

The `hhline` package*

David Carlisle
carlisle@cs.man.ac.uk

1994/05/23

Abstract

`\hhline` produces a line like `\hline`, or a double line like `\hline\hline`, except for its interaction with vertical lines.

1 Introduction

The argument to `\hhline` is similar to the preamble of an `array` or `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.
- * `*{3}{==#}` expands to `==#==#==#`, as in the `*`-form for the preamble.

If a double vline is specified (`||` or `::`) then the hlines produced by `\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 `t` and `b` must be used between two vertical rules. `|tb|` produces the same lines as `#`, but is much less efficient. The main use for these are to make constructions like `|t:` (top left corner) and `:b|` (bottom right corner).

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

*This file has version number v2.03, last revised 1994/05/23.

An example using most of these features is:

```

\begin{tabular}{|c|c|c|c|}
\hline|t:==:t:==:t|}
a&b&c&d\
\hline{|:==:|~|~|}
1&2&3&4\
\hline{#==#~|#}
i&j&k&l\
\hline{||--||--||}
w&x&y&z\
\hline{|b:==:b:==:b|}
\end{tabular}

```

a	b	c	d
1	2	3	4
i	j	k	l
w	x	y	z

The lines produced by L^AT_EX's `\hrule` consist of a single (T_EX primitive) `\hrule`. The lines produced by `\hline` are made up of lots of small line segments. T_EX 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 `picture` environment.)

If this effect causes a problem, you could try a different driver program, or if this is not possible, increasing `\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 `\doublerulesep` will be stored in `\box1`, this is not initialised until `\hline` is called as the user may change the parameters `\doublerulesep` and `\arrayrulewidth`. The two arguments to `\HH@box` are the widths (ie lengths) of the top and bottom rules.

```

2 \def\HH@box#1#2{\vbox{%
3 \hrule \@height \arrayrulewidth \@width #1
4 \vskip \doublerulesep
5 \hrule \@height \arrayrulewidth \@width #2}}

```

`\HH@add` Build up the preamble in the register `\toks@`.

```

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

```

`\HH@xexpast` We 'borrow' the version of `\@xexpast` from Mittelbach's `array.sty`, as this allows `#` to appear in the argument list.

`\HH@xexnoop`

```

7 \def\HH@xexpast#1*#2#3#4\@{%
8 \@tempcnta #2
9 \toks@=#1}\@temptokena=#3}%
10 \let\the@toksz\relax \let\the@toks\relax
11 \def\@tempa{\the@toksz}%
12 \ifnum\@tempcnta >0 \@whilenum\@tempcnta >0\do
13 {\edef\@tempa{\@tempa\the@toks}\advance \@tempcnta \m@ne}%
14 \let \@tempb \HH@xexpast \else
15 \let \@tempb \HH@xexnoop \fi

```

```

16 \def\the@toksz{\the\toks@}\def\the@toks{\the\@temptokena}%
17 \edef\@tempa{\@tempa}%
18 \expandafter \@tempb \@tempa #4\@{}
19
20 \def\HH@exnoop#1\@{}

```

`\hhline` Use a simplified version of `\mkpream` to break apart the argument to `\hhline`. Actually it is oversimplified, It assumes that the vertical rules are at the end of the column. If you were to specify `c|@{xx}|` in the array argument, then `\hhline` would not be able to access the first vertical rule. (It ought to have an `@` option, and add `\leaders` up to the width of a box containing the `@`-expression. We use a loop made with `\futurelet` rather than `\tfor` so that we can use `#` to denote the crossing of a double hline with a double vline.

`\if@firstamp` is true in the first column and false otherwise.

`\if@tempswa` is true if the previous entry was a vline (`:`, `|` or `#`).

```

21 \def\hhline#1{\omit\@firstamptrue\@tempswafalse

```

Put two rules of width `\doublerulesep` in `\box1`

```

22 \global\setbox\@ne\HH@box\doublerulesep\doublerulesep

```

If Mittelbach's `array.sty` is loaded, we do not need the negative `\hskip`'s around vertical rules.

```

23 \xdef\@tempc{\ifx\extrarowheight\HH@undef\hskip-.5\arrayrulewidth\fi}%

```

Now expand the `*`-forms and add dummy tokens (`\relax` and `'`) to either end of the token list. Call `\HH@let` to start processing the token list.

```

24 \HH@expast\relax#1*0x\@{\toks@}\expandafter\HH@let\@tempa'

```

`\HH@let` Discard the last token, look at the next one.

```

25 \def\HH@let#1{\futurelet\@tempb\HH@loop}

```

`\HH@loop` The main loop. Note we use `\ifx` rather than `\if` in version 2 as the new token `~` is active.

```

26 \def\HH@loop{%

```

If next token is `'`, stop the loop and put the lines into this row of the alignment.

```

27 \ifx\@tempb'\def\next##1{\the\toks@\cr}\else\let\next\HH@let

```

`|`, add a vertical rule (across either a double or single hline).

```

28 \ifx\@tempb|\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatru

```

```

29 \HH@add{\@tempc\vline\@tempc}\else

```

`:`, add a broken vertical rule (across a double hline).

```

30 \ifx\@tempb:\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatru

```

```

31 \HH@add{\@tempc\HH@box\arrayrulewidth\arrayrulewidth\@tempc}\else

```

`#`, add a double hline segment between two vlines.

```

32 \ifx\@tempb#\if@tempswa\HH@add{\hskip\doublerulesep}\fi\@tempswatru

```

```

33 \HH@add{\@tempc\vline\@tempc\copy\@ne\@tempc\vline\@tempc}\else

```

`~`, A column with no hline (this gives an effect similar to `\cline`).

```

34 \ifx\@tempb~\@tempswafalse

```

```

35 \if@firstamp\@firstampfalse\else\HH@add{&\omit}\fi

```

```

36 \HH@add{\hfil}\else

```

-, add a single hline across the column.

```
37 \ifx\@tempb-\@tempswafalse
38     \if@firststamp\@firststampfalse\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@firststamp\@firststampfalse\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
```

Otherwise ignore the token, with a warning.

```
46 \PackageWarning{hhline}%
47     {\meaning\@tempb\space ignored in \noexpand\hhline argument%
48     \MessageBreak}%
49 \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
```

Go around the loop again.

```
50 \next}
```

```
51 \endpackage)
```