The \texttt{hhline} package*

David Carlisle

2020/01/04

Abstract

\texttt{\textcolor{red}{hhline}} produces a line like \texttt{\textcolor{red}{\hline}}, or a double line like \texttt{\textcolor{red}{\hline\hline}}, except for its interaction with vertical lines.

1 Introduction

The argument to \texttt{\textcolor{red}{hhline}} is similar to the preamble of an \texttt{array} or \texttt{tabular}. It consists of a list of tokens with the following meanings:

- A double hline the width of a column.
- A single hline the width of a column.
- A column with no hline.
| A vline which ‘cuts’ through a double (or single) hline.
: A vline which is broken by a double hline.
# A double hline segment between two vlines.
t The top half of a double hline segment.
b The bottom half of a double hline segment.
* \texttt{*\{3\}{==#}} expands to \texttt{==####==}, as in the *-form for the preamble.

If a double vline is specified (|| or ::) then the hlines produced by \texttt{\textcolor{red}{hhline}} are broken. To obtain the effect of an hline ‘cutting through’ the double vline, use a # or omit the vline specifiers, depending on whether or not you wish the double vline to break.

The tokens \texttt{t} and \texttt{b} must be used between two vertical rules. \texttt{|tb|} produces the same lines as #, but is much less efficient. The main use for these are to make constructions like \texttt{|t:} (top left corner) and \texttt{:b|} (bottom right corner).

If \texttt{\textcolor{red}{hhline}} is used to make a single hline, then the argument should only contain the tokens -, ~ and | (and *-expressions).

* This file has version number v2.04, last revised 2020/01/04.
An example using most of these features is:

\begin{tabular}{||cc||c|c||}
\hline
\hline
a & b & c & d \\
\hline
1 & 2 & 3 & 4 \\
\hline
i & j & k & l \\
\hline
w & x & y & z \\
\hline
\end{tabular}

The lines produced by \LaTeX's \texttt{\hline} consist of a single (\LaTeX primitive) \texttt{\hrule}. The lines produced by \texttt{\hhline} are made up of lots of small line segments. \LaTeX will place these very accurately in the .dvi file, but the program that you use to print the .dvi file may not line up these segments exactly. (A similar problem can occur with diagonal lines in the \texttt{picture} environment.)

If this effect causes a problem, you could try a different driver program, or if this is not possible, increasing \texttt{\arrayrulewidth} may help to reduce the effect.

2 The Macros

1 (*package)

\HH@box

Makes a box containing a double hline segment. The most common case, both rules of length \texttt{\doublerulesep} will be stored in \texttt{\box1}, this is not initialised until \texttt{\hhline} is called as the user may change the parameters \texttt{\doublerulesep} and \texttt{\arrayrulewidth}. The two arguments to \texttt{\HH@box} are the widths (ie lengths) of the top and bottom rules.

2 \def\HH@box#1#2\{

3 \hrule \@height \arrayrulewidth \@width #1

4 \vskip \doublerulesep

5 \hrule \@height \arrayrulewidth \@width #2}\}

\HH@add

Build up the preamble in the register \texttt{\toks0}.

6 \def\HH@add#1\{\toks0\expandafter{\the\toks0#1}}

\HH@xexpast

We ‘borrow’ the version of \texttt{@xexpast} from Mittelbach’s \texttt{array.sty}, as this allows # to appear in the argument list.

7 \def\HH@xexpast#1\#2\#3\#4\#5\%

8 \@tempcnta \#2

9 \toks0\=(\#1)\@emptokena\=(\#3)\%

10 \let\the@toksz\relax \let\the@toksz\relax

11 \def\@tempa\{\the@toksz\}%

12 \ifnum\@tempcnta \> \@whilenum\@tempcnta \> \do

13 \{\edef\@tempa\{\the@toksz\}\advance\@tempcnta \@one\%

14 \let\@tempb \HH@xexpast \else

15 \let\@tempb \HH@xnoop \fi
\def\the@toksz{\the\toks@}\def\the@toks{\the\@temptokena}\
16 \edef\@tempa{\@tempa}\%\n
17 \def\HH@xexnoop#1\@@{}\%\n
18 \expandafter\@tempb\@tempa#4\@@}\%\n
19 \def\HH@xexnoop\{\}%\n
Use a simplified version of \@mkpream to break apart the argument to \hhline.\nActually it is oversimplified, It assumes that the vertical rules are at the end\nof the column. If you were to specify c|@{xx}| in the array argument, then \hhline\nwould not be able to access the first vertical rule. (It ought to have an @\noption, and add \leaders up to the width of a box containing the @-expression. We use\na loop made with \futurelet rather than \@tfor so that we can use # to denote\nthe crossing of a double hline with a double vline.\n\if@firstamp is true in the first column and false otherwise.\n\if@tempsw is true if the previous entry was a vline (:, | or #).\n\def\hhline#1{\omit\@firstamptrue\@tempswafalse\n
21 Put two rules of width \doublerulesep in \box1\n\global\setbox\@ne\HH@box\doublerulesep\doublerulesep\n
If Mittelbach's array.sty is loaded, we do not need the negative \hskip's around\nvertical rules.\n\def\@tempc{\ifx\extrarowheight\HH@undef\hskip-.5\arrayrulewidth\fi}\%\n
Now expand the *-forms and add dummy tokens ( \relax and ') to either end\nof the token list. Call \HH@let to start processing the token list.\n\def\HH@xexpast\relax#1*0x\@@\toks@{}expandafter\HH@let\@tempa'}\n
\HH@let\n
Discard the last token, look at the next one.\n\def\HH@let#1{\futurelet\@tempb\HH@loop}\n
The main loop. Note we use \ifx rather than \if in version 2 as the new token\# is active.\n\def\HH@loop\%\n
If next token is ') stop the loop and put the lines into this row of the alignment.\n\if\@tempb'\def\next##1{\the\toks@\cr}else\let\next\HH@let\n
|, add a vertical rule (across either a double or single hline).\n\if\@tempb|\if\@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue\n
28 \HH@add{\@tempc\vline\@tempc}\else\n
\HH@add{\@tempc\HH@box\arrayrulewidth\@tempc}\else\n
29 |, add a broken vertical rule (across a double hline).\n\if\@tempb|\if\@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue\n
30 \HH@add{\@tempc\HH@box\arrayrulewidth\arrayrulewidth\@tempc}\else\n
31 |, add a double hline segment between two vlines.\n\if\@tempb#\if\@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatrue\n
32 \HH@add{\@tempc\HH@box\arrayrulewidth\arrayrulewidth\@tempc}\else\n
33 #, A column with no hline (this gives an effect similar to \cline).\n\if\@tempb~\@tempswafalse\n
34 \if\@firstamp\@firstampfalse\else\HH@add{&\omit}\fi\n
35 \HH@add{\hfil}\else\n
36 }
-, add a single hline across the column.
37 \ifx\@tempb-\@tempswafalse
38 \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
39 \HH@add{\leaders\hrule\@height\arrayrulewidth\hfil}\else
=, add a double hline across the column.
40 \ifx\@tempb=\@tempswafalse
41 \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi
Put in as many copies of \box1 as possible with \leaders, this may leave gaps at
the ends, so put an extra box at each end, overlapping the \leaders.
42 \HH@add
43 \rlap{\copy\@ne}\leaders\copy\@ne\hfil\llap{\copy\@ne}\else
t, add the top half of a double hline segment, in a \rlap so that it may be used
with b.
44 \ifx\@tempb t\HH@add{\rlap{\HH@box\doublerulesep\z@}}\else
b, add the bottom half of a double hline segment in a \rlap so that it may be
used with t.
45 \ifx\@tempb b\HH@add{\rlap{\HH@box\z@\doublerulesep}}\else
space, Gobble the space and loop again.
46 \ifx\@tempb\@sptoken\let
47 \PackageWarning{hhline}{\meaning\@tempb space ignored in \noexpand\hhline argument}%
48 \MessageBreak}%
49 \fi\fi\fi\fi\fi\fi\fi\fi\fi
Go around the loop again.
50 \next
\HH@spacelet Helper macro to gobble a space token and continue the loop.
52 \lowercase{\def\HH@spacelet}{\futurelet\@tempb HH@loop}
53 (/package)