

# Le package `hhline`\*

David Carlisle  
carlisle@cs.man.ac.uk

Traduit de l'anglais par:  
Jean-Pierre Drucbert

2014/10/28

## Abstract

Ce package étend les possibilités de lignes horizontales et verticales dans un environnement `array` ou `tabular`. Il introduit la nouvelle commande `\hhline` qui produit une ligne comme `\hline` ou une double ligne comme `\hline\hline`, sauf pour leur interaction avec les lignes verticales.

## 1 Introduction

L'argument de `\hhline` est similaire au préambule d'un environnement `array` ou `tabular`. Il consiste en une liste d'éléments (tokens) ayant les significations suivantes:

- = Une double hline sur la largeur d'une colonne.
- Une simple hline sur la largeur d'une colonne.
- ~ Une colonne sans hline.
- | Une vline qui 'coupe' une hline simple ou double.
- : Une vline qui est brisée par une double hline.
- # Un double segment hline entre deux vlines.
- t La moitié supérieure d'un double segment hline.
- b La moitié inférieure d'un double segment hline.
- \* `{3}{==#}` s'expande en `==#==#==#`, comme dans la `*`-forme pour le préambule.

Si une double vline est spécifiée (`||` ou `::`) alors les hlines produites par `\hhline` sont brisées. Pour obtenir l'effet d'une hline 'passant à travers' la double vline, utilisez un `#` ou omettez les spécificateurs de vline, selon que vous vouliez ou non que la double vline soit brisée.

Les tokens `t` et `b` doivent être utilisés entre deux traits verticaux. `|tb|` produit les mêmes lignes que `#`, mais est moins efficace. L'utilisation principale pour ceci

---

\*Ce fichier a le numéro de version v2.03, dernière mise à jour le 2014/10/28.

est de faire des constructions telles que `|t:` (coin supérieur gauche) et `:b|` (coin inférieur droit).

Si `\hhline` est utilisée pour faire une simple hline, alors l'argument ne doit contenir que des tokens `-`, `~` et `|` (et des `*`-expressions).

Voici un exemple utilisant la plupart de ces possibilités:

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

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

Les lignes produites par la commande `\hline` de  $\text{\LaTeX}$  sont formées d'une seule primitive `\hrule` de  $\text{\TeX}$ . Les lignes produites par `\hhline` sont faites d'une quantité de petits segments de ligne.  $\text{\TeX}$  les placera très précisément dans le fichier `.dvi`, mais le programme que vous utilisez pour imprimer le fichier `.dvi` peut ne pas les aligner exactement. (Un problème similaire peut survenir avec les lignes obliques dans l'environnement `picture`).

Si ceci pose un problème, vous pouvez soit essayer un autre programme d'impression, si cela est possible, ou alors augmenter `\arrayrulewidth` pour essayer d'en réduire les effets.

## 2 Les macros (section non traduite)

```
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 \hhline 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
\HH@xexnoop # to appear in the argument list.
```

```

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@xexnoop#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@xexpast\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\@tempswatrue
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\@tempswatrue
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\@tempswatrue
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@firststamp\@firststampfalse\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

```

Go around the loop again.

```

50 \next}
51 \</package>

```