multicolrule — Decorative rules between columns*

Karl Hagen†

Released 2018/12/20

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

*This file describes version v1.1, last revised 2018/12/20.

†latex@polysyllabic.com

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.

Note—in case it isn't obvious yet—that this guide illustrates the basic line styles that multicolrule makes available throught the document. The default line-width is 0.4pt (thin), and the default color is `Maroon`. You can find examples of rules created with all available options in the file `mcrule-example.pdf`.

1.1 Bugs and Known Limitations

line-style=dots

There are likely bugs that remain to be uncovered, as well as missing features and inefficient methods that should be improved upon. The development code is maintained on github (`https://github.com/polysyllabic/multicolrule`), 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 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.

I have also noticed occasional instances, most noticeably when a `multicols` environment starts near the bottom of a page and the columns continue to the next one, where the rules are either somewhat shorter than they should be or shifted upward from where they belong. In the limited testing I have done, this appears to be a consequence of how multicol works, as the default rules show the same behavior. I may try to nail down this issue in future version, but as it's an edge case that disappears when you add page breaks or rewrite the text to alter how the columns are filled, it hasn't seemed worth taking the time to fix at this point.

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

[1] See, for example, the remarks in the documentation for the booktabs package

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.

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

1.2 License

line-style=dotted, width=ultra-thick

2 Package Options

2.1 Default Operation

line-style=dash-dot

If you load multicolrule with its default settings, it will enable 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 '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.

2.3 Option 'tikz'

You have access to a wider set of 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.

3 The User Interface

line-style=circles,
width=2pt

The multicolrule package has just a single user command:

\SetMCRule {⟨*key-value*⟩}

which 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
color	Set the color of the rule (see sec. 3.2)
color-model	Set the color model of the rule (see sec. 3.2)
custom-line	Set a custom tikz line for the rule (*tikz only*; see 3.1.2)
custom-pattern	Set a custom individual pattern for the rule (see 3.1.2)
custom-tile	Set a custom tiling pattern for the rule (see 3.1.2)
double	Draw two copies of the rule (see sec. 3.4)
line-style	Select the type of rule printed (see sec. 3.1)
single	Draw a single copy of the rule (*default*; see sec. 3.4)
repeat	Set the number of times to draw the rule (see sec. 3.4)
repeat-distance	Set the horizontal space between adjacent copies of repeated rules (see sec. 3.4)
triple	Draw three copies of the rule (see sec. 3.4)
width	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 choose a style for the rule with the line-style key. The default style is solid. In addition to the predefined styles, there are also several ways to get multicolrule to draw custom shapes in place of the column rule. The width of most line styles depends on the setting of \columnseprule, which is the default LaTeX length that controls the width of the column rule (see section 3.3).

Table 2 summarizes the available line styles. Most of the basic shapes used to form the patterns come in three versions, which differ only in how closely the pattern is spaced: normal, dense, and loose. These settings parallel those found in tikz, and those line styles whose names are identical to the line patterns in tikz (apart from the substitution of '-' for spaces) produce the same effect.

Table 2: Styles available for the line-style key

Style	Description
`circles`	A series of hollow circles (*tikz only*)
`dash-dot`	A dash followed by a square dot (*tikz only*)
`dash-dot-dot`	A dash followed by two square dots (*tikz only*)
`dashed`	A series of dashed lines
`dense-circles`	The same as `circles` but more closely spaced (*tikz only*)
`dense-dots`	The same as `dots` but more closely spaced
`dense-solid-circles`	The same as `solid-circles` but more closely spaced (*tikz only*)
`densely-dash-dot`	The same as `dash-dot` but more closely spaced (*tikz only*)
`densely-dash-dot-dot`	The same as `dash-dot-dot` but more closely spaced (*tikz only*)
`densely-dashed`	The same as `dashed` but more closely spaced
`densely-dotted`	The same as `dotted` but more closely spaced
`dots`	A series of dots drawn with the period (full-stop) of the current font
`dotted`	A series of square dots
`loose-dots`	The same as `dots` but spaced further apart
`loose-circles`	The same as `circles` but spaced further apart (*tikz only*)
`loose-solid-circles`	The same as `solid-circles` but spaced further apart (*tikz only*)
`loosely-dash-dot`	The same as `dash-dot` but spaced further apart (*tikz only*)
`loosely-dash-dot-dot`	The same as `dash-dot-dot` but spaced further apart (*tikz only*)
`loosely-dashed`	The same as `dashed` but spaced further apart
`loosely-dotted`	The same as `dotted` but spaced further apart
`solid`	A solid line (*default*)
`solid-circles`	A series of filled circles (*tikz only*)

3.1.1 Notes on the Styles

`line-style=solid`

The `solid` line style is the default. In fact, if you make no calls to `\SetMCRule` after loading multicolrule, the column divider will behave exactly as it does with the ordinary multicol package. You can alter its width and color either with the `width` and `color` keys described in sections 3.3 and 3.2, respectively, or you can set the width directly by changing the value of `\columnseprule` and renewing the `\columnseprulecolor` macro. Like all line styles, the solid line can be repeated as many times as you like (see section 3.4).

The `dots` style and its variants are rendered with a period (.) in the currently active font. This means that changing `\columnseprule` will not change the size of these dots, although, as with all rules, it will not appear at all if `\columnseprule` is set to 0pt.

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.

3.1.2 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, you do *not* specify a separate `line-style`.

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

```
custom-pattern=
{\HandRight}
{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 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 `\HandRight` symbol from bbding.

`custom-line=` {⟨*draw command*⟩}

```
custom-line={
\draw[line width=
\columnseprule] (TOP)
to [ornament=88]
(BOT);},
width=1pt
```

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`, 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` 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 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

```
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 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 `\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 `\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
`ultra-thin`	0.1pt
`very-thin`	0.2pt
`thin`	0.4pt
`semithick`	0.6pt
`thick`	0.8pt
`very-thick`	1.2pt
`ultra-thick`	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`.

The keys `single`, `double`, and `triple` are shorthand methods to set the number of repeats and the `repeat-distance` at the same time. If 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.

4 Implementation

⟨*package⟩
⟨@@=mcrule⟩

4.1 Preliminaries

```
\ProvidesExplPackage {multicolrule} {2018/12/20} {1.1}
  {Decorative~vertical~rules~between~columns}
```

We always need these packages.

```
\RequirePackage{l3keys2e}
\RequirePackage{xpatch}
\RequirePackage{xcolor}
\RequirePackage{scrlfile}
```

Define the messages we use.

```
\msg_new:nnn {multicolrule} {patch-success} {Patched~#1.}
\msg_new:nnn {multicolrule} {patch-failure} {Error~patching~#1.}
\msg_new:nnn {multicolrule} {tikz-required}
{The~'#1'~setting~requires~tikz~to~work.~Either~load~tikz~before~you~load~
  multicolrule~or~use~multicolrule's~'tikz'~package~option.}
\msg_new:nnn {multicolrule} {multicol-loaded} {You~are~using~the~'twocolumn'~
  option~with~multicol~already~loaded.~You~will~likely~run~into~problems.}
```

\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`.)

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.

```
\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}
```

4.2 Patching Output Routines

`\__mcrule_column_height:`
`\__mcrule_column_depth:`

Get the height and depth of the box appropriate to the supported mode.

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

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.

`\__mcrule_patch_mcol_output:N`

```
\cs_new_protected:Npn \__mcrule_patch_mcol_output:N #1
{
  \xpatchcmd{#1} {\vrule\@width\columnseprule} {\mcruledivider}
  {\msg_info:nnn {multicolrule} {patch-success} {#1}}
  {\msg_info:nnn {multicolrule} {patch-failure} {#1}}
}
```

`\__mcrule_patch_twocol_output:N`

```
\cs_new_protected:Npn \__mcrule_patch_twocol_output:N #1
{
  \xpatchcmd{#1} {\normalcolor\vrule\@width\columnseprule}
  {\columnseprulecolor\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}}{}
```

Provide the column-color macro from multicol.

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

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
  }
}
```

4.3 Creating the Rules

Utility functions for different rule types

`\mcruledivider` This is the function directly called by the patched output routines. It's given a LaTeX2 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`. 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.

```
\cs_new:Npn \mcruledivider
{
  \bool_lazy_and:nnT
  {\dim_compare_p:nNn {\columnseprule} > {0pt}}
  {\int_compare_p:nNn {\l__mcrule_repeat_int} > {0}}
  {
    \__mcrule_divider:
    \prg_replicate:nn {\l__mcrule_repeat_int - 1}
    {
      \hspace{\l__mcrule_repeat_distance_dim}
      \__mcrule_divider:
    }
  }
}
```

(*End definition for* `\mcruledivider`. *This function is documented on page* ??.)

```
\__mcrule_column_total_height:
\__mcrule_column_total_depth:
\__mcrule_extend_column_top:
\__mcrule_extend_column_bottom:
\__mcrule_extend_reserve:
```

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:}
}
```

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:
{
  \l__mcrule_extend_top_dim
}
```

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:} > {0pt}
    {\@colroom - \__mcrule_column_height: - \__mcrule_extend_reserve:}
    {0pt}
  }
  {\l__mcrule_extend_bot_dim}
}
```

The reserve 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} > {0pt}
  {\dim_eval:n {\l__mcrule_extend_reserve_dim + \multicolsep}}
  {0pt}
}
```

__mcrule_divider:

This is the internal 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.

```
\cs_new:Npn \__mcrule_divider: {\__mcrule_debug_log:n {default}
  \vrule\@width\columnseprule}
```

__mcrule_pattern:nnn

__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:}
    {
      \vfill
      \kern #2 \hbox:n{#1} \kern #3
      \vfill
    }
  }
}
```

```
\cs_new:Npn \__mcrule_debug_log:n #1
{
  \tl_log:n {#1}
  \dim_log:N \pagegoal
  \dim_log:N \maxdimen
  \dim_log:N \pagetotal
  \dim_log:n {\pagegoal - \pagetotal}
  \dim_log:N \page@free
  \dim_log:N \dimen@
  \dim_compare:nNnTF {\dimen@} = {\__mcrule_column_height:}
    {\tl_log:n {dimen@ = box~height}}
    {\dim_log:n {\__mcrule_column_height:}}
  \dim_log:N \@colroom
  \dim_log:n {\dimen\tw@}
  \dim_compare:nNnTF {\dimen\tw@} = {\__mcrule_column_depth:}
    {\tl_log:n {dimen@ = box~depth}}
    {\dim_log:n {\__mcrule_column_depth:}}
  \dim_log:n {\__mcrule_column_total_height:}
  \dim_log:n {\__mcrule_column_total_depth:}
}
```

`\__mcrule_tile_pattern:nnn` `\__mcrule_tile_pattern:nnn {⟨pattern⟩} {⟨space above⟩} {⟨space below⟩}`

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 spaces above and below must be fixed dimensions.

```
\cs_new_nopar:Npn \__mcrule_tile_pattern:nnn #1#2#3
{
  \__mcrule_debug_log:n {tile~#1}
  \box_move_down:nn {\__mcrule_column_total_depth:}
  {
    \vbox_to_ht:nn {\__mcrule_column_total_height:}
    {
      \cleaders \vbox:n
      {
        \kern #2 \hbox:n{#1} \kern #3
      }
    \vfill
    }
  }
}
```

`\__mcrule_line_pattern:nnnn` `\__mcrule_line_pattern:nnnn {⟨tikz-name⟩} {⟨height⟩} {⟨space above⟩} {⟨space below⟩}`

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 ⟨*height*⟩ separated by spaces above and/or below the line.

```
\cs_new:Npn \__mcrule_line_pattern:nnnn #1#2#3#4
{
  \__mcrule_debug_log:n {line~#1}
  \bool_if:NTF \g__mcrule_use_tikz_bool
  {
    \__mcrule_pattern_line:n {#1}
  }
  {
    \__mcrule_tile_pattern:nnn {\rule{\columnseprule}{#2}}{#3}{#4}
  }
}
```

`\__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.

```
\cs_new:Npn \__mcrule_solid_line:
{
  \__mcrule_debug_log:n {solid~line}
 \rule[-\__mcrule_column_total_depth:]{\columnseprule}{\__mcrule_column_total_height:}
}
```

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.

```
\bool_if:NTF \g__mcrule_use_tikz_bool
```

```
{
  \ExplSyntaxOff
  \RequirePackage{tikz}
  \ExplSyntaxOn
```

`\__mcrule_tikz_picture:n` `\__mcrule_tikz_picture:n {⟨draw function⟩}`

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.

```
\cs_set:Npn \__mcrule_tikz_picture:n #1
{
  \__mcrule_debug_log:n {tikz~picture}
  \begin{tikzpicture}[x=1pt,y=1pt,inner~sep=0pt,outer~sep=0pt,
   baseline={([yshift=\__mcrule_column_total_depth:]current~bounding~box.south)}]
  \node (TOP) at (0,\__mcrule_column_total_height:) {};
  \node (BOT) at (0,0) {};
  #1
    \end{tikzpicture}
}
```

`\__mcrule_pattern_line:n` `\__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.

```
\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}
}
```

`\__mcrule_circle:`

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

```
  \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}
  }
```

`\__mcrule_solid_circle:`

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

```
  \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}
  }
}
```

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

```
{
  \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}}
}
```

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:
{
  \tl_if_empty:NT \l__mcrule_color_name_tl
  {
    \tl_set:Nn \l__mcrule_color_name_tl {black}
  }
  \tl_if_empty:NTF \l__mcrule_color_model_tl
  {
    \cs_set:Npn \columnseprulecolor {\color{\l__mcrule_color_name_tl}}
  }
  {
    \cs_set:Npn \columnseprulecolor
    {\color[\l__mcrule_color_model_tl]{\l__mcrule_color_name_tl}}
  }
}
```

4.5 Key-Values

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

```
\keys_define:nn {mcrule}
{
  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:
    {\__mcrule_debug_log:n {default}
      \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_tile_pattern:nnn {\__mcrule_circle:}{1pt}{1pt}},
  line-style / dense-circles       .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn {\__mcrule_circle:}{1pt}{0pt}},
  line-style / loose-circles       .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn {\__mcrule_circle:}{2pt}{2pt}},
  line-style / solid-circles       .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{1pt}{1pt}},
  line-style / dense-solid-circles .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{1pt}{0pt}},
  line-style / loose-solid-circles .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn {\__mcrule_solid_circle:}{2pt}{2pt}},
  line-style / dotted              .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {dotted}{\columnseprule}{1pt}{1pt}},
  line-style / densely-dotted      .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {densely~dotted}{\columnseprule}{1pt}{0pt}},
  line-style / loosely-dotted      .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {loosely~dotted}{\columnseprule}{2pt}{2pt}},
  line-style / dashed              .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {dashed}{3pt}{1.5pt}{1.5pt}},
  line-style / densely-dashed      .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {densely~dashed}{3pt}{1pt}{1pt}},
  line-style / loosely-dashed      .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_line_pattern:nnnn {loosely~dashed}{3pt}{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}},
  line-style / densely-dash-dot-dot .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_pattern_line:n{densely~dash~dot~dot}},
  line-style / loosely-dash-dot-dot .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_pattern_line:n{loosely~dash~dot~dot}},
  color                             .code:n = {
    \tl_set:Nn \l__mcrule_color_name_tl {#1}
    \__mcrule_set_rule_color:
  },
  color-model                       .code:n = {
    \tl_set:Nn \l__mcrule_color_model_tl {#1}
    \__mcrule_set_rule_color:
  },
  custom-line         .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tikz_picture:n {#1}},
  custom-pattern      .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_pattern:nnn #1},
  custom-tile         .code:n = \cs_set:Npn \__mcrule_divider:
    {\__mcrule_tile_pattern:nnn #1},
```

```
  width               .choice:,
  width / ultra-thin  .code:n = \dim_set:Nn \columnseprule {0.1pt},
  width / very-thin   .code:n = \dim_set:Nn \columnseprule {0.2pt},
  width / thin        .code:n = \dim_set:Nn \columnseprule {0.4pt},
  width / semithick   .code:n = \dim_set:Nn \columnseprule {0.6pt},
  width / thick       .code:n = \dim_set:Nn \columnseprule {0.8pt},
  width / very-thick  .code:n = \dim_set:Nn \columnseprule {1.2pt},
  width / ultra-thick .code:n = \dim_set:Nn \columnseprule {1.6pt},
  width / unknown     .code:n = {\dim_set:Nn \columnseprule {#1}},
  repeat              .int_set:N = \l__mcrule_repeat_int,
  repeat-distance     .dim_set:N = \l__mcrule_repeat_distance_dim,
  single              .meta:n = {
    repeat = 1,
    repeat-distance = #1
  },
  single              .default:n = \columnseprule,
  double              .meta:n = {
    repeat = 2,
    repeat-distance = #1
  },
  double              .default:n = \columnseprule,
  triple              .meta:n = {
    repeat = 3,
    repeat-distance = #1
  },
  triple              .default:n = \columnseprule,
}
```

4.6 User Interface

With only one command, this section is short. All we do is set whatever keys the user passes. All the real work is done in the definitions above.

`\SetMCRule` `\SetMCRule {⟨key-value list⟩}`

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

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

Change History

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.