The dashrule package[*]

Scott Pakin
scott+dash@pakin.org

August 9, 2009

Abstract

The dashrule package makes it easy to draw a huge variety of dashed rules (i.e., lines) in LaTeX. dashrule provides a command, `\hdashrule`, which is a cross between LaTeX's `\rule` and PostScript's `setdash` command. `\hdashrule` draws horizontally dashed rules using the same syntax as `\rule` but with an additional, `setdash`-like parameter that specifies the pattern of dash segments and the space between those segments. Because dashrule's rules are constructed internally using `\rule` (as opposed to, e.g., PostScript `\special`s) they are fully compatible with every LaTeX back-end processor.

1 Usage

`\hdashrule` LaTeX's `\rule` command draws a rectangular blob of ink with a given width, height, and distance above the baseline. The dashrule package introduces an analogous command, `\hdashrule`, which draws the same blob of ink, but horizontally dashed. `\hdashrule` takes five parameters, two of which are optional:

```
\hdashrule [⟨raise⟩] [⟨leader⟩] {⟨width⟩} {⟨height⟩} {⟨dash⟩}
```

The ⟨*raise*⟩, ⟨*width*⟩, and ⟨*height*⟩ parameters have the same meaning as in LaTeX's `\rule` macro: the distance to raise the rule above the baseline and the width and height of the rule.

Because `\hdashrule` is implemented in terms of TeX's primitive leader commands (`\leaders`, `\cleaders`, and `\xleaders`), the dash pattern must be repeated an integral number of times. ⟨*leader*⟩ specifies what to do with the extra whitespace (always less than the width of the dash pattern) that this requirement introduces. The default, which corresponds to TeX's `\leaders` command, adds space to both ends of the rule so the dash patterns from multiple `\hdashrule`s line up. If ⟨*leader*⟩ is `c`, which corresponds to TeX's `\cleaders` command, an equal amount of whitespace is added to both ends of the rule. If ⟨*leader*⟩ is `x`, which corresponds to

[*] This document corresponds to dashrule v1.2, dated 2009/08/09.

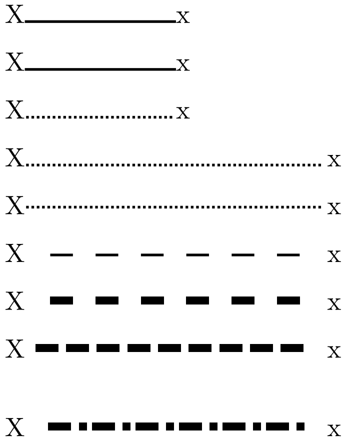
TeX's `\xleaders` command, the whitespace is divided up, and the same amount of whitespace separates each repetition of the dash pattern.

The ⟨*dash*⟩ argument specifies the dash pattern and is analogous to the *array* argument to PostScript's `setdash` function. That is, it is a list of space-separated ⟨*dimen*⟩s that alternate "on" and "off" distances. For instance, "`2pt 1pt`" means a 2 pt. rule, followed by a 1 pt. gap, followed by a 2 pt. rule, followed by a 1 pt. gap, and so forth. An odd number of ⟨*dimen*⟩s is no different; "`2pt`" alternates 2 pt. rules and 2 pt. gaps, and "`1pt 2pt 3pt`" repeats "1 pt. rule, 2 pt. gap, 3 pt. rule, 1 pt. gap, 2 pt. rule, 3 pt. gap." As a special case, an empty ⟨*dash*⟩ argument draws a solid rule.

2 Examples

The following are some typical ways to use `\hdashrule`. Each example changes from the previous in only one parameter. For clarity, underlines are used to indicate modified text, and the rule is bracketed by an upper- and lowercase "X".

```
\rule{2cm}{1pt}
\hdashrule{2cm}{1pt}{}
\hdashrule{2cm}{1pt}{1pt}
\hdashrule{4cm}{1pt}{1pt}
\hdashrule[0.5ex]{4cm}{1pt}{1pt}
\hdashrule[0.5ex]{4cm}{1pt}{3mm}
\hdashrule[0.5ex]{4cm}{1mm}{3mm}
\hdashrule[0.5ex]{4cm}{1mm}{3mm 3pt}
\hdashrule[0.5ex]{4cm}{1mm}{%
  3mm 3pt 1mm 2pt}
```

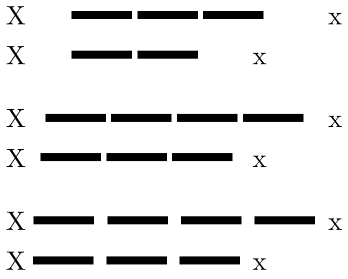


These next examples show the effect of using different leader types. Each leader is used with both a 4 cm wide rule and a 3 cm wide rule.

```
\hdashrule[0.5ex]{4cm}{1mm}{8mm 2pt}
\hdashrule[0.5ex]{3cm}{1mm}{8mm 2pt}

\hdashrule[0.5ex][c]{4cm}{1mm}{8mm 2pt}
\hdashrule[0.5ex][c]{3cm}{1mm}{8mm 2pt}

\hdashrule[0.5ex][x]{4cm}{1mm}{8mm 2pt}
\hdashrule[0.5ex][x]{3cm}{1mm}{8mm 2pt}
```



Notice how the dashes in the first pair of `\hdashrule`s line up; the rules in the second pair each have an equal amount of whitespace on either side of the rule; and the rules in the third pair have extra spaces within the dash pattern itself instead of around it. The `x` qualifier is rarely useful for dashed rules because it alters the pattern itself. However, `x` does enable rules with long dashes to better fill a comparatively small width, as in the following example:

```
\hdashrule[0.5ex][x]{3in}{2pt}{2cm 0pt}
```

X ━━━━━ ━━━━━ ━━━━━ x

The gaps in the above are clearly wider than `0pt`, but they *are* evenly spaced.

3 Differences from `setdash`

`\hdashrule` is different from PostScript's `setdash` command in the following ways:

- `setdash` takes on/off values in terms of PostScript points (TEX "big points" or "`bp`"), while `\hdashrule` requires explicit units.
- There is no equivalent of `setdash`'s *offset* parameter to specify a starting offset into the pattern. If you're desperate you can fake *offset* with a leading `\rule` and `\hspace`.

4 Implementation

```
⟨*package⟩
```

We load the ifmtarg package to help check if the final argument to `\hdashrule` is empty.

```
\RequirePackage{ifmtarg}
```

`\hdr@do@rule` This macro is exactly like LATEX's `\rule` except that the optional argument is required, and it has the side effect of pointing `\hdr@do@something` to `\hdr@do@skip`.

```
\def\hdr@do@rule[#1]#2#3{%
  \rule[#1]{#2}{#3}%
  \let\hdr@do@something=\hdr@do@skip
}
```

`\hdr@do@skip` This macro takes the same arguments as `\hdr@do@rule`, but instead of drawing a rule, it inserts an equivalent amount of horizontal whitespace. Additionally, it points `\hdr@do@something` to `\hdr@do@rule` as a side effect.

```
\def\hdr@do@skip[#1]#2#3{%
  \hspace*{#2}%
  \let\hdr@do@something=\hdr@do@rule
}
```

`\c@hdr@segments`
`\hdr@tally@segments`

Dash patterns containing an odd number of segments are treated differently from dash patterns containing an even number of segments. We therefore define a macro, `\hdr@tally@segments`, which counts the number of space-separated segments in a dash pattern and stores the tally in the `hdr@segments` counter. Note that `hdr@segments` should be initialized to `0` before invoking `\hdr@tally@segments`.

```
\newcounter{hdr@segments}
\def\hdr@tally@segments#1 {%
  \ifx#1!%
  \else
    \addtocounter{hdr@segments}{1}%
    \expandafter\hdr@tally@segments
  \fi
}
```

`\hdashrule`

This is the only macro in dashrule's external interface. (`\hdashrule@ii` does most of the work for `\hdashrule`, though.) All `\hdashrule` itself does is invoke `\hdashrule@i` with its first optional argument or `0.0pt` if none was provided. `\hdashrule@i`, in turn, invokes `\hdashrule@ii` with the two optional arguments, supplying `\empty` as the default value of the second optional argument.

```
\DeclareRobustCommand{\hdashrule}{\mbox{}\@testopt{\hdashrule@i}{0pt}}
```

`\hdashrule@i`

Supply `\empty` as the default second argument and call `\hdashrule@ii`.

```
\def\hdashrule@i[#1]{\@testopt{\hdashrule@ii[#1]}\empty}
```

`\hdashrule@ii`

Now we can do the real work for `\hdashrule`. `\hdashrule@ii` takes the following parameters:

#1	#2	#3	#4	#5
[⟨*raise*⟩]	[⟨*leader*⟩]	{⟨*width*⟩}	{⟨*height*⟩}	{⟨*dash*⟩}

The ⟨*raise*⟩, ⟨*width*⟩, and ⟨*height*⟩ parameters have the same meaning as in LaTeX's `\rule` macro. ⟨*leader*⟩ specifies the TeX leader function to use to fill ⟨*width*⟩ amount of space. It should be `c` for `\cleaders`, `x` for `\xleaders`, or nothing for ordinary `\leaders`. The ⟨*dash*⟩ argument specifies the dash pattern and is analogous to the *array* argument to PostScript's `setdash` function. That is, it is a list of space-separated ⟨*dimen*⟩s that alternate "on" and "off" distances.

```
\def\hdashrule@ii[#1][#2]#3#4#5{%
```

If the final argument, ⟨*dash*⟩, is empty, we fall back to using an ordinary `\rule` command. This is not terribly useful in practice but does make `\hdashrule` behave more like PostScript's `setdash`.

```
  \@ifmtarg{#5}{%
    \rule[#1]{#3}{#4}%
  }{%
```

Here begins the common case, in which the ⟨*dash*⟩ argument is nonempty.

`\hdr@do@something` The `\hdr@do@something` alias alternates between `\hdr@do@rule` and `\hdr@do@skip`, starting with `\hdr@do@rule`.

```
  \let\hdr@do@something=\hdr@do@rule
```

`\hdr@parse@dash` For every space-separated ⟨*dimen*⟩ in ⟨*dash*⟩, we invoke `\hdr@do@something` to draw a rule or a space, as appropriate. We define `\hdr@parse@dash` within `\hdashrule@ii` so we don't have to pass in `\hdashrule@ii`'s `#1` and `#4` on every invocation.

```
  \def\hdr@parse@dash##1 {%
    \ifx##1!%
    \else
      \hdr@do@something[#1]{##1}{#4}%
      \expandafter\hdr@parse@dash
    \fi
  }%
```

We now count the number of segments in the dash pattern so we can determine if we have an even or odd number of them.

```
  \setcounter{hdr@segments}{0}%
  \hdr@tally@segments#5 !
```

Finally, we invoke `\leaders`, `\cleaders`, or `\xleaders` to draw the dashed line, repeating the pattern until ⟨*width*⟩ space is filled. The trick here is that odd-lengthed pattern descriptions must be repeated to yield the complete pattern. For instance, the pattern "`1pt`" is actually short for "1 pt. rule, 1 pt. space," and "`2pt 4pt 6pt`" is an abridged version of "2 pt. rule, 4 pt. space, 6 pt. rule, 2 pt. space, 4 pt. rule, 6 pt. space." Although it is valid to repeat even-lengthed patterns as well—an earlier draft of `\hdashrule@ii` did just that—this produces inferior results because TEX's various leader commands do not split boxes. The longer the pattern, the less likely it will fit snugly into the given width.

```
  \ifodd\c@hdr@segments
    \csname#2leaders\endcsname
      \hbox{\hdr@parse@dash#5 #5 ! }%
      \hskip#3
  \else
    \csname#2leaders\endcsname
      \hbox{\hdr@parse@dash#5 ! }%
      \hskip#3
  \fi
  \mbox{}%
 }%
}
```

5 Future Work

dashrule v1.2 supports only horizontally dashed rules. Future versions (if any) may support vertically dashed rules as well. For the time being, the graphicx package's

\rotatebox can be used to define a \vdashrule in terms of a rotated \hdashrule.

The next logical step after adding a \vdashrule is to support dashed rectangles, which would be composed of \hdashrules and \vdashrules. Other possible enhancements would be a way of drawing dotted lines, presumably composed from the limited set of circle characters available in LaTeX's fonts.

Change History

v1.0

General: Initial version 1

v1.1

\hdashrule: Preceded the invocation of \@testopt with an empty box so that \hdashrule can now begin a paragraph . . . 4

v1.2

\hdr@parse@dash: Ended the rule with an empty box to enable \hdashrule to work within a tabular cell. Thanks to Piazza Luca for the bug report 5

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.