

multicolrule — Decorative rules between columns^{*}

Karl Hagen[†]

Released 2019/10/01

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’	4
3	The User Interface	4
3.1	Styles with the ‘line-style’ option	5
3.1.1	Custom Patterns	7
3.2	Colors	8
3.3	Width	8
3.4	Repeated Rules	8
3.5	Extended Rules	9
3.6	Rule Patterns	10
4	Implementation	11
4.1	Preliminaries	11
4.2	Patching Output Routines	13
4.3	Creating the Rules	15
4.3.1	Tikz-only Routines	19
4.4	Color	21
4.5	Patterns	21
4.6	Key-Values	22
4.7	User Interface	24

^{*}This file describes version v1.3, last revised 2019/10/01.

[†]latex@polysyllabic.com

Change History

24

Index

25

1 Introduction`line-style=dashed`

In \LaTeX , there are two lengths that control the formatting between columns of multicolumn text: `\columnsep` specifies the space between adjacent columns, and `\columnseprule` specifies the width of a solid vertical rule that is placed centered between the columns. The `multicol` package adds the ability to change the color of the rule, but in both vanilla \LaTeX and `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.¹ 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 `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 `mcrule-example.pdf`.

New for Version 1.3

Version 1.3 adds a strut rule. This places an invisible rule (a strut) that obeys the other options for the rule. Using a strut allows you to achieve effects such as selectively disabling one separator within a pattern or using the `extend` and `reserve` options to control the spacing to the text outside the `multicols` environment.

New for Version 1.2

Version 1.2 adds the ability to define patterns, which are aliases for a series of `\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 3.6).

New for Version 1.1

Version 1.1 supports drawing decorative rules if you have the `bidi` package loaded, which can occur automatically if you set a right-to-left language with `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 3.5).

1.1 Bugs and Known Limitations`line-style=dots`

The `multicolrule` package is written using `expl3` syntax, and so requires a less-than-ancient installation of \LaTeX . It requires the packages `l3keys2e`, `xparse`, `xpatch`, `xcolor`, `scrfile`, and depending on the mode of op-

eration may also require `multicol` and `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

¹See, for example, the remarks in the documentation for the `booktabs` package.

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 `tikz`.

The following are the issues I'm currently aware of that aren't `multicolrule` errors but which may cause buggy looking behavior:

This package works by patching the output routines of either `multicol` or the \TeX 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 affects the vertical layout was a deliberate design decision and is necessary to support the `extend-fill` and `extend-reserve` options. A future version may support drawing the rules to a background layer so that the text is not shifted.

1.2 License

`line-style=dotted,`
`width=ultra-thick`

The `multicolrule` package is copyright 2018–2019 by Karl Hagen. It may be distributed and/or modified under the conditions of the \TeX 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>.

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

2 Package Options

2.1 Default Operation

`line-style=dash-dot`

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 \TeX 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 package. 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` is only designed to work with one or the other at a time. If you try to use the `twocolumn` option when `multicol` has already been loaded, you will receive a warning and nothing is guaranteed. But the custom rules will at best only appear in the conventional two-column mode and not within a `multicols` environment.

3 The User Interface

`\SetMCRule` `\SetMCRule <key-value list>`

`line-style=circles,`
`width=2pt`

The main user command for `multicolrule` is `\SetMCRule`. It takes one parameter containing a key-value list of all options you want to set. You can issue this command in the preamble or the document body. Changes to the rule settings are local to the current group. For example, if you call `\SetMCRule` inside a `multicols` environment, the rule settings will revert to their previous values once the environment ends. Also note that any changes made with `\SetMCRule` when

multiple columns are active will appear starting on the same page as your current location when you issue the command, and will extend the height of the full column box. It is not possible to have a rule change styles in the middle of a page unless you close out one `multicols` environment and begin another.

Table 1 summarizes the keys available in `\SetMCRule`. The functions of each is described in detail in the sections that follow.

Table 1: `\SetMCRule` keys

Key	Purpose
<code>color</code>	Set the color of the rule (see sec. 3.2)
<code>color-model</code>	Set the color model of the rule (see sec. 3.2)
<code>custom-line</code>	Set a custom tikz line for the rule (<i>tikz only</i> ; see sec. 3.1.1)
<code>custom-pattern</code>	Set a custom individual pattern for the rule (see sec. 3.1.1)
<code>custom-tile</code>	Set a custom tiling pattern for the rule (see sec. 3.1.1)
<code>double</code>	Draw two copies of the rule (see sec. 3.4)
<code>expand</code>	Change the extend distance by the same amount at the top and bottom of the column (see sec. 3.5)
<code>extend-bot</code>	Set an extra amount to extend the rule at the bottom of the column (see sec. 3.5)
<code>extend-fill</code>	Extend rule to the bottom of the text area (<i>multicol only</i> ; see sec. 3.5)
<code>extend-reserve</code>	Space to reserve at bottom of text area when using <code>extend-fill</code> (<i>multicol only</i> ; see sec. 3.5)

Table 1: \SetMCRule keys (cont.)

Key	Purpose
<code>extend-top</code>	Set an extra amount to extend the rule at the top of the column (see sec. 3.5)
<code>line-style</code>	Select the type of rule printed (default= <i>default</i> ; see sec. 3.1)
<code>pattern-after</code>	Number of separators to delay before beginning to use the specified patterns (default=0; see sec. 3.6)
<code>pattern-for</code>	Number times separators to apply the patterns to before returning to default (default=-1; see sec. 3.6)
<code>patterns</code>	Specify one or more patterns use to draw rules. (default= <i>none</i> ; see sec. 3.6)
<code>single</code>	Draw a single copy of the rule (<i>default</i> ; see sec. 3.4)
<code>repeat</code>	Set the number of times to draw the rule (see sec. 3.4)
<code>repeat-distance</code>	Set the horizontal space between adjacent copies of repeated rules (see sec. 3.4)
<code>shift</code>	Shift the extend distance downward; this affects both the top and bottom of the column (see sec. 3.5)
<code>triple</code>	Draw three copies of the rule (see sec. 3.4)
<code>width</code>	Set the width of the rule (see sec. 3.3)

3.1 Styles with the ‘line-style’ option

`line-style=solid-circles, width=4pt`

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 L^AT_EX 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

Style	Description
<code>circles</code>	A series of hollow circles (<i>tikz only</i>)
<code>dash-dot</code>	A dash followed by a square dot (<i>tikz only</i>)
<code>dash-dot-dot</code>	A dash followed by two square dots (<i>tikz only</i>)
<code>dashed</code>	A series of dashed lines
<code>default</code>	A solid rule drawn the same way as the default multicol rule. Does not support extended rules.
<code>dense-circles</code>	The same as <code>circles</code> but more closely spaced (<i>tikz only</i>)
<code>dense-dots</code>	The same as <code>dots</code> but more closely spaced

Table 2: Available line-style settings (cont.)

Style	Description
<code>dense-solid-circles</code>	The same as <code>solid-circles</code> but more closely spaced (<i>tikz only</i>)
<code>densely-dash-dot</code>	The same as <code>dash-dot</code> but more closely spaced (<i>tikz only</i>)
<code>densely-dash-dot-dot</code>	The same as <code>dash-dot-dot</code> but more closely spaced (<i>tikz only</i>)
<code>densely-dashed</code>	The same as <code>dashed</code> but more closely spaced
<code>densely-dotted</code>	The same as <code>dotted</code> but more closely spaced
<code>dots</code>	A series of dots drawn with the period (full-stop) of the current font
<code>dotted</code>	A series of square dots
<code>loose-dots</code>	The same as <code>dots</code> but spaced further apart
<code>loose-circles</code>	The same as <code>circles</code> but spaced further apart (<i>tikz only</i>)
<code>loose-solid-circles</code>	The same as <code>solid-circles</code> but spaced further apart (<i>tikz only</i>)
<code>loosely-dash-dot</code>	The same as <code>dash-dot</code> but spaced further apart (<i>tikz only</i>)
<code>loosely-dash-dot-dot</code>	The same as <code>dash-dot-dot</code> but spaced further apart (<i>tikz only</i>)
<code>loosely-dashed</code>	The same as <code>dashed</code> but spaced further apart
<code>loosely-dotted</code>	The same as <code>dotted</code> but spaced further apart
<code>solid</code>	A solid line, like <code>default</code> , but supports extending rules
<code>solid-circles</code>	A series of filled circles (<i>tikz only</i>)
<code>strut</code>	An invisible (0pt) strut is drawn in place of a solid rule.

`line-style=solid`

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 ren-

dered 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`.

The `strut` style draws a 0pt rule. Like every other line style, however, the value of `\columnseprule` must be greater than 0pt for it to be drawn. An invisible rule can be useful if you want to disable a rule in the middle of a cycle of patterns or in conjunction with the `extend` various `extend` options. See section 3.5.

3.1.1 Custom Patterns

`custom-tile = {⟨pattern⟩} {⟨space above⟩} {⟨space below⟩}`

```
custom-tile=
{\SparkleBold}
{16pt}{16pt}
```

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 `\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.



`custom-pattern = {⟨pattern⟩} {⟨shift down⟩} {⟨shift up⟩}`

```
custom-pattern= {\Hand-
dRight} {0pt}{0pt}
```

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 cen-



tered 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 `\HandRight` symbol from `bbding`.

`custom-line = {⟨draw command⟩}`

```
custom-line={\path
(TOP) to [orna-
ment=88] (BOT);}
```

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 `\draw` or `\path`, without the surrounding `tikzpicture` environment.

Before the drawing command is called, `multicolrule` will set up a `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 `\draw` or `\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 `\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 nothing limits you to drawing a picture that fits within the space between the columns. If the rule is wider than the available space, it will be centered between the columns and overlap the text. Normally, of course, that will be undesirable, but you can use it to your advantage in certain cases. The file `mcrule-example.pdf` contains examples showing the effect of a rule that is too wide, as well as a custom rule which includes horizontal rules at the top and bottom of the column.²

²This latter rule was developed as an answer to [StackExchange question 473828](#).

3.2 Colors

```
line-style=solid,
width=2pt
color-model=cmy,
color={0.7,0.5,0.3}
```

You can set colors for the rule through the `color` and, optionally, the `color-model` keys. `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 `\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 `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 `\columnseprulecolor`. If you are running in `twocolumn` mode without `multicol`, this command will be provided and colors will work the same way they do with `multicol`. Note that setting the `color` key causes `\columnseprulecolor` to be redefined within the current group only. If you directly redefine `\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

```
line-style= dash-dot-dot,
width=thick
```

You can set the width of the rule with the `width` key. Legal values are any explicit dimension or dimension expression, as well as one of the names listed in table 3. These names 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 `\columnseprule`, just as in vanilla \TeX ,

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 `\columnseprule` to calculate their actual width, the value of `\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 3: Sizes of named line widths

Name	Width
<code>ultra-thin</code>	0.1pt
<code>very-thin</code>	0.2pt
<code>thin</code>	0.4pt
<code>semithick</code>	0.6pt
<code>thick</code>	0.8pt
<code>very-thick</code>	1.2pt
<code>ultra-thick</code>	1.6pt

3.4 Repeated Rules

```
line-style= dash-dot-dot,
triple=2pt
```

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 `\columnseprule`. Note that you must enter an actual dimension expression for this distance. The names used for line widths are not accepted.

The keys `single`, `double`, and `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 `repeat-distance` is set to `\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

`line-style=dashed,`
`expand=-16pt`

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 `extend-top` and `extend-bot`, 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 `extend-bot` 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.

Note that positive values for extending the rules should be used with caution and only in situations where you need a special effect for one column or a small `multicols` environment. (See section 3.6 for a way to limit the change to one or a few columns.)

`line-style=solid,`
`extend-fill`

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 is only relevant for the `multicols` environment. It will have no effect with either `multicols*` or the plain \TeX two-column mode.

If you want text below and on the same page as the `multicols` environment when using `extend-fill`, you can reserve space for it with `extend-reserve`, which takes a dimension expression specifying the vertical

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 `expand` and `shift` keys provide shorthands for two common situations. You can use `expand` to set the same value for `extend-top` and `extend-bot`. For example, `expand=-16pt` is equivalent to `extend-bot=-16pt`, `extend-top=-16pt`. The `shift` key moves the rule downward for positive values and upward for negative ones without changing the overall length of the rule. More precisely, `shift=x` translates to `extend-bot=x`, `extend-top=-x`. For example, `shift=16pt` is equivalent to `extend-bot=16pt`, `extend-top=-16pt`.

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 `multicols` 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 {<name>} {<key-value list>}
```

`patterns={right-hand,
left-hand}`
See the code sample
below for the defini-
tions of the patterns

A *pattern* refers to a bundle of settings used by **multicolrule**. Although you can use patterns as a shortcut to save you a little typing, their main purpose is to let you to alter individual rules within a multicol environment. For example, if you have three-column text, you can make the left rule different from the right one. In two-column text, you can have different rules for alternating pages.

You declare a pattern for a line style with the command `\DeclareMCRulePattern`. The `<name>` should consist of letters and hyphens only. The `<key-value list>` can contain all keys that are valid for `\SetMCRule`

with the exception of **patterns**, which is filtered out. In other words, if you put something like `patterns=foo` in the pattern definition, 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 the prior settings. 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 that the patterns do not cycle within a single column separator if you use the **repeat** key. This cycle is global, so if the number column separators 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 with the first pattern in the list.

The columns above were defined with the following commands:

```
\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}
```

`patterns=shrink-me,
pattern-for=1`
See the code sample
below for the defini-
tion of 'shrink-me'

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.³ You could accomplish this task with the code below.

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 definition of the pattern, even if they come after the `patterns` key. Such settings will take effect before the pattern is applied and will reappear after the pattern ends, if it does.

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

```
\DeclareMCRulePattern{shrink-me}{line-style=solid,
  extend-top=-3\baselineskip}
\begin{multicols}{4}
  \SetMCRule{patterns=shrink-me,pattern-after=1,pattern-for=2}
  ...
\end{multicols}
```

4 Implementation

```
1 <*package>
2 <@@=mcrule>
```

4.1 Preliminaries

```
3 \ProvidesExplPackage {multicolrule} {2019/10/01} {1.3}
4 {Decorative vertical rules between columns}
```

We always need these packages.

```
5 \RequirePackage{l3keys2e}
6 \RequirePackage{xpatch}
7 \RequirePackage{xcolor}
8 \RequirePackage{scrfile}
```

Define the messages we use.

```
9 \msg_new:nnn {multicolrule} {patch-success} {Patched~#1.}
10 \msg_new:nnn {multicolrule} {patch-failure} {Error~patching~#1.}
11 \msg_new:nnnn {multicolrule} {tikz-required} {Tikz~required}
12 {The~'~#1'~setting~requires~tikz~to~work.~Either~load~tikz~before~you~load~
13  multicolrule~or~use~multicolrule's~'tikz'~package~option.}
14 \msg_new:nnnn {multicolrule} {multicol-loaded} {Multicol~loaded} {You~are~
15  using~the~'twocolumn'~option~with~multicol~already~loaded.~You~will~likely~
16  run~into~problems.}
17 \msg_new:nnnn {multicolrule} {pattern-undefined} {Pattern~undefined}
18 {The~multicolrule~pattern~'~#1'~has~not~been~defined.}
```

Flags for package options

```
\g__mcrule_twocolumn_bool
\g__mcrule_use_tikz_bool
19 \bool_new:N \g__mcrule_twocolumn_bool
20 \bool_new:N \g__mcrule_use_tikz_bool
```

(End definition for `\g__mcrule_twocolumn_bool` and `\g__mcrule_use_tikz_bool`.)

³Remember that you have one less column separator than you have columns.

`\l__mcrule_repeat_int` Variables to support repeated copies of the rule.

```

\l__mcrule_repeat_distance_dim
21 \int_new:N \l__mcrule_repeat_int
22 \int_set:Nn \l__mcrule_repeat_int {1}
23 \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` Variables to control the distance to extend the rule above and below the natural column height.

```

\l__mcrule_extend_bot_dim
\l__mcrule_extend_fill_bool
\l__mcrule_extend_reserve_dim
24 \dim_new:N \l__mcrule_extend_top_dim
25 \dim_new:N \l__mcrule_extend_bot_dim
26 \bool_new:N \l__mcrule_extend_fill_bool
27 \dim_new:N \l__mcrule_extend_reserve_dim

```

(End definition for `\l__mcrule_extend_top_dim` and others.)

`\l__mcrule_color_name_tl` Keep name and color model so we can set them separately while retaining the value of the other one.

```

\l__mcrule_color_model_tl
28 \tl_new:N \l__mcrule_color_name_tl
29 \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` Variables to support defined patterns.

```

\g__mcrule_pattern_count_int
\g__mcrule_pattern_for_int
\g__mcrule_pattern_after_int
\l__mcrule_pattern_list_seq
30 \prop_new:N \g__mcrule_patterns_prop
31 \int_new:N \g__mcrule_pattern_count_int
32 \int_new:N \g__mcrule_pattern_for_int
33 \int_new:N \g__mcrule_pattern_after_int
34 \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.

```

35 \@ifpackageloaded{tikz}
36 {
37   \bool_gset_true:N \g__mcrule_use_tikz_bool
38 }{}

```

Set up the keys for package options and process them.

```

39 \keys_define:nn {mcrule-opts}
40 {
41   twocolumn .bool_gset:N = \g__mcrule_twocolumn_bool,
42   tikz      .bool_gset:N = \g__mcrule_use_tikz_bool,
43   tikz      .default:n   = true,
44 }
45 \ProcessKeysOptions{mcrule-opts}

```

4.2 Patching Output Routines

Now that we know what mode we're going to run in, we patch the output routine(s) to substitute our custom rule for the vanilla one. Since multicol doesn't fully support twocolumn mode, we patch one or the other, but not both.

First we set some stubs for functions we'll need to redirect depending on the mode we're operating in.

`__mcrule_column_height:` Returns the fixed height of the columns. The actual code to calculate the height is set when we set the appropriate mode.

```
46 \cs_new:Npn \__mcrule_column_height: {}
```

(End definition for __mcrule_column_height:.)

`__mcrule_column_depth:` Returns the maximum depth of the columns. The actual code to calculate the depth is set when we set the appropriate mode.

```
47 \cs_new:Npn \__mcrule_column_depth: {}
```

(End definition for __mcrule_column_depth:.)

`__mcrule_column_overflow:` Returns the amount by which text overflows the bottom of the columns. This situation occurs in multicol when `\maxbalancingoverflow` is greater than 0 (by default it's 12pt) and there would otherwise be a widow at the end of the environment, so we don't check for it in two-column mode.

```
48 \cs_new:Npn \__mcrule_column_overflow: {0pt}
```

(End definition for __mcrule_column_overflow:.)

`__mcrule_patch_mcol_output:N` We create a helper macro to simplify patching the appropriate part of the relevant multicol routines. The search and replace texts are identical across several routines, so only the name of the function to be patched needs to be passed as a parameter. We make `\columnseprulecolor` part of `\mcruledivider` so that we can set the color as part of a style pattern.

```
49 \cs_new_protected:Npn \__mcrule_patch_mcol_output:N #1
50 {
51   \xpatchcmd{#1} {\columnseprulecolor\vrule\@width\columnseprule}
52   {\mcruledivider}
53   {\msg_info:nnn {multicolrule} {patch-success} {#1}}
54   {\msg_info:nnn {multicolrule} {patch-failure} {#1}}
55 }
```

(End definition for __mcrule_patch_mcol_output:N.)

`__mcrule_patch_twocol_output:N` The same idea as above, only for the vanilla twocolumn mode.

```
56 \cs_new_protected:Npn \__mcrule_patch_twocol_output:N #1
57 {
58   \xpatchcmd{#1} {\normalcolor\vrule\@width\columnseprule}
59   {\mcruledivider}
60   {\msg_info:nnn {multicolrule} {patch-success} {#1}}
61   {\msg_info:nnn {multicolrule} {patch-failure} {#1}}
62 }
```

(End definition for `__mcrule_patch_twocol_output:N`)

Now the actual patching begins.

```
63 \bool_if:NTF \g__mcrule_twocolumn_bool
64 {
65   \@ifpackageloaded{multicol}
66   {\msg_warning:nn {multicolrule} {multicol-loaded}}{}}
```

Default \TeX lacks `\columnseprulecolor`, so if we're in two-column mode, we provide it here.

```
67 \cs_gset:Npn \columnseprulecolor {\normalcolor}
```

In vanilla twocolumn mode, the column height and depth can be taken directly from `\@outputbox`.

```
68 \cs_gset:Npn \__mcrule_column_height: {\box_ht:N \@outputbox}
69 \cs_gset:Npn \__mcrule_column_depth: {\box_dp:N \@outputbox}
```

Now patch the relevant code in `\@outputdblcol`, replacing the hard-coded rule with a macro that we can overwrite.

```
70 \__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 `scrfile` to insert the patch if `bidi` is loaded later on.

```
71 \AfterPackage!{bidi}
72 {
73   \__mcrule_patch_twocol_output:N \RTL@outputdblcol
74   \__mcrule_patch_twocol_output:N \LTR@outputdblcol
75 }
76 }
```

Now patch for `multicol`.

```
77 {
78   \RequirePackage{multicol}
79   \__mcrule_patch_mcol_output:N \LR@column@boxes
80   \__mcrule_patch_mcol_output:N \RL@column@boxes
```

In LTR mode, we are invoked after a column has been typeset, which destroys their boxes in the process. So the only boxes we can be sure still exist are `\mult@rightbox`, which will be set to the same column height as all the others, and `\mult@nat@firstbox`, which contains the first column at its natural height.

```
81 \cs_gset:Npn \__mcrule_column_height:
82 {
83   \box_ht:N \mult@rightbox
84 }
```

Since the depth can differ from column to column, we use `\dimen\tw@`, which `multicol` uses to hold the maximum depth of all the columns already typeset, but as it won't have reached the final one yet, we check that too. This could lead to an inconsistent height in the event that there are 3 or more columns and a middle column has a significantly larger depth than either the previous columns or the last column, but for now it does not seem worth accounting for a condition that is likely to be very rare in actual user documents.

```
85 \cs_gset:Npn \__mcrule_column_depth:
86 {
```

```

87     \dim_max:nn {\dimen\tw@}{\box_dp:N \mult@rightbox}
88   }

```

To avoid widows, multicol allows some overflow, by default up to 12pt, and so it's possible that some text will overflow beyond the fixed bottom of the column. In this case, our rule won't descend far enough. To correct, we measure the column overflow as the difference between the natural height of the first box and the height of the last column, and add that amount if it's greater than 0pt.

```

89 \cs_gset:Npn \__mcrule_column_overflow:
90 {
91   \dim_max:nn {\box_ht:N \mult@nat@firstbox - \box_ht:N \mult@rightbox}{0pt}
92 }

```

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

```

93 \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. First we replace `bidi`'s copies of the column boxes routines with our patched version.

```

94 \AfterPackage!{bidi}
95 {
96   \cs_gset_eq:NN \LTR@column@boxes \LR@column@boxes
97   \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.

```

98   \cs_gset_eq:NN \LRmulticolcolumns \LTRmulticolcolumns
99   \cs_gset_eq:NN \RLmulticolcolumns \RTLmulticolcolumns
100 }
101 }

```

4.3 Creating the Rules

Now we declare utility functions for different rule types.

\mcruledivider

This is the function directly called by the patched output routines. Its main purpose 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.

```
102 \cs_new_protected:Npn \mcruledivider
103 {
```

If the pattern-after counter is set, wait that many iterations of the rule before we apply the patterns.

```
104   \int_compare:nNnTF {\g__mcrule_pattern_after_int} > {\c_zero_int}
105   {
106     \int_gdecr:N \g__mcrule_pattern_after_int
107   }
108   {
```

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.

```
109     \bool_lazy_and:nnT
110     {\int_compare_p:nNn {\seq_count:N \l__mcrule_pattern_list_seq} > {\c_zero_int}}
111     {\int_compare_p:nNn {\g__mcrule_pattern_for_int} = {\c_zero_int}}
112     {
113       \int_gincr:N \g__mcrule_pattern_count_int
114       \int_compare:nNnT {\g__mcrule_pattern_count_int} >
115       {\seq_count:N \l__mcrule_pattern_list_seq}
116       {
117         \int_gset:Nn \g__mcrule_pattern_count_int {\c_one_int}
118       }
119       \tl_set:Nx \l_tmpa_tl {\seq_item:Nn \l__mcrule_pattern_list_seq
120       {\g__mcrule_pattern_count_int} }
121       \tl_if_blank:VF \l_tmpa_tl
122       {
123         \__mcrule_set_pattern:V \l_tmpa_tl
124       }
125       \int_compare:nNnT {\g__mcrule_pattern_for_int} > {\c_zero_int}
126       {
127         \int_gdecr:N \g__mcrule_pattern_for_int
128       }
129     }
130   }
```

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

```
131   \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.

```
132   \bool_lazy_and:nnT
133   {\dim_compare_p:nNn {\columnseprule} > {\c_zero_dim}}
134   {\int_compare_p:nNn {\l__mcrule_repeat_int} > {\c_zero_int}}
135   {
136     \mcrule_divider:
137     \prg_replicate:nn {\l__mcrule_repeat_int - \c_one_int}
138     {
139       \hspace{\l__mcrule_repeat_distance_dim}
140       \mcrule_divider:
141     }
142   }
143 }
```

`_mcrule_column_total_height:` Get column height and depth with any explicit alterations.

```
\_mcrule\_column\_total\_depth:
144 \cs_new:Npn \__mcrule\_column\_total\_height:
145 {
146   \dim_eval:n {\__mcrule\_column\_height: + \__mcrule\_column\_depth: +
147     \__mcrule\_extend\_column\_top: + \__mcrule\_column\_overflow: + \__mcrule\_extend\_column\_bottom:}
148 }
149 \cs_new:Npn \__mcrule\_column\_total\_depth:
150 {
151   \dim_eval:n {\__mcrule\_column\_depth: + \__mcrule\_column\_overflow: +
152     \__mcrule\_extend\_column\_bottom:}
153 }
```

(End definition for `__mcrule_column_total_height:` and `__mcrule_column_total_depth:`.)

`__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.

```
154 \cs_new:Npn \__mcrule\_extend\_column\_top:
155 {
156   \l__mcrule\_extend\_top\_dim
157 }
```

(End definition for `__mcrule_extend_column_top:`.)

`_mcrule_extend_column_bottom:` The extend-fill option, which is only applicable with multicolumn, 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.

```
158 \cs_new:Npn \__mcrule\_extend\_column\_bottom:
159 {
160   \bool_lazy_and:nnTF
161     {\bool_if_p:n {\l__mcrule\_extend\_fill\_bool}}
162     {\bool_not_p:n {\g__mcrule\_twocolumn\_bool}}
163   {
164     \dim_compare:nNnTF
165       {\@colroom - \__mcrule\_column\_height: - \__mcrule\_extend\_reserve:} > {\c_zero_dim}
166       {\@colroom - \__mcrule\_column\_height: - \__mcrule\_extend\_reserve:}
167       {\c_zero_dim}
168   }
169   {\l__mcrule\_extend\_bot\_dim}
170 }
```

(End definition for `__mcrule_extend_column_bottom:`.)

`__mcrule_extend_reserve:` The reserved space is the amount of user-provided space we want, but we also have to account for the space added with `\multicolsep`.

```
171 \cs_new:Npn \__mcrule\_extend\_reserve:
172 {
173   \dim_compare:nNnTF {\l__mcrule\_extend\_reserve\_dim} > {\c_zero_dim}
174   {\dim_eval:n {\l__mcrule\_extend\_reserve\_dim + \multicolsep}}
175   {\c_zero_dim}
176 }
```

(End definition for `__mcrule_extend_reserve:`)

`\mcrule_divider:`

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.

```
177 \cs_new:Npn \mcrule_divider: {\vrule\@width\columnseprule}
```

`__mcrule_pattern:nnn`

`__mcrule_mcrule_pattern:nnn` $\{\langle pattern \rangle\}$ $\{\langle space\ above \rangle\}$ $\{\langle space\ below \rangle\}$

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 $\langle space\ above \rangle$ and $\langle space\ below \rangle$ arguments must be dimension expressions.

```
178 \cs_new_nopar:Npn \__mcrule_pattern:nnn #1#2#3
179 {
180   \box_move_down:nn {\__mcrule_column_total_depth:}
181   {
182     \vbox_to_ht:nn {\__mcrule_column_total_height:}
183     {
184       \tex_vfill:D
185       \tex_kern:D \dim_eval:n {#2} \exp_stop_f:
186       \hbox:n{#1}
187       \tex_kern:D \dim_eval:n {#3} \exp_stop_f:
188       \tex_vfill:D
189     }
190   }
191 }
```

(End definition for `__mcrule_pattern:nnn`.)

`__mcrule_tile_pattern:nnn`

`\mcrule_tile_pattern:nnn` $\{\langle pattern \rangle\}$ $\{\langle space\ above \rangle\}$ $\{\langle space\ below \rangle\}$

Typesets multiple copies of pattern, tiled so as to occupy a vertical box that is the height of the current column. The pattern must be something that can go in a horizontal box. The $\langle space\ above \rangle$ and $\langle space\ below \rangle$ arguments must be dimension expressions.

```
192 \cs_new_nopar:Npn \__mcrule_tile_pattern:nnn #1#2#3
193 {
194   \box_move_down:nn {\__mcrule_column_total_depth:}
195   {
196     \vbox_to_ht:nn {\__mcrule_column_total_height:}
197     {
198       \tex_cleaders:D \vbox:n
199       {
200         \tex_kern:D \dim_eval:n {#2} \exp_stop_f:
201         \hbox:n{#1}
202         \tex_kern:D \dim_eval:n {#3} \exp_stop_f:
203       }
204       \tex_vfill:D
205     }
206   }
207 }
```

(End definition for `__mcrule_tile_pattern:nnn`.)

`__mcrule_line_pattern:nnnn` `__mcrule_mcrule_line_pattern:nnnn` $\{\langle tikz-name \rangle\}$ $\{\langle height \rangle\}$ $\{\langle space\ above \rangle\}$
 $\{\langle space\ below \rangle\}$

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 $\langle height \rangle$ separated by spaces above and/or below the line.

```

208 \cs_new:Npn \__mcrule_line_pattern:nnnn #1#2#3#4
209 {
210   \bool_if:NTF \g__mcrule_use_tikz_bool
211   {
212     \__mcrule_pattern_line:n {#1}
213   }
214   {
215     \__mcrule_tile_pattern:nnn {\rule{\columnseprule}{#2}}{#3}{#4}
216   }
217 }

```

(End definition for `__mcrule_line_pattern:nnnn`.)

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

```

218 \cs_new:Npn \__mcrule_solid_line:
219 {
220   \rule[-\__mcrule_column_total_depth:]{\columnseprule}{\__mcrule_column_total_height:}
221 }

```

(End definition for `__mcrule_solid_line:.`)

`__mcrule_strut:` Uses a zero-width rule, regardless of the actual value of `\columnseprule`.

```

222 \cs_new:Npn \__mcrule_strut:
223 {
224   \rule[-\__mcrule_column_total_depth:]{0pt}{\__mcrule_column_total_height:}
225 }

```

(End definition for `__mcrule_strut:.`)

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.

```

226 \bool_if:NTF \g__mcrule_use_tikz_bool
227 {
228   \ExplSyntaxOff
229   \RequirePackage{tikz}
230   \ExplSyntaxOn

```

`__mcrule_tikz_picture:n` `__mcrule_tikz_picture:n` $\{\langle draw\ function \rangle\}$

Set up the `tikzpicture` environment and declare two nodes, named (TOP) and (BOT). This way we can pass a `\draw` routine directly, without worrying about the line's coordinates.

```

231 \cs_set:Npn \__mcrule_tikz_picture:n #1
232 {
233   \begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt,
234     baseline={([yshift=\__mcrule_column_total_depth:]current~bounding~box.south)}]
235   \node (TOP) at (0,\__mcrule_column_total_height:) {};

```

```

236 \node (BOT) at (0,0) {};
237 #1
238 \end{tikzpicture}
239 }

```

(End definition for `__mcrule_tikz_picture:n`)

`__mcrule_pattern_line:n`

```
\__mcrule_mcrule_pattern_line:n {\tikz pattern}
```

For the tikz versions of the predefined lines, we just draw a line the length of the column box. *\tikz pattern* should contain the name of a line style that tikz recognizes.

```

240 \cs_set:Npn \__mcrule_pattern_line:n #1
241 {
242   \begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt,
243     baseline={([yshift=\__mcrule_column_total_depth:]current~bounding~box.south)}]
244   \draw[line~width=\columnseprule,#1] (0,\__mcrule_column_total_height) -- (0,0);
245   \end{tikzpicture}
246 }

```

(End definition for `__mcrule_pattern_line:n`)

`__mcrule_circle:` Draw a hollow circle with a diameter equal to `\columnseprule`. This will be used as a tile pattern.

```

247 \cs_set:Npn \__mcrule_circle:
248 {
249   \begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt]
250   \draw (0,0) circle[radius=.5\columnseprule];
251   \end{tikzpicture}
252 }

```

(End definition for `__mcrule_circle:`)

`__mcrule_solid_circle:` Draw a filled circle with a diameter equal to `\columnseprule`. This will be used as a tile pattern.

```

253 \cs_set:Npn \__mcrule_solid_circle:
254 {
255   \begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt]
256   \fill (0,0) circle[radius=.5\columnseprule];
257   \end{tikzpicture}
258 }
259 }

```

(End definition for `__mcrule_solid_circle:`)

In case tikz functions are not active, we provide stubs that issue error messages.

```

260 {
261   \cs_set:Npn \__mcrule_tikz_picture:n #1
262     {\msg_error:nnn {multicolrule} {tikz-required} {#1}}
263   \cs_new:Npn \__mcrule_pattern_line:n #1
264     {\msg_error:nnn {multicolrule} {tikz-required} {#1}}
265   \cs_new:Npn \__mcrule_circle:
266     {\msg_error:nnn {multicolrule} {tikz-required} {circles}}
267   \cs_new:Npn \__mcrule_solid_circle:
268     {\msg_error:nnn {multicolrule} {tikz-required} {solid-circles}}
269 }

```

4.4 Color

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

```

270 \cs_new_protected:Npn \__mcrule_set_rule_color:
271 {
272   \tl_if_empty:NT \l__mcrule_color_name_tl
273   {
274     \tl_set:Nn \l__mcrule_color_name_tl {black}
275   }
276   \tl_if_empty:NTF \l__mcrule_color_model_tl
277   {
278     \cs_set:Npn \columnseprulecolor {\color{\l__mcrule_color_name_tl}}
279   }
280   {
281     \cs_set:Npn \columnseprulecolor
282     {\color[\l__mcrule_color_model_tl]{\l__mcrule_color_name_tl}}
283   }
284 }

```

(End definition for `__mcrule_set_rule_color:`)

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.

```

285 \cs_new_protected:Npn \__mcrule_set_pattern_list:n #1
286 {
287   \seq_set_split:Nnn \l__mcrule_pattern_list_seq {,} {#1}
288   \int_gzero:N \g__mcrule_pattern_count_int
289   \int_gzero:N \g__mcrule_pattern_after_int
290   \int_gset:Nn \g__mcrule_pattern_for_int {-1}
291 }

```

(End definition for `__mcrule_set_pattern_list:n`)

`__mcrule_set_pattern:n` Set the keys an individual pattern. To avoid potential recursion and loops, we filter out the key ‘pattern’ when it appears in a pattern definition.

```

292 \cs_new_protected:Npn \__mcrule_set_pattern:n #1
293 {
294   \prop_get:NnNTF \g__mcrule_patterns_prop {#1} \l_tmpa_tl
295   {
296     \keys_set_filter:nnV {mcrule} {patterns} \l_tmpa_tl
297   }
298   {
299     \msg_error:nnn {multicolrule} {pattern-undefined} {#1}
300   }
301   \tl_set:Nn \l_tmpa_tl {\prop_item:Nn \g__mcrule_patterns_prop {#1}}
302 }
303 \cs_generate_variant:Nn \__mcrule_set_pattern:n {V}

```

(End definition for `__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.

```

304 \keys_define:nn {mcrule}
305 {
306   extend-top           .dim_set:N = \l__mcrule_extend_top_dim,
307   extend-bot           .dim_set:N = \l__mcrule_extend_bot_dim,
308   extend-fill          .bool_set:N = \l__mcrule_extend_fill_bool,
309   extend-fill          .default:n = true,
310   extend-reserve       .dim_set:N = \l__mcrule_extend_reserve_dim,
311   expand                .code:n = {
312     \dim_set:Nn \l__mcrule_extend_bot_dim {#1}
313     \dim_set:Nn \l__mcrule_extend_top_dim {#1}
314   },
315   shift                .code:n = {
316     \dim_set:Nn \l__mcrule_extend_bot_dim {#1}
317     \dim_set:Nn \l__mcrule_extend_top_dim {\fp_to_dim:n {-1 * \l__mcrule_extend_bot_dim}}
318   },
319   line-style           .choice:,
320   line-style / default .code:n = \cs_set:Npn \mcrule_divider:
321     {\vrule\@width\columnseprule},
322   line-style / solid   .code:n = \cs_set:Npn \mcrule_divider:
323     {\__mcrule_solid_line:},
324   line-style / strut   .code:n = \cs_set:Npn \mcrule_divider:
325     {\__mcrule_strut:},
326   line-style / dots    .code:n = \cs_set:Npn \mcrule_divider:
327     {\__mcrule_tile_pattern:nnn {.}{1pt}{1pt}},
328   line-style / dense-dots .code:n = \cs_set:Npn \mcrule_divider:
329     {\__mcrule_tile_pattern:nnn {.}{1pt}{0pt}},
330   line-style / loose-dots .code:n = \cs_set:Npn \mcrule_divider:
331     {\__mcrule_tile_pattern:nnn {.}{2pt}{2pt}},
332   line-style / circles .code:n = \cs_set:Npn \mcrule_divider:
333     {\__mcrule_tile_pattern:nnn {\__mcrule_circle:}{1pt}{1pt}},
334   line-style / dense-circles .code:n = \cs_set:Npn \mcrule_divider:
335     {\__mcrule_tile_pattern:nnn {\__mcrule_circle:}{1pt}{0pt}},
336   line-style / loose-circles .code:n = \cs_set:Npn \mcrule_divider:
337     {\__mcrule_tile_pattern:nnn {\__mcrule_circle:}{2pt}{2pt}},
338   line-style / solid-circles .code:n = \cs_set:Npn \mcrule_divider:
339     {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{1pt}{1pt}},
340   line-style / dense-solid-circles .code:n = \cs_set:Npn \mcrule_divider:
341     {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{1pt}{0pt}},
342   line-style / loose-solid-circles .code:n = \cs_set:Npn \mcrule_divider:
343     {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{2pt}{2pt}},
344   line-style / dotted    .code:n = \cs_set:Npn \mcrule_divider:
345     {\__mcrule_line_pattern:nnnn {dotted}{\columnseprule}{1pt}{1pt}},
346   line-style / densely-dotted .code:n = \cs_set:Npn \mcrule_divider:
347     {\__mcrule_line_pattern:nnnn {densely-dotted}{\columnseprule}{1pt}{0pt}},
348   line-style / loosely-dotted .code:n = \cs_set:Npn \mcrule_divider:
349     {\__mcrule_line_pattern:nnnn {loosely-dotted}{\columnseprule}{2pt}{2pt}},
350   line-style / dashed    .code:n = \cs_set:Npn \mcrule_divider:
351     {\__mcrule_line_pattern:nnnn {dashed}{3pt}{1.5pt}{1.5pt}},
352   line-style / densely-dashed .code:n = \cs_set:Npn \mcrule_divider:
353     {\__mcrule_line_pattern:nnnn {densely~dashed}{3pt}{1pt}{1pt}},

```



```

354 line-style / loosely-dashed .code:n = \cs_set:Npn \mcrule_divider:
355   {\__mcrule_line_pattern:nnnn {loosely~dashed}{3pt}{3pt}{3pt}},
356 line-style / dash-dot .code:n = \cs_set:Npn \mcrule_divider:
357   {\__mcrule_pattern_line:n{dash~dot}},
358 line-style / densely-dash-dot .code:n = \cs_set:Npn \mcrule_divider:
359   {\__mcrule_pattern_line:n{densely~dash~dot}},
360 line-style / loosely-dash-dot .code:n = \cs_set:Npn \mcrule_divider:
361   {\__mcrule_pattern_line:n{loosely~dash~dot}},
362 line-style / dash-dot-dot .code:n = \cs_set:Npn \mcrule_divider:
363   {\__mcrule_pattern_line:n{dash~dot~dot}},
364 line-style / densely-dash-dot-dot .code:n = \cs_set:Npn \mcrule_divider:
365   {\__mcrule_pattern_line:n{densely~dash~dot~dot}},
366 line-style / loosely-dash-dot-dot .code:n = \cs_set:Npn \mcrule_divider:
367   {\__mcrule_pattern_line:n{loosely~dash~dot~dot}},
368 color .code:n = {
369   \tl_set:Nn \l__mcrule_color_name_tl {#1}
370   \__mcrule_set_rule_color:
371 },
372 color-model .code:n = {
373   \tl_set:Nn \l__mcrule_color_model_tl {#1}
374   \__mcrule_set_rule_color:
375 },
376 custom-line .code:n = \cs_set:Npn \mcrule_divider:
377   {\__mcrule_tikz_picture:n {#1}},
378 custom-pattern .code:n = \cs_set:Npn \mcrule_divider:
379   {\__mcrule_pattern:nnn #1},
380 custom-tile .code:n = \cs_set:Npn \mcrule_divider:
381   {\__mcrule_tile_pattern:nnn #1},
382 width .choice:,
383 width / ultra-thin .code:n = \dim_set:Nn \columnseprule {0.1pt},
384 width / very-thin .code:n = \dim_set:Nn \columnseprule {0.2pt},
385 width / thin .code:n = \dim_set:Nn \columnseprule {0.4pt},
386 width / semithick .code:n = \dim_set:Nn \columnseprule {0.6pt},
387 width / thick .code:n = \dim_set:Nn \columnseprule {0.8pt},
388 width / very-thick .code:n = \dim_set:Nn \columnseprule {1.2pt},
389 width / ultra-thick .code:n = \dim_set:Nn \columnseprule {1.6pt},
390 width / unknown .code:n = \dim_set:Nn \columnseprule {#1},
391 repeat .int_set:N = \l__mcrule_repeat_int,
392 repeat-distance .dim_set:N = \l__mcrule_repeat_distance_dim,
393 single .meta:n = {
394   repeat = 1,
395   repeat-distance = #1
396 },
397 single .default:n = \columnseprule,
398 double .meta:n = {
399   repeat = 2,
400   repeat-distance = #1
401 },
402 double .default:n = \columnseprule,
403 triple .meta:n = {
404   repeat = 3,
405   repeat-distance = #1
406 },
407 triple .default:n = \columnseprule,

```

```

408     patterns      .code:n      = \__mcrule_set_pattern_list:n {#1},
409     patterns      .groups:n    = {patterns},
410     pattern-after  .int_gset:N = \g__mcrule_pattern_after_int,
411     pattern-for    .int_gset:N = \g__mcrule_pattern_for_int,
412 }

```

4.7 User Interface

\SetMCRule Set all keys for **multicolrule**

$$\backslash \text{SetMCRule } \{\langle \textit{key-value list} \rangle\}$$

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

```

413 \NewDocumentCommand{\SetMCRule}{m}
414 {
415   \keys_set:nn {mcrule} {#1}
416 }

```

(End definition for `\SetMCRule`. This function is documented on page 4.)

\DeclareMCRulePattern

$$\backslash\text{DeclareMCRule } \{\langle name \rangle\} \langle key\text{-}value \text{ list} \rangle$$

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

```

417 \NewDocumentCommand{\DeclareMCRulePattern}{m m}
418 {
419   \prop_gput:Nnn \g__mcrule_patterns_prop {#1} {#2}
420 }

```

(End definition for `\DeclareMCRulePattern`. This function is documented on page 10.)

Change History

v1.0	General: Initial public release	1	Move \columnseprulecolor inside \mcruledivider	13	
v1.1	General: Added extend-top, extend-bot, extend-fill, and extend-reserve keys	22	v1.2a	General: Added expand and shift shortcuts	22
	Allow extended rules	1		Check depth of final column	14
	Support bidi	1		Improve documentation	1
v1.2	General: Add pattern support	16		_mcrule_column_overflow::New internal function	13
	Added patterns, pattern-after, and pattern-for keys	22		_mcrule_column_total_depth:: Account for overflow text	17
	Allow per-column rule changes	1		_mcrule_pattern::nnn: Allow evaluated dimensions for kerns	18
	Define rule patterns	1		_mcrule_tile_pattern::nnn: Allow evaluated dimensions for kerns	18
	_mcrule_patch_mcol_output:N: Move \columnseprulecolor inside \mcruledivider	13	v1.3	_mcrule_strut::New function	19
	_mcrule_patch_twocol_output:N:				

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	
<code>\AfterPackage</code>	<i>14</i>
C	
<code>\cleaders</code>	<i>3</i>
<code>\color</code>	<i>278, 282</i>
<code>\columnsep</code>	<i>2</i>
<code>\columnseprule</code>	<i>2, 5–9,</i> <i>16, 19, 51, 58, 133, 177, 215, 220, 244,</i> <i>250, 256, 321, 345, 347, 349, 383, 384,</i> <i>385, 386, 387, 388, 389, 390, 397, 402, 407</i>
<code>\columnseprulecolor</code>	<i>6–</i> <i>8, 13, 14, 21, 24, 51, 67, 131, 278, 281</i>
D	
<code>\DeclareMCRule</code>	<i>24</i>
<code>\DeclareMCRulePattern</code>	<i>10</i>
<code>\DeclareMCRulePattern</code>	<i>10, 10, <u>417</u></i>
<code>\definecolor</code>	<i>8</i>
<code>\draw</code>	<i>7, 19, 244, 250</i>
H	
<code>\HandRight</code>	<i>7</i>
L	
<code>\LRmulticolcolumns</code>	<i>15, 15, 15, 93, 98</i>
<code>\LTRmulticolcolumns</code>	<i>98</i>
M	
<code>\maxbalancingoverflow</code>	<i>13</i>
mcrule commands:	
<code>\mcrule_divider:</code>	<i>16,</i> <i>16, 18, 22, 136, 140, 177, 320, 322,</i> <i>324, 326, 328, 330, 332, 334, 336, 338,</i> <i>340, 342, 344, 346, 348, 350, 352, 354,</i> <i>356, 358, 360, 362, 364, 366, 376, 378, 380</i>
<code>\mcrule_tile_pattern:nnn</code>	<i>18</i>
mcrule internal commands:	
<code>__mcrule_circle:</code>	<i>247, 265, 333, 335, 337</i>
<code>\l__mcrule_color_model_tl</code>	<i>28, 276, 282, 373</i>
<code>\l__mcrule_color_name_tl</code>	<i>28, 272, 274, 278, 282, 369</i>
<code>__mcrule_column_depth:</code>	<i>47, 69, 85, 146, 151</i>
<code>__mcrule_column_height:</code>	<i>46, 68, 81, 146, 165, 166</i>
<code>__mcrule_column_overflow:</code>	<i>48, 89, 147, 151</i>
<code>__mcrule_column_total_depth:</code>	<i>144, 180, 194, 220, 224, 234, 243</i>
<code>__mcrule_column_total_height:</code>	<i>144, 182, 196, 220, 224, 235, 244</i>
<code>\l__mcrule_extend_bot_dim</code>	<i>24, 169, 307, 312, 316, 317</i>
<code>__mcrule_extend_column_-</code> bottom:	<i>147, 152, <u>158</u></i>
<code>__mcrule_extend_column_top:</code>	<i>147, <u>154</u></i>
<code>\l__mcrule_extend_fill_bool</code>	<i>24, 161, 308</i>
<code>__mcrule_extend_reserve:</code>	<i>165, 166, <u>171</u></i>
<code>\l__mcrule_extend_reserve_dim</code>	<i>24, 173, 174, 310</i>
<code>\l__mcrule_extend_top_dim</code>	<i>24, 156, 306, 313, 317</i>
<code>__mcrule_line_pattern:nnnn</code>	<i>208, 345, 347, 349, 351, 353, 355</i>
<code>__mcrule_mcrule_line_pattern:nnnn</code>	<i>19</i>
<code>__mcrule_mcrule_pattern:nnn</code>	<i>18</i>
<code>__mcrule_mcrule_pattern_-</code> line:n	<i>20</i>
<code>__mcrule_patch_mcol_output:N</code>	<i>49, 79, 80</i>
<code>__mcrule_patch_twocol_output:N</code>	<i>56, 70, 73, 74</i>
<code>__mcrule_pattern:nnn</code>	<i>178, 379</i>
<code>\g__mcrule_pattern_after_int</code>	<i>30, 104, 106, 289, 410</i>
<code>\g__mcrule_pattern_count_int</code>	<i>30, 113, 114, 117, 120, 288</i>
<code>\g__mcrule_pattern_for_int</code>	<i>30, 111, 125, 127, 290, 411</i>
<code>__mcrule_pattern_line:n</code>	<i>212, <u>240</u>, 263, 357, 359, 361, 363, 365, 367</i>
<code>\l__mcrule_pattern_list_seq</code>	<i>30, 110, 115, 119, 287</i>
<code>\g__mcrule_patterns_prop</code>	<i>30, 294, 301, 419</i>
<code>\l__mcrule_repeat_distance_dim</code>	<i>21, 139, 392</i>
<code>\l__mcrule_repeat_int</code>	<i>16, <u>21</u>, 134, 137, 391</i>
<code>__mcrule_set_pattern:n</code>	<i>123, <u>292</u></i>

<code>__mcrule_set_pattern_list:n</code>	285, 408
<code>__mcrule_set_rule_color:</code>	270, 370, 374
<code>__mcrule_solid_circle:</code>	253, 267, 339, 341, 343
<code>__mcrule_solid_line:</code>	218, 323
<code>__mcrule_strut:</code>	24, 222, 325
<code>__mcrule_tikz_picture:n</code> 231, 261, 377	
<code>__mcrule_tile_pattern:nnn</code>	192, 215, 327, 329, 331, 333, 335, 337, 339, 341, 343, 381
<code>\g__mcrule_twocolumn_bool</code>	19, 41, 63, 162
<code>\g__mcrule_use_tikz_bool</code>	19, 37, 42, 210, 226
<code>\mcruledivider</code>	13, 16, 24, 52, 59, 102
<code>\MCSetRule</code>	18
<code>\multicolsep</code>	9, 17, 174
O	
options:	
<code>tikz</code>	3
<code>twocolumn</code>	4
P	
<code>\path</code>	7
R	
<code>\RLmulticolcolumns</code>	15, 15, 99
<code>\RTLmulticolcolumns</code>	99
S	
<code>\SetMCRule</code>	4
<code>\SetMCRule</code>	2, 4, 4, 6, 10, 413
<code>\SparkleBold</code>	7
T	
TeX and L ^A T _E X 2 _ε commands:	
<code>\@colroom</code>	165, 166
<code>\@outputbox</code>	14, 68, 69
<code>\@outputdblcol</code>	14, 70
<code>\LR@column@boxes</code>	79, 96
<code>\LTR@column@boxes</code>	96
<code>\LTR@outputdblcol</code>	74
<code>\mc@align@columns</code>	15
<code>\mult@nat@firstbox</code>	14, 91
<code>\mult@rightbox</code>	14, 83, 87, 91
<code>\RL@column@boxes</code>	80, 97
<code>\RTL@column@boxes</code>	97
<code>\RTL@outputdblcol</code>	73
<code>\vrule</code>	19
tex commands:	
<code>\tex_cleaders:D</code>	198
<code>\tex_kern:D</code>	185, 187, 200, 202
<code>\tex_vfill:D</code>	184, 188, 204
<code>tikz (option)</code>	3
<code>twocolumn (option)</code>	4
<code>\twocolumn</code>	1, 4