spaced further apart</td>
</tr>
<tr>
<td><code>loose-circles</code></td>
<td>The same as <code>circles</code> but spaced further apart (tikz only)</td>
</tr>
<tr>
<td><code>loose-solid-circles</code></td>
<td>The same as <code>solid-circles</code> but spaced further apart (tikz only)</td>
</tr>
<tr>
<td><code>loosely-dash-dot</code></td>
<td>The same as <code>dash-dot</code> but spaced further apart (tikz only)</td>
</tr>
<tr>
<td><code>loosely-dash-dot-dot</code></td>
<td>The same as <code>dash-dot-dot</code> but spaced further apart (tikz only)</td>
</tr>
<tr>
<td><code>loosely-dashed</code></td>
<td>The same as <code>dashed</code> but spaced further apart</td>
</tr>
<tr>
<td><code>loosely-dotted</code></td>
<td>The same as <code>dotted</code> but spaced further apart</td>
</tr>
<tr>
<td><code>solid</code></td>
<td>A solid line, like <code>default</code>, but supports extending rules</td>
</tr>
<tr>
<td><code>solid-circles</code></td>
<td>A series of filled circles (tikz only)</td>
</tr>
</tbody>
</table>

The **default** and **solid** line styles are nearly the same, except that the **solid** line (as of version 1.1) supports the rule-extension commands described in section 3.5. This means that if you want a solid rule with altered top or bottom extensions, you must explicitly set the line style to **solid**. If you make no calls to \SetMCRule after loading **multicolrule**, the column divider will continue to behave exactly as it does with the ordinary multicol package.

You can alter the rule’s width and color either through \SetMCRule with the `width` and `color` keys described in sections 3.3 and 3.2, respectively, or directly by changing the value of \columnseprule and renewing the \columnseprulecolor macro.

The **dots** style and its variants are rendered with a period (.) in the currently active font. This is one of the styles, mentioned above, that do not change their size as the line width increases. The same is true of custom tiles.

The **dotted** styles differ from `dots` in that the former are squares with side lengths equal to \columnseprule. This mirrors the behavior of the equivalently named dotted patterns in tikz.

### 3.1.1 Custom Patterns

```latex
\texttt{custom-tile} = \{\langle pattern\rangle \} \{\langle space\ above\rangle \} \{\langle space\ below\rangle \}
```

There are three options to create custom rules with **multicolrule**. The first is the **custom-tile** key. This creates a rule consisting of vertically stacked boxes of arbitrary content—the tile—running the height of the column separator. The **custom-tile** key takes three parameters, which must all be enclosed brackets and may not be omitted. The first should contain the tokens you want to appear as the content of the tile. The second is a dimension specifying the leading vertical space to apply above each copy of the tile. The third is a dimension specifying the trailing vertical space to insert below each copy of the tile.

The rule in this section uses the \texttt{\SparkleBold} symbol from bbding. Notice that when you use the **custom-tile**
parameter, of any of the other custom key commands, you do not specify a separate line-style. If you try to provide both, the last style given in the list will be the one that is kept.

\begin{verbatim}
custom-pattern = \{\text{\textbackslash{HandRight}}\} \{(shift down)\} \{(shift up)\}
\end{verbatim}

The second custom option is with the custom-pattern key. The syntax is identical to that for custom-tile, but the content you specify will appear once per page or column pair (if the columns occupy less than a full page). This content will be vertically centered if the second and third parameters are both 0pt. You can shift the content down by increasing the second parameter, and up by increasing the third. The rule in this section uses the \textbackslash{HandRight} symbol from bbdng.

\begin{verbatim}
custom-line = \{(\text{\textbackslash{draw command}})\}
\end{verbatim}

The third custom pattern involves setting your own tikz drawing function using the key custom-line. The rule in this section is drawn with an ornament from pgfornaments. Obviously, this feature requires tikz support. The value you provide to the custom-line key should consist of a tikz command, such as \textbackslash{draw} or \textbackslash{path}, without the surrounding \texttt{tikzpicture} environment.

Before the drawing command is called, \texttt{multicolrule} will set up a \texttt{tikzpicture} with both the x- and y-coordinates scaled to points, and two nodes, named (TOP) and (BOT), which are set to the coordinates of the top and bottom of the rule. You can then specify your own \texttt{\textbackslash{draw}} or \texttt{\textbackslash{path}} function in whatever way you like. The rule separating these columns was drawn with a decorative element from the pgfornaments package.

This function will use the color set in \texttt{\columnseprulecolor} if you don’t set it explicitly within the tikz command, but you must provide everything else necessary to draw the line correctly, including the line width. Note that this function should be considered experimental. It works for single-line commands such as the one shown in the example, but I haven’t tested it with anything more elaborate.

### 3.2 Colors

You can set colors for the rule through the color and, optionally, the color-model keys. \texttt{multicolrule} loads the xcolor package to manage colors, and the color parameter accepts any name that xcolor recognizes, either natively or as the result of any names you have defined with \texttt{\definecolor} (see the xcolor documentation). Note that if you want to use color names that are defined through the one of xcolor’s package options, you must load xcolor before both \texttt{multicolrule} and tikz with the relevant options.

To specify a color by a numeric specification, you use the color-model parameter to specify any color model that xcolor recognizes (rgb, cmy, etc), and color to hold the color-specification list. Because that list is itself comma-separated, you must enclose it in brackets.

The current color setting can always be found in \texttt{\columnseprulecolor}. If you are running in twocolumn mode without multicols, this command will be provided and colors will work the same way they do with multicols. Note that setting the color key causes \texttt{\columnseprulecolor} to be redefined within the current group only. If you directly redefine \texttt{\columnseprulecolor}, the color of the custom rule will reflect this setting. This way, the settings of any packages that might alter the rule color will be respected.
3.3 Width

You can set the width of the rule with the `width` key. Legal values are any explicit dimension or dimension expression, as well as with names that parallel those used by `tikz`, except that spaces in the key names are replaced with hyphens.

The current width of the rule is kept in \texttt{\columnseprule}, just as in vanilla \LaTeX, and if it is set separately, the custom rule’s width will reflect this change. Note that although some line styles do not depend directly on \texttt{\columnseprule} to calculate their actual width, the value of \texttt{\columnseprule} must be greater than 0pt for any rule to appear. This behavior is intentional and is in keeping with the way the default column rules work.

<table>
<thead>
<tr>
<th>Name</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>ultra-thin</td>
<td>0.1pt</td>
</tr>
<tr>
<td>very-thin</td>
<td>0.2pt</td>
</tr>
<tr>
<td>thin</td>
<td>0.4pt</td>
</tr>
<tr>
<td>semithick</td>
<td>0.6pt</td>
</tr>
<tr>
<td>thick</td>
<td>0.8pt</td>
</tr>
<tr>
<td>very-thick</td>
<td>1.2pt</td>
</tr>
<tr>
<td>ultra-thick</td>
<td>1.6pt</td>
</tr>
</tbody>
</table>

3.4 Repeated Rules

You can draw multiple, adjacent copies of any rule by setting the number of times to draw the rule with the `repeat` key. The space between copies is controlled with the `repeat-distance` key. Initially, this distance is set to \texttt{\columnseprule}. Note that you must enter an actual dimension expression for this distance. The names used for line widths are not accepted.

The keys \texttt{single}, \texttt{double}, and \texttt{triple} are shorthand methods to set the number of repeats and the `repeat-distance` at the same time. If you use the key without a value \texttt{repeat-distance} is set to \texttt{\columnseprule}.

There are no checks made to ensure that repeated rules will fit in the available space between columns, so you should be careful using these commands, especially with thicker rules.

3.5 Extended Rules

You can specify an additional amount by which the top or bottom of the rule projects beyond the column’s natural length with the keys \texttt{extend-top} and \texttt{extend-bottom}, each of which can be set to a dimension expression. Extending the top of the rule with a positive dimension will push the columns down from any preceding material. A positive value for \texttt{extend-bottom} does the same in the other direction when a column ends in the middle of a page, but the rule will extend into the bottom margin if the column goes to the end of the page, and so you probably only want to use this in very limited situations where
THE USER INTERFACE

you need a special effect for one column or a small \multicol environment. Overprinting and other bizarre effects can result from extending the rule in the wrong place. Negative values for both keys may be more generally useful, as they have the effect of shrinking the rule. This behavior is illustrated with the rule for this section.

The extend-fill key is a boolean option that, when set to true, will extend the rule to occupy any space between the bottom of the columns and the end of the text area. Providing the key with no value is equivalent to extend-fill=true. This option has no effect unless the \multicol package is loaded.

If you want text below the \multicols environment when using extend-fill, you can reserve space for it with extend-reserve, which takes a dimension expression specifying the vertical space to leave available after the rule. If the value is greater than zero, the height of the extended line will be reduced by the reserved amount plus the value of \multicolsep. In other words, you only have to specify the actual space you need for the text itself, not the space that \multicol adds automatically below the columns. Note that if the amount you request for reserved space is less than the amount actually available at the end of the page, the rule will not extend below the columns and you probably will find this material spilling onto the next page anyway.

3.6 Rule Patterns

\DeclareMCRulePattern \langle name \rangle \langle key-value list \rangle

A "pattern" refers to a bundle of settings used by \multicolrule. You can declare a pattern for a line style with the command \DeclareMCRulePattern. The \langle name \rangle should consist of letters and hyphens only. The \langle key-value list \rangle can contain all keys that are valid for \SetMCRule with the exception of patterns. If you put something like patterns=foo in the definition of a pattern, you won’t get an error, but it will be ignored.

Once you have declared a pattern, you can use it as a value for the patterns argument of \SetMCRule. This key can accept either a single pattern or a comma-separated list of patterns. If you use a comma-separated list, make sure you enclose it in braces.

When a pattern is in effect, its settings are applied on top of whatever the prior settings are. If you set the key to an empty list, any patterns currently in effect will be canceled, and \multicolrule will revert to the previous settings.

If the patterns key contains more than one pattern, \multicolrule will cycle through the list of patterns, using one pattern each time a rule is drawn between columns. (Note, the patterns do not cycle within a single column separator if you use the repeat key.) This cycle is global, so if the number of columns is not a multiple of the number of patterns and you start a new \multicols environment with the same patterns in effect, the cycle will pick up where it left off. Every time you set new patterns, however, the cycle begins anew.

The columns above were defined with the following:

\DeclareMCRulePattern{left-hand}{custom-tile={\HandLeft}{8pt}{8pt}}
\DeclareMCRulePattern{right-hand}{custom-tile={\HandRight}{8pt}{8pt}}
\begin{multicols}{3}
\SetMCRule{patterns={right-hand,left-hand}}
... 
\end{multicols}
If you want to alter the rule only for certain column separators, you can use the `pattern-after` and `pattern-for` keys, both of which take integer values, in conjunction with `patterns`.

The `pattern-for` key means "use the given pattern or patterns for this many column separators only." Afterwards, the pattern will be disabled, meaning that it won’t be applied any more and only the settings applied directly will be in effect until it is reset. A negative value to this key means that the patterns will be repeated indefinitely. The default is \(-1\).

The `pattern-after` key means "wait until after this many column separators before starting to apply the pattern. The default is 0. If you use it in conjunction with `pattern-for`, the count of modified column separators begins after the skipped columns.

For example, suppose you have four-column text and want to alter the third column separator on the first page of the environment only. You could accomplish this task with the code above.

Using predefined patterns adds processing overhead, since they must be applied each time the rule is drawn. Therefore it is more efficient to avoid patterns unless you need to actually change the line style from column to column, although if you compile on a reasonably modern computer, you are unlikely to notice too much delay.

Note that any settings you provide in the same command where you apply a `patterns` key do not alter the definition of the pattern. If you do this, you are altering the settings in effect before the pattern is applied.

Shrinking the final two column separators in four-column text:

\begin{verbatim}
\texttt{\textbackslash DeclareMCRulePattern{shrink-me}{line-style=solid,\
extend-top=-3\baselineskip}}
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash begin{multicols}{4}}
\texttt{\textbackslash SetMCRule{patterns=shrink-me,pattern-after=1,pattern-for=2}}
\texttt{\textbackslash \ldots\textbackslash end{multicols}}
\end{verbatim}

\section{Implementation}

\paragraph{4.1 Preliminaries}

We always need these packages.

\begin{verbatim}
\texttt{\textbackslash Requires\textbackslash Package{l3keys2e}}
\texttt{\textbackslash Requires\textbackslash Package{xpatch}}
\texttt{\textbackslash Requires\textbackslash Package{xcolor}}
\texttt{\textbackslash Requires\textbackslash Package{scrlfile}}
\end{verbatim}

Define the messages we use.

\begin{verbatim}
\texttt{\textbackslash msg\_new:nnn\{multicolrule\}\{patch\-success\}\{Patched\-#1\.\}}
\texttt{\textbackslash msg\_new:nnn\{multicolrule\}\{patch\-failure\}\{Error\-patching\-#1\.\}}
\texttt{\textbackslash msg\_new:nnn\{multicolrule\}\{tikz\-required\}\{Tikz\-required\}}
\end{verbatim}

\footnote{Remember that you have one less column separator than you have columns.}

patterns=shrink-me, pattern-for=1
See the code sample below for the definition of 'shrink-me'.
4 IMPLEMENTATION

{The `'#1'` setting requires tikz to work. Either load tikz before you load multicolrule or use multicolrule's `tikz` package option.}
\msg_new:nnnn {multicolrule} {multicol-loaded} {Multicol loaded} {You are using the `twocolumn` option with multicol already loaded. You will likely run into problems.}
\msg_new:nnnn {multicolrule} {pattern-undefined} {Pattern undefined} {The multicolrule pattern `'#1'` has not been defined.}
\g__mcrule_twocolumn_bool \g__mcrule_use_tikz_bool
Flags for package options
\bool_new:N \g__mcrule_twocolumn_bool
\bool_new:N \g__mcrule_use_tikz_bool
(End definition for `\g__mcrule_twocolumn_bool` and `\g__mcrule_use_tikz_bool`)
\l__mcrule_repeat_int \l__mcrule_repeat_distance_dim
Variables to support repeated copies of the rule.
\int_new:N \l__mcrule_repeat_int
\int_set:Nn \l__mcrule_repeat_int {1}
\dim_new:N \l__mcrule_repeat_distance_dim
(End definition for `\l__mcrule_repeat_int` and `\l__mcrule_repeat_distance_dim`)
\l__mcrule_extend_top_dim \l__mcrule_extend_bot_dim \l__mcrule_extend_fill_bool \l__mcrule_extend_reserve_dim
Variables to control the distance to extend the rule above and below the natural column height.
\dim_new:N \l__mcrule_extend_top_dim
\dim_new:N \l__mcrule_extend_bot_dim
\bool_new:N \l__mcrule_extend_fill_bool
\dim_new:N \l__mcrule_extend_reserve_dim
(End definition for `\l__mcrule_extend_top_dim` and others)
\l__mcrule_color_name_tl \l__mcrule_color_model_tl
Keep name and color model so we can set them separately while retaining the value of the other one.
\tl_new:N \l__mcrule_color_name_tl
\tl_new:N \l__mcrule_color_model_tl
(End definition for `\l__mcrule_color_name_tl` and `\l__mcrule_color_model_tl`)
\g__mcrule_patterns_prop \g__mcrule_pattern_count_int \g__mcrule_pattern_for_int \g__mcrule_pattern_after_int \l__mcrule_pattern_list_seq
Variables to support defined patterns.
\prop_new:N \g__mcrule_patterns_prop
\int_new:N \g__mcrule_pattern_count_int
\int_new:N \g__mcrule_pattern_for_int
\int_new:N \g__mcrule_pattern_after_int
\seq_new:N \l__mcrule_pattern_list_seq
(End definition for `\g__mcrule_patterns_prop` and others)
If tikz is already loaded, enable tikz-sensitive line styles unless the user explicitly disables them. If tikz is not already loaded, these functions are disabled unless they are explicitly loaded.
\ifpackageloaded{tikz}
\{\bool_gset_true:N \g__mcrule_use_tikz_bool\}
\}
Set up the keys for package options and process them.
\begin{verbatim}
\keys_define:nn {mcrule-opts}
  { twocolumn .bool_gset:N = \g__mcrule_twocolumn_bool,
    tikz .bool_gset:N = \g__mcrule_use_tikz_bool,
    tikz .default:n = true,
  }
\ProcessKeysOptions{mcrule-opts}
\end{verbatim}

### 4.2 Patching Output Routines

Get the height and depth of the box appropriate to the supported mode.
\begin{verbatim}
\__mcrule_column_height:
\__mcrule_column_depth:
\end{verbatim}

\__mcrule_patch_mcol_output:N

\begin{verbatim}
\cs_new:Npn \__mcrule_patch_mcol_output:N #1
  { \xpatchcmd{#1} \columnseprulecolor\vrule\@width\columnseprule
      {\mcruledivider}
    {\msg_info:nnn {multicolrule} {patch-success} {#1}}
    {\msg_info:nnn {multicolrule} {patch-failure} {#1}}
  }
\end{verbatim}

\__mcrule_patch_twocol_output:N

The same idea as above, only for the vanilla twocolumn mode.
\begin{verbatim}
\cs_new_protected:Npn \__mcrule_patch_twocol_output:N #1
  { \xpatchcmd{#1} \normalcolor\vrule\@width\columnseprule
    \mcruledivider
    {\msg_info:nnn {multicolrule} {patch-success} {#1}}
    {\msg_info:nnn {multicolrule} {patch-failure} {#1}}
  }
\bool_if:NTF \g__mcrule_twocolumn_bool
  { \@ifpackageloaded{multicol}
    { \msg_warning:nn {multicolrule} {multicol-loaded}}{}
  }
\end{verbatim}

Provide the column-color macro from multicol.
\begin{verbatim}
\cs_gset:Npn \columnseprulecolor \normalcolor\vrule\@width\columnseprule
\cs_gset:Npn \__mcrule_column_height: \box_ht:N \@outputbox
\cs_gset:Npn \__mcrule_column_depth: \box_dp:N \@outputbox
\__mcrule_patch_twocol_output:N \@outputdblcol
\end{verbatim}
Now patch the relevant code in `\@outputdblcol`, replacing the hard-coded rule with a macro that we can overwrite.

```
\__mcrule_patch_twocol_output:N \@outputdblcol

bidi has two output routines to patch, and it insists on being loaded after xcolor, tikz, and multicol, so it must always be loaded after us. We use \AfterPackage from scrlfile to insert the patch if bidi is loaded later on.

```
\AfterPackage!{bidi}
{
\__mcrule_patch_twocol_output:N \RTL@outputdblcol
\__mcrule_patch_twocol_output:N \LTR@outputdblcol
}
```

Now patch for multicol.

```
\RequirePackage{multicol}
\__mcrule_patch_mcol_output:N \LR@column@boxes
\__mcrule_patch_mcol_output:N \RL@column@boxes

Although taking the height of \mult@rightbox is a reliable way to get the column height, the same isn’t true for the depth, so we use \dimen\tw@, which multicol uses to hold the maximum depth of all the columns, instead.

```
\cs_gset:Npn \__mcrule_column_height: {\box_ht:N \mult@rightbox}
\cs_gset:Npn \__mcrule_column_depth: {\dimen\tw@}

We need to reissue \LRmulticolcolumns to update the actual code in \mc@align@columns.

```
\LRmulticolcolumns

The bidi package supplies its own versions of most core multicol functions, including the output boxes. Much of this is unnecessary, as current versions of multicol support printing the columns in right-to-left order, and the effect is to leave the original multicol definitions loaded but unused. As a result, after these changes, the multicol commands \LRmulticolcolumns and \RLmulticolcolumns have no visible effect. But as bidi also reworks the footnotes extensively, it’s easier just to patch the equivalent output routines rather than rewrite it properly.

```
\AfterPackage!{bidi}
{
\cs_gset_eq:NN \LTR@column@boxes \LR@column@boxes
\cs_gset_eq:NN \RTL@column@boxes \RL@column@boxes

While we’re at it, we also redefine \LRmulticolcolumns and \RLmulticolcolumns so they work the way people expect them to.

```
\cs_gset_eq:NN \LRmulticolcolumns \LTRmulticolcolumns
\cs_gset_eq:NN \RLmulticolcolumns \RTLmulticolcolumns

```
4 IMPLEMENTATION

4.3 Creating the Rules

Utility functions for different rule types

\mcruledivider

This is the function directly called by the patched output routines. It has a \LaTeX{} name so the user can redefine it if necessary. Its main function is to call the internal function \mcrule_divider:, which contains the actual rule-typesetting instructions, the number of times specified in \l_mcrule_repeat_int. \multicol puts the rule in a group in order to keep the color contained, which means that any local changes here will be lost at the end of the rule. For this reason, we must set the pattern, if any, here in order to support having different line styles between different columns.

\begin{verbatim}
\cs_new_protected:Npn \mcruledivider 
{
    \IfThen\else\fi \int_compare:nNnTF \g__mcrule_pattern_after_int > \c_zero_int 
    {\int_gdecr:N \g__mcrule_pattern_after_int }
    {Don't change if the pattern is empty or the \pattern_for counter has expired. The way the logic works here, negative values of \pattern_for result in an indefinite number of repeats.
    \bool_lazy_and:nnT \int_compare_p:nNn \seq_count:N \l__mcrule_pattern_list_seq > \c_zero_int
                        \int_compare_p:nNn \g__mcrule_pattern_for_int = \c_zero_int 
    {\int_gincr:N \g__mcrule_pattern_count_int
     \int_compare:nNnT \g__mcrule_pattern_count_int > \seq_count:N \l__mcrule_pattern_list_seq
     {\int_gset:Nn \g__mcrule_pattern_count_int \c_one_int
      \tl_set:Nx \l_tmpa_tl \seq_item:Nn \l__mcrule_pattern_list_seq \g__mcrule_pattern_count_int
     \int_gdecr:N \g__mcrule_pattern_for_int }
     {\tl_if_blank:VF \l_tmpa_tl 
     \__mcrule_set_pattern:V \l_tmpa_tl
     \int_compare:nNnT \g__mcrule_pattern_for_int > \c_zero_int 
     \int_gdecr:N \g__mcrule_pattern_for_int}
     }
    
    \columnseprulecolor
    We only call \mcrule_divider: if \columnseprule > 0, so that all line styles can be turned off by setting it to 0, just as is the case with the vanilla rules.
\end{verbatim}

If the \pattern_after counter is set, wait that many iterations of the rule before we apply the patterns.

\begin{verbatim}
\int_compare:nNnTF \g__mcrule_pattern_after_int > \c_zero_int 
    {\int_gdecr:N \g__mcrule_pattern_after_int }
    {Don't change if the pattern is empty or the \pattern_for counter has expired. The way the logic works here, negative values of \pattern_for result in an indefinite number of repeats.
    \bool_lazy_and:nnT \int_compare_p:nNn \seq_count:N \l__mcrule_pattern_list_seq > \c_zero_int
                        \int_compare_p:nNn \g__mcrule_pattern_for_int = \c_zero_int 
    {\int_gincr:N \g__mcrule_pattern_count_int
     \int_compare:nNnT \g__mcrule_pattern_count_int > \seq_count:N \l__mcrule_pattern_list_seq
     {\int_gset:Nn \g__mcrule_pattern_count_int \c_one_int
      \tl_set:Nx \l_tmpa_tl \seq_item:Nn \l__mcrule_pattern_list_seq \g__mcrule_pattern_count_int
     \int_gdecr:N \g__mcrule_pattern_for_int }
     {\tl_if_blank:VF \l_tmpa_tl 
     \__mcrule_set_pattern:V \l_tmpa_tl
     \int_compare:nNnT \g__mcrule_pattern_for_int > \c_zero_int 
     \int_gdecr:N \g__mcrule_pattern_for_int}
     }
    
    \columnseprulecolor
    \end{verbatim}

Now that the pattern has been changed we can set the color.

\begin{verbatim}
\columnseprulecolor
\end{verbatim}

We only call \mcrule_divider: if \columnseprule > 0, so that all line styles can be turned off by setting it to 0, just as is the case with the vanilla rules.
\_mcrule\_divider:
\prg\_replicate:nn {\_\_mcrule\_repeat\_int - \c\_one\_int}
{
\hspace{\_\_mcrule\_repeat\_distance\_dim}
\mcrule\_divider:
}
}

(End definition for \mcrule\_divider. This function is documented on page ??.)

\_\_mcrule\_column\_total\_height:
\_\_mcrule\_column\_total\_depth:

Get column height and depth with any explicit alterations.
\cs\_new:Npn \_\_mcrule\_column\_total\_height:
{
\dim\_eval:n {\_\_mcrule\_column\_height: + \_\_mcrule\_column\_depth: +
\_\_mcrule\_extend\_column\_top: + \_\_mcrule\_extend\_column\_bottom:}
}
\cs\_new:Npn \_\_mcrule\_column\_total\_depth:
{
\dim\_eval:n {\_\_mcrule\_column\_depth: + \_\_mcrule\_extend\_column\_bottom:}
}

\_\_mcrule\_extend\_column\_top:

Currently, the extend amount for the top is just the \l\_@@\_extend\_top\_dim distance. In the future we may allow more complex criteria, such as by odd or even page, or on a particular page. Although these might theoretically be useful, I'm not going to implement them until someone comes along with a use-case for it.
\cs\_new:Npn \_\_mcrule\_extend\_column\_top:
{
\bool\_lazy\_and:nnTF
{\bool\_if\_p:n {\l\_\_mcrule\_extend\_fill\_bool}}
{\bool\_not\_p:n {\g\_\_mcrule\_twocolumn\_bool}}
{
\dim\_compare:nNnTF
{\@colroom - \_\_mcrule\_column\_height: - \_\_mcrule\_extend\_reserve:} > \{\c\_zero\_dim
{\@colroom - \_\_mcrule\_column\_height: - \_\_mcrule\_extend\_reserve:}
{\c\_zero\_dim}
}
{\l\_\_mcrule\_extend\_top\_dim}
}

\_\_mcrule\_extend\_column\_bottom:

The extend-fill option, which is only applicable with multicol, extends the rule from the bottom of the column to the end of the text area, minus whatever reserved space the user specifies. If there's less space available than requested, we give everything we can.
\cs\_new:Npn \_\_mcrule\_extend\_column\_bottom:
{
\bool\_lazy\_and:nnTF
{\bool\_if\_p:n {\l\_\_mcrule\_extend\_fill\_bool}}
{\bool\_not\_p:n {\g\_\_mcrule\_twocolumn\_bool}}
{
\dim\_compare:nNnTF
{\@colroom - \_\_mcrule\_column\_height: - \_\_mcrule\_extend\_reserve=} > \{\c\_zero\_dim
{\@colroom - \_\_mcrule\_column\_height: - \_\_mcrule\_extend\_reserve:}
{\c\_zero\_dim}
}
{\l\_\_mcrule\_extend\_bot\_dim}
}
The reserved space is the amount of user-provided space we want, but we also have to account for the space added with \multicolsep.

\cs_new:Npn \_mcrule_extend_reserve:
{
\dim_compare:nNnTF {\l__mcrule_extend_reserve_dim} > {\c_zero_dim}
{\dim_eval:n {\l__mcrule_extend_reserve_dim + \multicolsep}}
{\c_zero_dim}
}

\__mcrule_extend_reserve:
This is the routine that contains the instructions to draw one copy of rule between columns. The default is identical to the original definition used by multicol. It will be reset each time the user calls \MCSetRule to specify a new line style.

\cs_new:Npn \mcrule_divider: {\vrule\@width\columnseprule}
\mcrule_divider:
\__mcrule_pattern:nnn {⟨pattern⟩} {⟨space above⟩} {⟨space below⟩}
Typesets a single copy of a pattern, vertically centered, in a vertical box that is the height of the current column. The pattern must be something that can go in a horizontal box. The spaces above and below must be fixed dimensions.

\cs_new_nopar:Npn \__mcrule_pattern:nnn #1#2#3
{\box_move_down:nn {\__mcrule_column_total_depth:}
{\vbox_to_ht:nn {\__mcrule_column_total_height:}
{\tex_vfill:D \tex_kern:D #2 \hbox:n{#1} \tex_kern:D #3 \tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
}\tex_kern:D #2 \hbox:n(#1) \tex_kern:D #3
}\tex_vfill:D
This function can draw a line pattern using either a tikz name or directly (as a tiled pattern). The latter case is currently limited to line patterns that can be described in terms of a solid line of length \((\text{height})\) separated by spaces above and/or below the line.

\begin{verbatim}
\cs_new:Npn \_mcrule_line_pattern:nnnn #1#2#3#4
{\bool_if:NTF \g_mcrule_use_tikz_bool
 {\__mcrule_pattern_line:n {#1}}
 {\__mcrule_tile_pattern:nnn \rule{\columnseprule}{#2}}{#3}{#4}
}
\end{verbatim}

Unlike the default solid line, which is created with a simple \texttt{\vrule}, this version allows us to extend the line beyond the natural space of the column.

\begin{verbatim}
\cs_new:Npn \_mcrule_solid_line:
{\rule[-\__mcrule_column_total_depth:]{\columnseprule}\__mcrule_column_total_height:}
\end{verbatim}

\subsection*{4.3.1 Tikz-only Routines}

If we’re supporting tikz, make sure it’s loaded and redefine the relevant functions. We turn off expl3 syntax to load the package because tikz relies on 2e catcodes, especially for spaces.

\begin{verbatim}
\bool_if:NTF \g_mcrule_use_tikz_bool
 {\ExplSyntaxOff \RequirePackage{tikz} \ExplSyntaxOn}
\end{verbatim}

\begin{verbatim}
\cs_set:Npn \_mcrule_tikz_picture:n #1
{\begin{tikzpicture}[x=1pt,y=1pt,inner-sep=0pt,outer-sep=0pt, baseline=([yshift=\__mcrule_column_total_depth:]\currentboundingbox.south)]
 \node (TOP) at (0,\__mcrule_column_total_height:) {};
 \node (BOT) at (0,0) {};
 #1
 \end{tikzpicture}}
\end{verbatim}
For the \texttt{tikz} versions of the predefined lines, we just draw a line the length of the column box. \texttt{\langle tikz pattern \rangle} should contain the name of a line style that \texttt{tikz} recognizes.

\begin{verbatim}
\cs_set:Npn \__mcrule_pattern_line:n #1
{\begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt,
  baseline={([yshift=\__mcrule_column_total_depth:]current~bounding~box.south)}]
\draw[line~width=\columnseprule,#1] (0,\__mcrule_column_total_height:) -- (0,0);
\end{tikzpicture}}
\end{verbatim}

\texttt{\__mcrule_pattern_line:n} \texttt{\langle tikz pattern \rangle}

\begin{verbatim}
\cs_set:Npn \__mcrule_circle:
{\begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt]
\draw (0,0) circle[radius=.5\columnseprule];
\end{tikzpicture}}
\end{verbatim}

\texttt{\__mcrule_circle:}

\begin{verbatim}
\cs_set:Npn \__mcrule_solid_circle:
{\begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt]
\fill (0,0) circle[radius=.5\columnseprule];
\end{tikzpicture}}
\end{verbatim}

\texttt{\__mcrule_solid_circle:}

\texttt{\__mcrule_tikz_picture:n} \texttt{\langle tikz pattern \rangle}

\begin{verbatim}
\cs_set:Npn \__mcrule_tikz_picture:n #1
{\msg_error:nnn {multicolrule} {tikz-required} {#1}}
\cs_new:Npn \__mcrule_pattern_line:n #1
{\msg_error:nnn {multicolrule} {tikz-required} {#1}}
\cs_new:Npn \__mcrule_circle:
{\msg_error:nnn {multicolrule} {tikz-required} {circles}}
\cs_new:Npn \__mcrule_solid_circle:
{\msg_error:nnn {multicolrule} {tikz-required} {solid-circles}}
\end{verbatim}

\texttt{\__mcrule_tikz_picture:n} \texttt{\langle tikz pattern \rangle}

In case \texttt{tikz} functions are not active, we provide stubs that issue error messages.
4 IMPLEMENTATION

4.4 Color

\__mcrule_set_rule_color:
Reset color definition in \columnseprulecolor by name or by model and color specification.

\cs_new_protected:Npn \__mcrule_set_rule_color:n #1
\seq_set_split:Nnn \l__mcrule_pattern_list_seq {,} {#1}
\int_gzero:N \g__mcrule_pattern_count_int
\int_gzero:N \g__mcrule_pattern_after_int
\int_gset:Nn \g__mcrule_pattern_for_int {-1}
\__mcrule_set_pattern:n #1
\cs_generate_variant:Nn \__mcrule_set_pattern:n {V}
\__mcrule_set_pattern:n

4.5 Patterns

\__mcrule_set_pattern_list:n
Sets a comma-separated list of patterns as a sequence for later use. The global counter that indicates where we are in the list is also reset here, so setting a list of patterns always means that the next rule will use the first pattern in the list.

\cs_new_protected:Npn \__mcrule_set_pattern_list:n #1
\seq_set_split:Nnn \l__mcrule_pattern_list_seq {,} {#1} [#1]
\int_gzero:N \g__mcrule_pattern_count_int
\int_gzero:N \g__mcrule_pattern_after_int
\int_gset:Nn \g__mcrule_pattern_for_int {-1}
\__mcrule_set_pattern:n #1
\cs_generate_variant:Nn \__mcrule_set_pattern:n {V}
\__mcrule_set_pattern:n
4.6 Key-Values

Set up all the key definitions. For the line styles, this involves resetting \mcrule_divider: to an appropriate value.

\keys_define:nn {mrule}
{
  extend-top \dim_set:N = \l__mcrule_extend_top_dim,
  extend-bot \dim_set:N = \l__mcrule_extend_bot_dim,
  extend-fill \bool_set:N = \l__mcrule_extend_fill_bool,
  extend-fill \default:n = true,
  extend-reserve \dim_set:N = \l__mcrule_extend_reserve_dim,
  line-style .choice:,
  line-style / default .code:n = \cs_set:Npn \mcrule_divider: {\vrule\@width\columnseprule},
  line-style / solid .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_solid_line:},
  line-style / dots .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_tile_pattern:nnn {.}{1pt}{1pt}},
  line-style / dense-dots .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_tile_pattern:nnn {.}{1pt}{0pt}},
  line-style / loose-dots .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_tile_pattern:nnn {.}{2pt}{2pt}},
  line-style / circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_circle:}{1pt}{1pt}},
  line-style / dense-circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_circle:}{1pt}{0pt}},
  line-style / loose-circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_circle:}{2pt}{2pt}},
  line-style / solid-circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_solid_circle:}{1pt}{1pt}},
  line-style / dense-solid-circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_solid_circle:}{1pt}{0pt}},
  line-style / loose-solid-circles .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_solid_circle:}{2pt}{2pt}},
  line-style / dotted .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {dotted}{\columnseprule}{1pt}{1pt}},
  line-style / densely-dotted .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {densely-dotted}{\columnseprule}{1pt}{0pt}},
  line-style / loosely-dotted .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {loosely-dotted}{\columnseprule}{2pt}{2pt}},
  line-style / dashed .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {dashed}{\columnseprule}{1.5pt}{1.5pt}},
  line-style / densely-dashed .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {densely-dashed}{\columnseprule}{3pt}{3pt}},
  line-style / loosely-dashed .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_line_pattern:nnn {loosely-dashed}{\columnseprule}{3pt}{3pt}},
  line-style / dash-dot .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_pattern_line:n{dash-dot}},
  line-style / densely-dash-dot .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_pattern_line:n{densely-dash-dot}},
  line-style / loosely-dash-dot .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_pattern_line:n{loosely-dash-dot}},
  line-style / dash-dot-dot .code:n = \cs_set:Npn \mcrule_divider: {\__mcrule_pattern_line:n{dash-dot-dot}},
}
4 IMPLEMENTATION

4.7 User Interface

\SetMCRule Set all keys for multicolrule
Change History

\SetMCRule \{⟨key-value list⟩\}

All we do here is pass the argument to expl3’s key-setting routine.

\NewDocumentCommand{\SetMCRule}{m} \{ \keys_set:nn \{mcrule\} \{#1\} \}

(End definition for \SetMCRule. This function is documented on page ??.)

\DeclareMCRulePattern

Declare a new style pattern.

\DeclareMCRule \{(name)\} \{⟨key-value list⟩\}

If a pattern of that name exists, it will be overwritten silently.

\NewDocumentCommand{\DeclareMCRulePattern}{m m} \{ \prop_gput:Nnn \g__mcrule_patterns_prop \{#1\} \{#2\} \}

(End definition for \DeclareMCRulePattern. This function is documented on page ??.)

Change History

<table>
<thead>
<tr>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>General: Initial public release</td>
</tr>
<tr>
<td></td>
<td>v1.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>v1.2</td>
<td>General: Added patterns, pattern-after,</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A
\\\AfterPackage ................................ 13

C
\\\cleaders .................................... 3
\\\color ........................................ 258, 262
\\\columnsep .................................... 2
\\\columnseprule ................................ 2, 3,
\6, 8, 14, 50, 57, 122, 165, 199, 204, 224,
\\\columnseprulecolor .......................... 6, 7, 12, 19, 22, 50, 66, 120, 258, 261

D
\\\DeclareMCRulePattern ....................... 9
\\\DefineMCRulePattern ....................... 9, 387
\\\definecolor .................................. 7
\\\draw ........................................ 7, 17, 224, 230
Index

\HandRight ...................... 7

int commands:
\int_compare:nNnTF ............. 94, 104, 114

keys commands:
\keys_set_filter:nnn ............ 276

\LTrmulticolcolumns ........... 13, 13, 13, 83, 88
\LTRmulticolcolumns ........... 88

mcrule commands:

mcrule internal commands:
\__mcrule_circle: ................ 18, 227, 245, 303, 305, 307
\__mcrule_color_model_tl ...... 28, 256, 262, 343
\__mcrule_color_name_tl ...... 28, 252, 254, 258, 262, 339
\__mcrule_column_depth: ........ 12, 47, 68, 82, 135, 140
\__mcrule_column_height: ...... 12, 46, 67, 81, 135, 153, 154
\__mcrule_column_total_depth: \int, 138, 168, 180, 204, 214, 223
\__mcrule_column_total_height: \int, 133, 170, 182, 204, 215, 224
\_\_mcrule_extend_bot_dim ....... 24, 157, 287
\_\_mcrule_extend_column_bottom: \int, 136, 140, 146
\_\_mcrule_extend_column_top: \int, 136, 142
\_\_mcrule_extend_fill_bool .. \int, 149, 288
\_\_mcrule_extend_reserve: ... \int, 153, 154, 159
\_\_mcrule_extend_reserve_dim \int, 161, 162, 190
\_\_mcrule_extend_top_dim ... \int, 144, 286
\_\_mcrule_line_pattern:nnnn ... \int, 17, 192, 315, 317, 319, 321, 323, 325
\_\_mcrule_patch_mcol_output:N \int, 12, 48, 79, 80
\_\_mcrule_patch_twocol_output:N \int, 12, 55, 69, 70, 73, 74
\_\_mcrule_pattern:nnn ........ \int, 16, 166, 349
\g__mcrule_pattern_after_int ... \int, 30, 94, 96, 269, 380
\g__mcrule_pattern_count_int ... \int, 30, 103, 104, 107, 109, 268
\g__mcrule_pattern_for_int ... \int, 30, 101, 114, 116, 270, 381
\_\_mcrule_pattern_line:n ... \int, 78, 196, 220, 243, 327, 329, 331, 333, 335, 337
\l__mcrule_pattern_list_seq .. \int, 30, 100, 103, 109, 267
\g__mcrule_patterns_prop ....... \int, 30, 274, 281, 389
\_\_mcrule_repeat_distance_dim \int, 21, 128, 362
\_\_mcrule_repeat_int ........ \int, 14, 21, 123, 126, 361
\_\_mcrule_set_pattern:n ...... \int, 19, 112, 272, 283
\_\_mcrule_set_pattern_list:n \int, 19, 265, 378
\_\_mcrule_set_rule_color: .... \int, 19, 250, 340, 344
\_\_mcrule_solid_circle: ....... \int, 18, 233, 247, 309, 311, 313
\_\_mcrule_solid_line: .. \int, 17, 202, 295
\_\_mcrule_tikz_picture:n ..... \int, 17, 211, 241, 347
\_\_mcrule_tile_pattern:nnn ... \int, 16, 178, 199, 297, 299, 301, 303, 305, 307, 309, 311, 313, 351
\g__mcrule_twocolumn_bool ... \int, 19, 41, 62, 150
\g__mcrule_use_tikz_bool ....... \int, 19, 37, 42, 194, 206
\mcruledivider ........ \int, 12, 22, 51, 58, 92
\MCSetRule ................... \int, 16
\multicolsep .................. \int, 9, 16, 162

options:
\tikz \int, 3
\twocolumn \int, 4

\path ......................... 7

\RLmulticolcolumns ........... 13, 13, 89
\RTLmulticolcolumns ........... 89

\SetMCRule ................... 4
<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\SetMCRule</td>
<td>2, 4, 6, 9, 383</td>
</tr>
<tr>
<td>\SparkleBold</td>
<td>7</td>
</tr>
<tr>
<td>\multirightbox</td>
<td>13, 81</td>
</tr>
<tr>
<td>\column@boxes</td>
<td>80, 87</td>
</tr>
<tr>
<td>\output@dblcol</td>
<td>73</td>
</tr>
<tr>
<td>\vrule</td>
<td>17</td>
</tr>
<tr>
<td>\mc@align@columns</td>
<td>13</td>
</tr>
</tbody>
</table>

**TEX and \LaTeX commands:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>@colroom</td>
<td>153, 154</td>
</tr>
<tr>
<td>@outputbox</td>
<td>67, 68</td>
</tr>
<tr>
<td>@output@dblcol</td>
<td>69, 70</td>
</tr>
<tr>
<td>\LR@column@boxes</td>
<td>79, 86</td>
</tr>
<tr>
<td>\LTR@column@boxes</td>
<td>86</td>
</tr>
<tr>
<td>\LTR@output@dblcol</td>
<td>74</td>
</tr>
<tr>
<td>\MC@align@columns</td>
<td>13</td>
</tr>
</tbody>
</table>

**\TeX commands:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tex_cleaders:D</td>
<td>184</td>
</tr>
<tr>
<td>\tex_kern:D</td>
<td>173, 186</td>
</tr>
<tr>
<td>\tex_vfill:D</td>
<td>172, 174, 188</td>
</tr>
</tbody>
</table>

**tikz (option)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tikz (option)</td>
<td>3</td>
</tr>
</tbody>
</table>

**twocolumn (option)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\twocolumn</td>
<td>1, 4</td>
</tr>
</tbody>
</table>