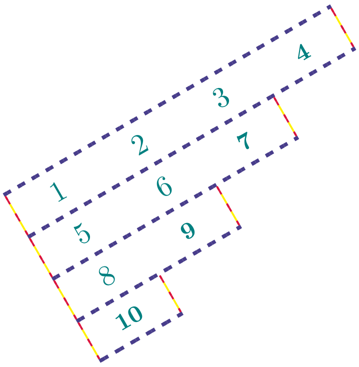
# Tabu Package Updates

January 12, 2019

The tabu package patches internal commands of many packages and is liable to stop working if those internal interfaces change.

Unfortunately the original author appears to be out of contact and the package had not been updated for some years.

This manual documents version 2.8 from 2011, but the distributed package contains fixes reported since that time, see `https://github.com/tabu-fixed/tabu`



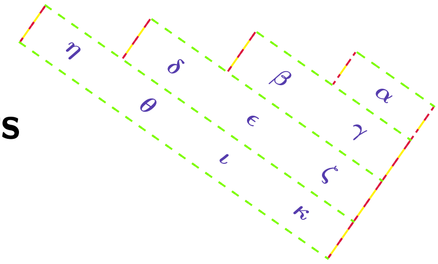
# $\tau_{\aleph}b\subset$[*]

## tabu and longtabu

## Flexible LaTeX tabulars

FC

2011/02/26 – version 2.8 release



### Abstract

This package defines a single environment **tabu** to make all kinds of tabulars in text or in math mode provided that they do not split across pages.

An environment **longtabu** – based on D. Carlisle longtable package – is also provided to make tabulars that can stretch out on several pages, while keeping some features (not all of them) of the **tabu** environment.

**tabu** is more flexible that **tabular**, **tabular***, **tabularx** and **array** and extends the possibilities. All tabulars in this document were made with the **tabu** environment, *of course*... The implementation is optimised to minimise the measurements required to put all together.

$\tau_{\aleph}b\subset$ likes colors too, with special lines that are able to keep the alignment of the surrounded text... and also like numbers with the possibility to embed siunitx S (or s) columns. $\tau_{\aleph}b\subset$ does not modify any of the macro defined by array.sty or in the LaTeX kernel[1].

$\tau_{\aleph}b\subset$ requires $\varepsilon$-TeX and the standard package array.sty. Natural widths of columns are computed (but not printed ) by the code of varwidth by D. Arseneau. Finally **longtabu** is based on longtable.

## Contents

---

[*] This documentation is produced with the DocStrip utility, and required $\tau_{\aleph}b\subset$ with its linegoal option.

| | | |
|---|---|---|
| ⟶ To get the package, | run: | etex tabu.dtx |
| ⟶ To get the documentation | run (thrice): | pdflatex tabu.dtx |
| To get the index, | run: | makeindex -s gind.ist tabu.idx |

The .dtx file is embedded into this pdf file thank to embedfile by H. Oberdiek.

[1] Inside the tabu environment a few macros are modified... this was compulsory !

*6.1.1 \tabudecimal*
*6.1.2 You should know how it works...*

# Summary of the features provided by τℵb⊂

| | |
|---|---|
| **tabu** | is like `tabular` in text mode and like `array` in math mode when there is no `X` column in its preamble |
| **longtabu** | is like `longtable` with the possibility to use `tabu X` columns and vertical lines with the extended syntax. |
| **`{tabu} to` ⟨*dimen*⟩** | specifies the target width of the whole tabular. This is like `tabular*` with an automatic stretchability that can be overwritten with `@{\extracolsep {dimen}}` in front of the preamble. |
| **`{tabu} spread` ⟨*dimen*⟩** | has no equivalent in LaTeX: the final width is ⟨*dimen*⟩ wider than the natural width that can be obtained with `spread 0pt`. |
| **`\|[width,color]`** | vertical lines have an optional parameter. |
| **`X[coef,align,type]`**<br>**`X[coef,align,type,$]`** | `X` columns widths are adjusted in order for the whole tabular to fit the target width. The target width is a dimension either:<br>➟ directly specified with `{tabu} to`⟨*dimen*⟩<br>➟ computed from the natural width: `{tabu} spread`⟨*dimen*⟩<br>➟ by default `\linewidth` (or `\linegoal` with the linegoal package option).<br>`coef` scales the widths of the `X` columns, if there are more than one `X` column.<br>`align` is either `r`, `c`, `l` or `j` (or R C L J) and `type` can be `p` (default), `m` or `b`.<br>`X[$]` makes a math `X` column (*ie.* `>{$}X<{$}`)<br>`X[$$]` display math `X` column: `>{$\displaystyle }X<{$}` |
| **`X[−coef,align,type]`** | `X` columns widths are first computed with the absolute value: \|coef\|. Then the width is made narrower down to the natural width of the column if possible.<br>In any case, the final width does not exceed the one obtained with `X[`\|coef\|`]`. |
| **`X[X options]{S[S options]}`** | Embed a siunitx `S` column into a `tabu-X` column. |
| **`\everyrow {code}`** | Allows to add horizontal lines automatically for every row.<br>The settings can be changed inside the `tabu` |
| **`\rowfont [align]{font spec}`** | Modify the font and optionally the alignment of each cell in one row. |
| **`\tabulinesep` =⟨*dimen*⟩** | More control on vertical spacing of lines in a way very close to cellspace's method (dynamic vertical spacing adjustment). |
| **`\extrarowsep` =⟨*dimen*⟩** | Control vertical spacing (`\extrarowheight` and `\extrarowdepth`): fixed vertical spacing adjustment. `\tabulinesep` generally gives better results. |
| **`\tabudecimal {\usermacro }`** | a help to align numbers easily inside a column. |
| **`\savetabu {user-name}`** | Saves the `tabu` preamble and its parameters. The command must appear at the end of a line. |
| **`\usetabu {user-name}`** | Makes a `tabu` of exactly the same shape as the one saved with `\savetabu`. All parameters (`target`, `preamble`, `stretch` *etc.*) are restored.<br>This command is put alone in the preamble in place of the columns specifications. |
| **`\preamble {user-name}`** | Makes a `tabu` with the same preamble as the one saved with `\savetabu`. The only `preamble` is restored, not the `target` nor any other parameter.<br>This command is put alone in the preamble in place of the columns specifications. |

## Summary of the features provided by Tℵb⊂

| | |
|---|---|
| `\tabulinestyle {line spec}` | Sets the current line style to be used for `\|` and `\tabucline` |
| `\newtabulinestyle {name=spec,...}` | Defines a line style for use with `\tabucline [name]` or with `\|[name]` |
| `\tabucline [spec]{start-stop}` | Draws a line comparable to `\hline`. The line ⟨*spec*⟩ can contain information for making a dash or dotted line (f.ex. `[on 3pt off 6pt]`) and a color name.<br>The line spec can also be defined with `\newtabulinestyle` |
| `\taburulecolor \|dbl rule sep\|{rule color}` | sets the color for rules (`\hline`, `\firsthline`...) |
| `\taburowcolors [skip]`⟨*number*⟩`{first .. last}` | Sets the color series to make alternate background colors for rows |
| `\tabuphantomline` | inserts a phantom (*ie.*invisible) line inside the `tabu`<br>May be usefull with `\multicolumn` in some cases. |
| `\tracingtabu` $= 0, 1, 2, 3, 4$ | Reports informations in the .log file about the steps of the algorithm for `tabu X` columns, and the informations saved by `\savetabu`. |

# 1 Examples and counterexamples

## Let's begin in colors !

Tℵb⊂ provides facilities to put horizontal and vertical leaders in a tabular. The package xcolor must be loaded of course. Background colors for cells are left to package colortbl which is fully compatible with Tℵb⊂.

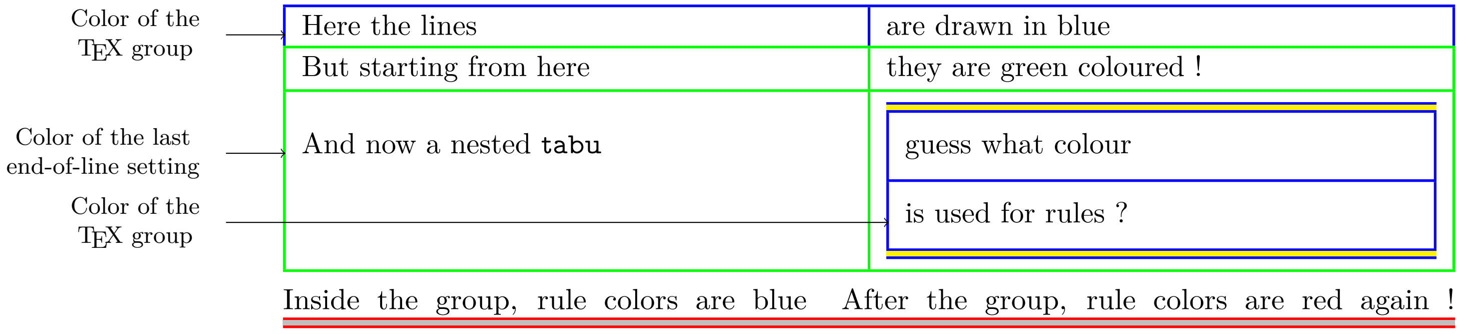
## 1.1 "Locally global" settings and their scopes

\tabulinestyle
\taburulecolor
\taburowcolors
\everyrow

Tℵb⊂ observes TEX grouping levels for the settings of rule colors (\taburulecolor) and styles (\tabulinestyle), and \everyrow. There is however a subtility for nested tabu environments as described in this example:

Listing 1: Locally global settings and their scopes

```
\taburulecolor |gray!50|{red} \arrayrulewidth=1pt
{
  \taburulecolor |yellow|{blue}
     \begin{tabu}{|X|X|} \hline
     Here the lines & are drawn in blue \\ \taburulecolor{green} \hline
     But starting from here & they are green coloured !        \\ \hline
     And now a nested tabu & \begin{tabu}{X} \firsthline\hline
                              guess what colour \\ \hline
                              is used for rules ?\\ \lasthline\hline
                            \end{tabu} \\ \hline
     \end{tabu}
    % Inside the group, rule colors are blue
}
% After the group, rule colors are red again !
\begin{tabu}{X}\hline\hline\indent\end{tabu}
```



| Here the lines | are drawn in blue |
|---|---|
| But starting from here | they are green coloured ! |
| And now a nested tabu | guess what colour<br>is used for rules ? |

Inside the group, rule colors are blue After the group, rule colors are red again !

The "rules" are the following:

- If outside of a tabu environment, the settings are local to the TEX group. Every tabular drawn inside this group will inherit from the settings of that group.
- If \taburulecolor (or \everyrow or \tabulinestyle) is used inside a cell of the tabular, this is the same: the settings a local to that cell, and any nested tabular will inherit from the setting of that cell.
- When used after the end of a row, the settings are globally changed from that point until the end of the tabular, or until a new setting is set at the end of a further row (TEXnically, this is done inside a \noalign group). But a nested tabu does not inherit from this "global" setting, and inherits from the settings of the TEX group instead.

If \arrayrulecolor or \doublerulesepcolor (from package colortbl) are used instead of \taburulecolor then colors are globally overwritten.

A counterexample from the xcolor package: \rowcolors does not like \cline, \cmidrule etc.[2]

```
\rowcolors{2}{green!25}{yellow!50}
\begin{tabular}{cc} \toprule
\repeatcell2{
    rows=5,
    text/col1=test ,
    text/col2=row \number\rownum} \\
    test & other row \number\rownum \\ \cmidrule
       {1-2}
    test & other row \number\rownum \\
    test & other row \number\rownum \\    \
       bottomrule
\end{tabular}
```

| | |
|---|---|
| test | row 1 |
| test | row 2 |
| test | row 3 |
| test | row 4 |
| test | row 5 |
| test | other row 6 |
| test | other row 8 |
| test | other row 9 |

The \rownum counter is not reliable in the case of \cline or \cmidrule.

In addition, the first coloured row is yellow, while one could have expected it green...

For tabu color changes are called at \everyrow:

```
\taburowcolors [2] 2{green!25 .. yellow!50}
\begin{tabu}{*2{X[c]}} \toprule
\repeatcell2{
    rows=5,
    text/col1=test ,
    text/col2=row \thetaburow} \\
    test & other row \thetaburow \\ \cmidrule
       {1-2}
    test & other row \thetaburow \\
    test & other row \thetaburow \\    \bottomrule
\end{tabu}
```

| | |
|---|---|
| test | row 1 |
| test | row 2 |
| test | row 3 |
| test | row 4 |
| test | row 5 |
| test | other row 6 |
| test | other row 7 |
| test | other row 8 |

tabu does not use “real” alternate colors but colorseries provided by package xcolor. This allow some gradations:

```
\taburowcolors 5{green!25 .. yellow!50}
\begin{tabu}{X[-1]X}
\repeatcell 2{
      rows=10,
      text/col1=test ,
      text/col2={Row number
                  \row$=$\thetaburow},
      }
\end{tabu}
```

| | |
|---|---|
| test | Row number 1=1 |
| test | Row number 2=2 |
| test | Row number 3=3 |
| test | Row number 4=4 |
| test | Row number 5=5 |
| test | Row number 6=6 |
| test | Row number 7=7 |
| test | Row number 8=8 |
| test | Row number 9=9 |
| test | Row number 10=10 |

## 1.2 X column widths computation

The new algorithm implemented in version 2.8 requires only one measure of the width of the table in any case. This speeds up the convergence of the algorithm.

```
\begin{tabu} to 140mm {|X[1,l] | X[2,c] | X[3,c] | X[1,r]|}
      |\dotfill |     & Text     & Text       & Text \\
      Text            & Text     & Text       & Text
\end{tabu}
```

Example provided by M. Manuel Kuehner

| | | | |
|---|---|---|---|
| \| . . . . . . \| | Text | Text | Text |
| Text | Text | Text | Text |
| 1X= 17.5$mm$ | 2X = 35$mm$ | 3X = 52.5$mm$ | 1X = 17.5$mm$ |

$$X = (140mm - 8 \times \texttt{\textbackslash tabcolsep} - 5 \times \texttt{\textbackslash arrayrulewidth})/7 = 17.4896mm$$

---

[2] Because color changes are done at \everycr, which is not exactly the same as tabu \everyrow!

## 1.3 Inserting Verbatim material (fancyvrb)

Though the content of the `tabu` environment is collected for measuring purpose, it is possible to insert verbatim material with the `tabu*` variant of the environment. The content is then carefully collected and re-scanned (with `\scantokens`). During the process, the `@` letter is read with the category code it has been given at the entry inside the environment (it is possible to say `\makeatletter` before `\begin {tabu* }`).

Example:

| | |
|---|---|
| It is possible to insert Verbatim material with some `\csname control sequences \endcsname` inside a `tabu` and inside `X` columns. Negativ coefficients work well too, adjusting the width of the `X` column to the natural width if it is finally less than the width computed with the absolute value of the coefficient.<br>A complete `Verbatim` environment is also admissible.<br>But you must use the star form of the environment: `tabu*` which uses `\scantokens`. | Verbatim environments must be put<br>alone on their lines (in the input<br>file) for nothing is allowed<br>after \begin{Verbatim} or<br>\end{Verbatim}.<br>Another point to know is that<br>\begin and \end control sequences<br>should match otherwise, you must<br>enclose the Verbatim environment<br>inside braces.<br>This is related to the fact that tabu<br>collects its body, and looks for<br>matching pairs of \begin ... \end ! |

`tabu*` is useless when nested inside another tabular. The star form of the environment should be used only for the outermost table ! Comments are removed, unless the % character is given a category code of 12 (or 11) before the entry inside the environment.

```
\tabulinestyle{on2pt Crimson!60 off3pt yellow!50} \tabulinesep=2mm
\makeatletter \@makeother\%
\begin{tabu*}spread 0pt {|X[-1]X|} \tabucline-
This is a small \Verb+\Verbatim+\par
insertion
&
\begin{Verbatim}[listparameters={\topsep=-\ht\strutbox}]
And this is a complete % with some comments
Verbatim environment   % every now and then
\end{Verbatim}
\\ \tabucline-
\end{tabu*}
```

| | |
|---|---|
| Here a small \Verbatim<br>insertion | And this is a complete % with some comments<br>Verbatim environment % every now and then |

It's not possible to insert a lstlisting environment presently, but you can save such an environment in a `\vbox` and insert it inside the `tabu` of course.

## 1.4 Maths inside tabu X columns

```
$\begin{tabu}spread .5in |{*3{X[$c]}}|
    \alpha & \beta & \gamma \\
    \sum_i \frac{a_i}{x_i} & 0 & \cdot \\
\end{tabu}$
```

$$\left|\begin{array}{ccc} \alpha & \beta & \gamma \\ \textstyle\sum_i \frac{a_i}{x_i} & 0 & \cdot \end{array}\right|$$

X[$] columns

```
$\begin{tabu}spread .5in |{*3{X[$$c]}}|
    \alpha & \beta & \gamma \\
    \sum_i \frac{a_i}{x_i} & 0 & \cdot \\
\end{tabu}$
```

$$\left|\begin{array}{ccc} \alpha & \beta & \gamma \\ \displaystyle\sum_i \frac{a_i}{x_i} & 0 & \cdot \end{array}\right|$$

X[$$] columns

### 1.5 Embedding **sunitx** S columns inside X columns

A S column from siunitx can be embedded into a X column of Tℵb⊂... with the following limitations:

- The X column must be centered: X[c] to keep the alignment,
- The optional alignment parameter of \rowfont must not be used.

```
\newcolumntype Y{S[group-four-digits=true,
                   round-mode=places,
                   round-precision=2,
                   round-integer-to-decimal=true,
                   per-mode=symbol,
                   detect-all]}
\tabucolumn Y
\tabulinestyle{1pt GreenYellow}
\begin{tabu}spread 8pt{|*2{Y|}c} \tabucline-
\rowfont\bfseries
      {January}  &{February} &... \\
                  \tabucline[1pt on2pt GreenYellow
                     ]-
      12.324    &745.32    &... \\
      21.13     &0         &... \\
      213.3245  &12.342    &... \\
      2143.12   &324.325   &... \\\tabucline-
\end{tabu}
```

| January | February | ... |
|---|---|---|
| 12.32 | 745.32 | ... |
| 21.13 | 0.00 | ... |
| 213.32 | 12.34 | ... |
| 2 143.12 | 324.33 | ... |

Column widths are not exactly the same

```
\newcolumntype Z{X[c]{%
    S[group-four-digits=true,
      round-mode=places,
      round-precision=2,
      round-integer-to-decimal=true,
      per-mode=symbol]}}
\tabucolumn Z
\begin{tabu}spread 8pt{|*2{Z|}c} \tabucline-
\rowfont\bfseries
      {January}  &{February} &... \\
        \tabucline[1pt on2pt GreenYellow]-
      12.324    &745.32    &... \\
      21.13     &0         &... \\
      213.3245  &12.342    &... \\
      2143.12   &324.325   &... \\\tabucline-
\end{tabu}
```

| **January** | **February** | **...** |
|---|---|---|
| 12.32 | 745.32 | ... |
| 21.13 | 0.00 | ... |
| 213.32 | 12.34 | ... |
| 2 143.12 | 324.33 | ... |

Column widths are exactly the same

\tabucolumn is there to say Tℵb⊂ that the column type has to be treated with a high priority in the rewriting process.

Another possibility to print number is provided with \tabudecimal.

# 2 The tabu environment

## 2.1 tabu, tabu to and tabu spread

\begin {tabu} [pos] {tabular preamble}
\begin {tabu} to ⟨*dimen*⟩ [pos] {tabular preamble}
\begin {tabu} spread ⟨*dimen*⟩ [pos] {tabular preamble}

The tabu environment behaves mostly like tabular: the preamble is parsed by the macros in array.sty and some measures are performed before printing. tabu improves tabular and array:

- **footnotes** and index words are allowed inside tabu, unlike tabularx, footnote links are not broken when compiled with hyperref. The syntax \footnote [number]{⟨*text*⟩} is allowed in tabu and longtabu (this is not implemented for longtable yet...)
- X columns are implemented with an *optional* parameter for the **width-coefficient** (which can be negativ: see next section), the **alignment** (r, c, l, or j, and R, C, L or J for

ragged2e settings) and the **column type** (`p`, `m`, or `b`).
`tabu` has a default target width when used with `X` columns, making nesting even easier.

- You are used to the `tabular` environment in text mode, and `array` environment in math mode, but `tabu` works in both modes and its name does not change... `X` columns are also possible in math mode; delarray shortcuts for delimiters are available in both math and text modes.
- A `tabu` environment can contain another tabular of any kind: `tabular`, `tabular*` , `tabularx` or `tabu` itself can be placed in any cell of a `tabu`. Conversely, `tabu` can be placed in a `tabular`, `tabularx` *etc.*.
- `tabu` provides facilities for vertical and horizontal lines, and for the insertion of verbatim text inside `X` columns.
- `tabu` is fully compatible with colortbl, delarray, hhline, makecell, booktabs, siunitx, dcolumn, warpcol, *etc.*. When you are inside a `tabu` environment, you can use `\raggedleft`, `\raggedright` and `\centering` without special care about `\arraybackslash` and conversely `\\` has its "normal" meaning inside a list of items that may appear in a `X` column...

`\begin {tabu} to`⟨*dimen*⟩ is like `tabular*` but the inter-columns space is given a stretchability of 1fil, in other words `@{\extracolsep {0pt plus 1fil}}` is inserted by default at the beginning of the tabular preamble, unless another value for `\extracolsep` is specified. Therefore "`tabu to`" fills in width the specified ⟨*dimen*⟩.

`\begin {tabu} spread`⟨*dimen*⟩ does a tabular whose width is ⟨*dimen*⟩ wider than its natural width. `@{\extracolsep {0pt plus 1fil}}` is inserted by default if ⟨*dimen*⟩> 0.

## 2.2 longtabu, longtabu to and longtabu spread

```
\begin {longtabu} [l | c | r] {tabular preamble}
\begin {longtabu} to ⟨dimen⟩ [l | c | r] {tabular preamble}
\begin {longtabu} spread ⟨dimen⟩  [l | c | r] {tabular preamble}
```

`longtabu` is just like `tabu` but page breaks are allowed between rows of the table. `longtabu` is based on the longtable package which must be loaded, and all features of the `longtable` environment works inside `longtabu`: `\endhead`, `\endfirsthead`, `\endfoot`, `\endlastfoot` and `\caption`.

`longtabu` enhances the `longtable` environment with the possibility to use `X` columns and line specifications for horizontal and vertical rules. `longtabu` is thus much easier than ltxtable.

The following commands provided for `tabu` do not work with `longtabu`:

| **tabu command** | **Not available** | **Not (yet) implemented** | **Comment** |
|---|---|---|---|
| `\tabucline` | | ❇ | `\tabucline` does not care of page breaks presently: use `\hline` instead. |
| `\usetabu` | ✗ | | but `\savetabu` and `\preamble` work. |
| mathematical mode | ✗ | | `longtable` is not designed to work in math mode. |
| delarray shortcuts | ✗ | | a delimiter cannot be spanned over pages... |
| `\tabuphantomline` | ✗ | | useless inside `longtabu` |

However, `tabu X columns`, `\rowfont`, `\extrarowsep`, `\tabulinesep`, `\tabudecimal`, `\tabucline` (with restrictions on page breaks), `\taburulecolor`, `\tabulinestyle`, `\taburowcolors`, `\preamble`, `{longtabu} to` , `{longtabu} spread` work inside `longtabu`.

## 2.3 tabu X columns – Mastering horizontal space

`tabu` `X` columns can be viewed as an enhancement of `tabularx` `X` columns, but do not interact with them, for they are defined only for a short time during the parsing of the preamble:

- **width coefficients** can optionally be given to `X` columns
  ex. **`X[2.5]X[1]`** is the same as **`X[2.5]X`** and the same as **`X[5]X[2]`**
  This means that the first `X` column will be two and a half wider than the second one or that the first `X` column width will be $^5/_7$ of the whole tabular width.

  | | |
  |---|---|
  | X[2.5] | X |

- **negativ width coefficients** can be given to `X` columns:
  ex. **`X[-2.5]X[1]`** or **`X[-2.5]X`** or **`X[-5]X[2]`**
  In this case, the first `X` column will be ***at most*** two and a half wider than the second one, and if the *natural width* of the first `X` column is finally less than 2.5 × (the width of the second column) then it will be narrowed down to this natural width.
  **The following `tabu`s have the same preamble:**
  `\begin {tabu} to\linewidth {|X[−2.5c]|X[c]|}:`

  | | |
  |---|---|
  | X[−2.5] | X |

  | | |
  |---|---|
  | Negativ coefficients make `X` columns close to standard `l`, `c` and `r` columns. | X |

- horizontal alignment specification is made easier with **`X[5,r]X[2,c]`** for example. Vertical alignment can be specified as well with `X[5,r,m]X[2,p,c]` (commas are not required, but `X[2cm]` or `X[4pc]` could be misunderstood – not by TEX: by you...).

  | Modifier | Meaning | Default |
  |---|---|---|
  | `l`, `c`, `r`, `j`, `L`, `C`, `R`, `J` | left, centered, right, justified | `j` |
  | `p`, `m`, `b` | `X` column is converted into `p`, `m` or `b` column | `p` |
  | `$` | `X[$]` is a shortcut for: `>{$}X<{$}` | |
  | `$$` | `X[$$]` is a shortcut for: `>{$\displaystyle }X<{$}` | |

- `tabu` `X` columns can be spanned with `\multicolum`.

- `tabu` `X` columns can be used with "`tabu spread`" for small tabulars.

- `tabu` `X` columns can contain any type of `tabular`, `tabular*`, `tabularx` or `tabu` without special care about the syntax. `tabu` can also be put inside `tabular`, `tabular*` and `tabularx`. As long as `tabu` with `X` columns has a *default target*, nesting `tabu` with `X` columns is easy. Furthermore, the default global alignment of a nested `tabu` is `t` (for `top`) while the default global alignment of a `tabu` in a paragraph is `c` (for `centered`).

- The "algorithm" (or the arithmetic) to get the target width for `tabu` `X` columns is the same as the one used by `tabularx`. `\hfuzz` is the "tolerance" for the whole tabular width. We use ε-TEX `\dimexpr`

  instead of TEX primitives (with round/truncate bias correction).

- Convergence to the target width is optimised: the `\halign` preamble is not re-built at each trial, but only expanded again, until the target is reached. Though optimized, the process is the same as the one implemented for `tabularx` and in particular the content of the `tabu` environment is collected as soon as a `tabu` `X` column is found in the preamble. This implies restrictions on catcode modifications and verbatim text inside a `tabu` with `X` columns.

- If the width of the whole tabular is not specified with "`tabu to`" it is considered to be `\linewidth`. The linegoal package option makes the default width equal to `\linegoal`. Compilation must then be done with pdfTEX either in `pdf` or `dvi` mode, and package `linegoal` is loaded. `\linegoal` requires pdfTEX for its `\pdfsavepos` primitive and the `zref-savepos`: if the `tabu` is not alone in its paragraph *ie.*if the target is not `\linewidth`, then two compilations (or more) are required to get the correct target.
  Default target for nested `tabu` environments is always `\linewidth`, which equals to the column width inside `p`, `m`, `b` and `X` columns.

- As long as the `\halign` content is expanded more than once, protections against counters incrementation, whatsits (*write*) index entries, footnotes *etc.*. are set up: the mechanism of `tabularx` is reimplemented and enhanced for `tabu` `X` columns. `\tabuDisableCommands` can be used to neutralize the expansion of additional macros during the trials.

### X columns with “tabu spread”

`tabu` `X` columns can be used with “`tabu spread`” to adjust the column widths of tabulars that contain only small pieces of text. The question is: how to make a tabular the width of the line, with 6 columns; the columns 1, 2, 5 and 6 are of equal widths and the widths of columns 3 and 4 are only one half. As possible solution:

```
\begin{tabu} to\linewidth{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

But the text in each cell is very short: one single character, and you prefer the table to be tight, but don’t know the exact width of the whole:

```
\begin{tabu} spread 0pt{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
 1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

But now it’s definitely too narrow, then give it some more space:

```
\begin{tabu} spread 2in{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
 1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

`tabu spread` is useless with long columns: the following tabular was made with this preamble:

```
\begin{tabu} spread 3cm{@{}X[9]X[4]|X|}
```

| | | |
|---|---|---|
| "Like the air we breathe, Sherlock Holmes is everywhere. His pipe-smoking, deer stalkered image peers at us from ads in Yellow Pages, to signs for neighbourhood crime-watch; from billboards to the classroom; from film and television to the public library, and now over the Internet. He long ago transcended the boundaries of 19th Century London[3] to become an international best-seller and has been accepted as part of British folklore. Holmes is alive to millions." | There the text was too long, and `tabu spread` behaves as if you didn’t give it a target.<br>The result of this example is the same as if one had written `\begin {tabu}to\linewidth`. | **Sherlock Holmes** |
| The “official” web site: http://www.sherlockholmes.com/ | | |

In the preamble, `@{}` means that the margin is removed.

### Negativ width coefficients for X columns

```
\tabulinestyle{3pt ForestGreen}
\begin{tabu}{|X[-1m]|X[c m]|}
              \tabucline- \savetabu{FirstNegativTest}
  $\begin{tabu}({X[-1$]X[-1$c]})
     \alpha & \beta \\
     \gamma & \delta + \epsilon + \zeta + \eta + \theta
  \end{tabu}$
  &
  This is a tabu with negativ width coefficients for \textt X columns
  \\ \tabucline-
\end{tabu}
```

| | |
|---|---|
| $\begin{pmatrix}\alpha & \beta \\ \gamma & \delta+\epsilon+\zeta+\eta+\theta\end{pmatrix}$ | This is a tabu with negativ width coefficients for `X` columns |

---

[3] Capital of the U.K. (too see a linked footnote)

| | |
|---|---|
| $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta+\epsilon+\zeta+\eta+\theta \end{pmatrix}$ | And this is the same with \tabulinesep set to $2pt$. |

### Multicolumn in tabu

\tabuphantomline

The process of \multicolumn implies the TEX primitive \omit which discards the tabular preamble for the spanned columns. Discarding the preamble means discarding the information about the widths of the columns. This explains why the following example does not work properly:

```
\begin{tabu}{|X|X|X[2]|} \tabucline-
\multicolumn2{|c|}{Hello} & World \\ \tabucline-
\end{tabu}
```

The correct result can be obtained by the mean of a phantom line, that will remain invisible unless your preamble contains special @ or ! columns that prints some text:

```
\begin{tabu}{|X|X|X[2]|}  \tabucline-
\multicolumn2{|c|}{Hello} & World  \\ \tabucline-
\tabuphantomline
\end{tabu}
```

| | |
|---|---|
| Hello | World |

Remember you may need \tabuphantomline in conjunction with \savetabu and \usetabu with \multicolumn. Even if it is possible to add a \tabuphantomline in any line of the tabu, it is a good practice to append it *at the end* of the tabu, for it may introduce indesirable side effects on vertical alignment otherwise, when tabu is nested inside another tabular.

In particular, \tabuphantomline should not be followed by \cr or \\ or \tabularnewline...

The need for this command could disappear in a future release, but this requires a complete new implementation of \multicolumn...

## 2.4 \tabulinesep and \extrarowsep – Mastering vertical space

\tabulinesep =⟨*dimen*⟩
\tabulinesep =^⟨*dimen*⟩
\tabulinesep =_ ⟨*dimen*⟩
\tabulinesep =^⟨*dimen*⟩_⟨*dimen*⟩
\tabulinesep =_⟨*dimen*⟩^⟨*dimen*⟩

\tabulinesep sets the *minimal* vertical space allowed between the cell content and the cell border. The macro may be prefixed by \global (even inside a \noalign group)[4].

It is possible to set the "top limit" (a TEX dimension called \abovetabulinesep) and the "bottom limit" independently with the syntaxes:

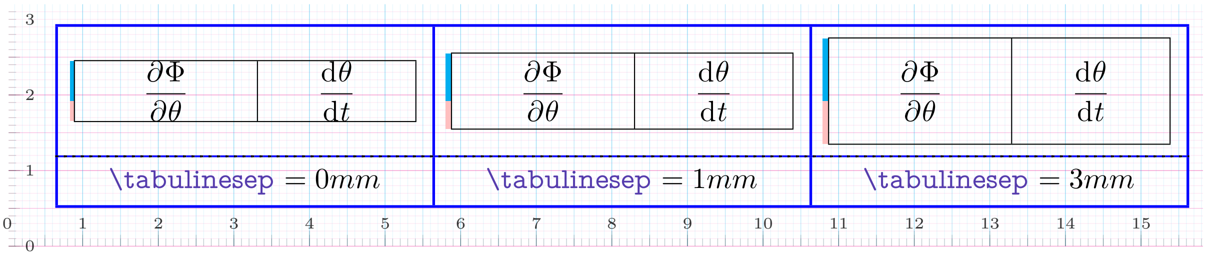
| | |
|---|---|
| \tabulinesep =^⟨*dimen*⟩ | sets \abovetabulinesep |
| \tabulinesep =_⟨*dimen*⟩ | sets \belowtabulinesep |
| \tabulinesep =_⟨*dimen*⟩^⟨*dimen*⟩ | sets \belowtabulinesep and \abovetabulinesep. |

These parameters can be used in text and math modes to give more vertical space between lines, especially when using math formulae.

Examples (with \tracingtabu = 3 and interfaces-\papergraduate to see the struts):

---

[4] However \tabulinesep is not a dimension ! You can't test, for example, \ifdim \tabulinesep $> 0pt$ ! Test \abovetabulinesep and \belowtabulinesep instead, if needed.



| | | |
|---|---|---|
| $\frac{\partial\Phi}{\partial\theta}$ $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ | $\frac{\partial\Phi}{\partial\theta}$ $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ | $\frac{\partial\Phi}{\partial\theta}$ $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ |
| `\tabulinesep` $= 0mm$ | `\tabulinesep` $= 1mm$ | `\tabulinesep` $= 3mm$ |

0 1 2 3
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

`\tabulinesep` is a soft parameter, and leads to rows which do not share the same height.

```
\extrarowsep =⟨dimen⟩
\extrarowsep =^⟨dimen⟩
\extrarowsep =_ ⟨dimen⟩
\extrarowsep =^⟨dimen⟩_⟨dimen⟩
\extrarowsep =_⟨dimen⟩^⟨dimen⟩
```

`\extrarowsep` is an extra vertical space which is added to each row, inconditionally. `array.sty` provides the TEX dimension `\extrarowheight` and τℵb⊂ provides `\extrarowdepth` in addition.

As a result, the rows can share the same height/depth but the spacing is not dynamic. `\tabulinesep` can be used even with positive values for `\extrarowsep`, for `tabu` inserts only one strut per row and vertical spacing computations are possible in all cases.

The macro can be prefixed by `\global` as well, even inside a `\noalign` group[5].

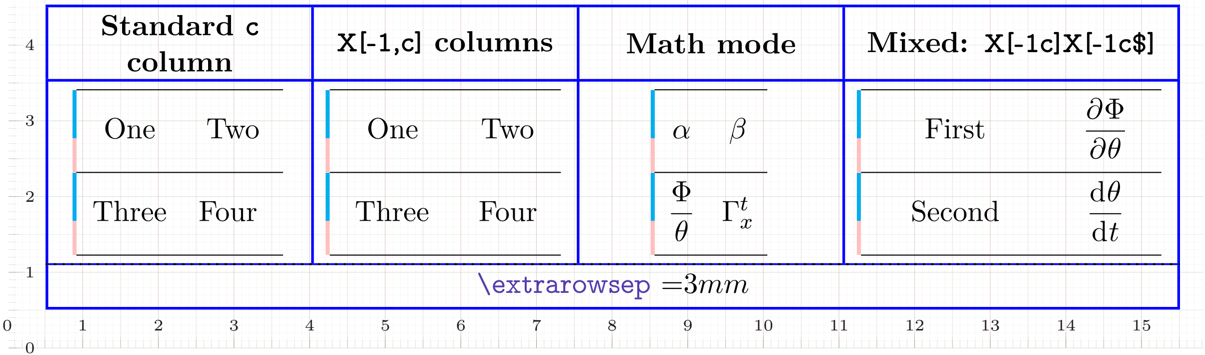
Set `\extrarowheight` and `\extrarowdepth` to different values, with the syntaxes:

| | |
|---|---|
| `\extrarowsep` =^⟨*dimen*⟩ | sets `\extrarowheight`<br>`\extrarowdepth` is unchanged |
| `\extrarowsep` =_⟨*dimen*⟩ | sets `\extrarowdepth`<br>`\extrarowheight` is unchanged |
| `\extrarowsep` =_⟨*dimen*⟩^⟨*dimen*⟩ | sets `\extrarowdepth` and `\extrarowheight`. |

Both `\extrarowheight` and `\extrarowdepth` are scaled by `\arraystretch` (a scaling *macro*[6] of `array.sty`) if `\arraystretch` $> 1$…

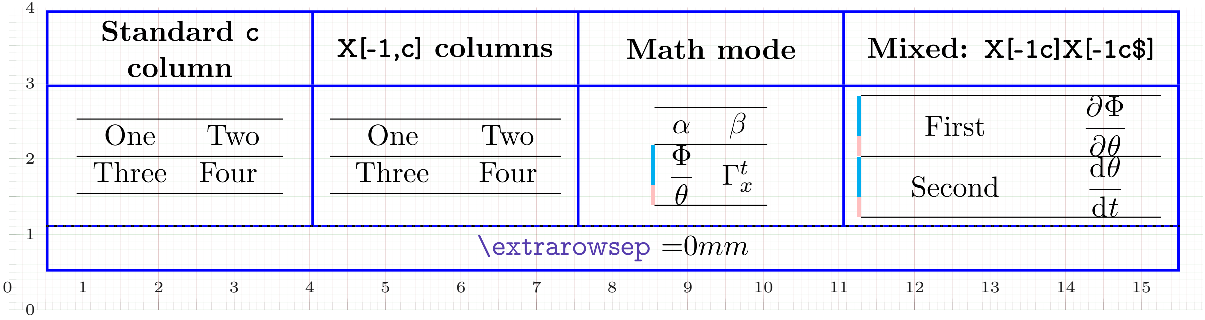
These parameters can be used in text and math modes.

Examples (with `\tracingtabu` $= 3$ and interfaces-`\papergraduate` to see the struts):



| **Standard c column** | **X[-1,c] columns** | **Math mode** | **Mixed: X[-1c]X[-1c$]** |
|---|---|---|---|
| One Two | One Two | $\alpha$ $\beta$ | First $\frac{\partial\Phi}{\partial\theta}$ |
| Three Four | Three Four | $\frac{\Phi}{\theta}$ $\Gamma_x^t$ | Second $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ |
| `\extrarowsep` $=3mm$ | | | |

0 1 2 3 4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



| **Standard c column** | **X[-1,c] columns** | **Math mode** | **Mixed: X[-1c]X[-1c$]** |
|---|---|---|---|
| One Two | One Two | $\alpha$ $\beta$ | First $\frac{\partial\Phi}{\partial\theta}$ |
| Three Four | Three Four | $\frac{\Phi}{\theta}$ $\Gamma_x^t$ | Second $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ |
| `\extrarowsep` $=0mm$ | | | |

0 1 2 3 4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

---

[5] However `\extrarowsep` is not a dimension ! You can't test, for example, `\ifdim` `\extrarowsep` $> 0pt$ ! Test `\extrarowheight` and `\extrarowdepth` instead, if needed.

[6] `\arraystretch` is not a dimension but a macro that stores a scaling factor.

## 2.5 tabu in math mode

On the left, you can see the famous Maxwell-Lorentz equations for electromagnetic field in vacuum, publicated in 1873.

$$\left(\begin{array}{rl} \operatorname{div}\vec{E} &= \dfrac{\rho}{\epsilon_0} \\ \operatorname{div}\vec{B} &= 0 \\ \overrightarrow{\operatorname{rot}}\,\vec{E} &= -\dfrac{\partial\vec{B}}{\partial t} \\ \overrightarrow{\operatorname{rot}}\,\vec{B} &= \mu_0\,\vec{j} + \mu_0\epsilon_0\,\dfrac{\partial\vec{E}}{\partial t} \end{array}\right.$$

In this example, the big `tabu` is: `\begin {tabu} to\linewidth {XX[-1$]}`.

The nested `tabu` (in math mode) uses delarray shortcut: its preamble is: `\begin{tabu}({rl}`.

`\tabulinesep` has been set to **`2pt`**. Horizontal rules are booktabs `\toprule` and `\bottomrule`.

| **`array`** | **`tabu`** | **`tabu spread`** $1em$ |
|---|---|---|
| $\left\lvert\begin{matrix}\alpha & \beta \\ \gamma & \delta\end{matrix}\right\rvert$ | $\left\lvert\begin{matrix}\alpha & \beta \\ \gamma & \delta\end{matrix}\right\rvert$ | $\left\lvert\begin{matrix}\alpha & \beta \\ \gamma & \delta\end{matrix}\right\rvert$ |

Here, vertical lines are made with delarray shortcuts: `$\begin{tabu} spread 1em |{cc}|`

Vertical lines inside the tabular preamble gives: $\left\lvert\begin{matrix}\alpha & \beta \\ \gamma & \delta\end{matrix}\right\rvert$

This was an example of `\savetabu`...`\usetabu` to keep the alignment.

# 3 Lines leaders and colors inside tabu

## 3.1 First important remark

The features provided in this section are quite experimental: they are not generally taken for good typography. You can use $\tau_{\aleph}b\subset$ with package booktabs for example, which provides properly designed commands for horizontal rules in tabulars. arydshln is pretty good too, but it modifies a huge amount of macros of `array.sty`, something that $\tau_{\aleph}b\subset$ does not.

Lines in `tabu` printed in this document are mostly made with booktabs.

## 3.2 Vertical lines: `|` has an optional parameter

Inside `tabu` environment, the vertical line marker `|` has an *optional* argument which is the width of the vertical rule. The default width remains `\arrayrulewidth` of course. The optional argument for `|` can also contain the name of a color. color *names* are only possible, not a color specification by the mean of a color model. The width of the line if specified, must come before the color name and... as for `X` columns parameters, commas are optional.

Example:

│█│Hello│World│█│ — The `tabu` you see on the left was made with the code on the right.

```
\begin {tabu}{||[5pt]|c|c||[5pt]|}
    Hello & World
\end {tabu}
```

│🟥│Hello│World│🟪│ — The `tabu` you see on the left was made with the code on the right.

```
\begin {tabu}{||[5pt red]|c|c||[5pt Indigo]|}
    Hello & World
\end {tabu}
```

This example was printed inside a `tabu*` whose preamble is: `X[−1m] X[m] X[-2m]`

It is not a necessary to protect the optional argument with braces: `[{...}]`. because $\tau_{\aleph}b\subset$ takes care the `|` token to be rewritten before any other column type (the same for `tabu` `X` columns, and siunitx `S` columns). The rewriting process is divided into three stages under control inside a `tabu` environment.

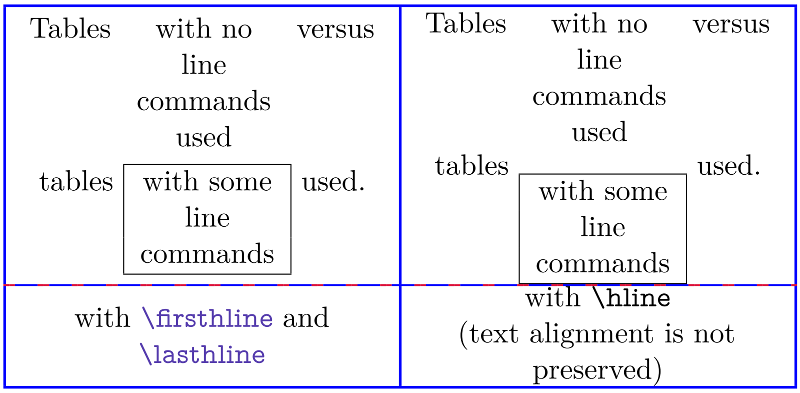
Use `tabu*`
to insert Verbatim
environments !

## 3.3 Multiple \firsthline and \lasthline

```
\firsthline [extratabsurround]        make multiple lines !
\firstline [extratabsurround]\hline
\lasthline [extratabsurround]
\lastline [extratabsurround]\hline
```

\firsthline and \lasthline are defined in `array.sty` and can be used to preserve the alignment of text, when using horizontal lines. Besides, the optional argument can be used to change (locally) the `\extratabsurround` dimension.

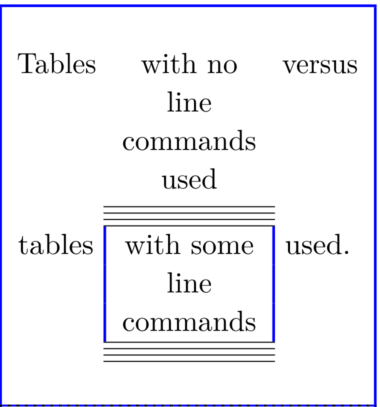
The example of `array` documentation is:



Now with `tabu` you can make double, triple (or more) \firsthline or \lasthline as in:

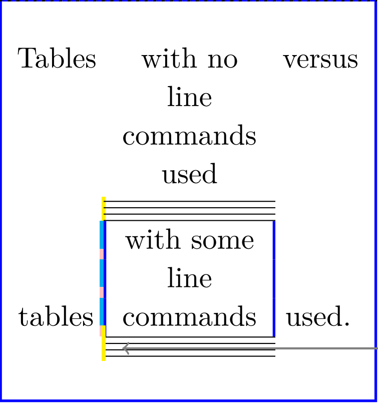
Top alignment

```
 Tables
 \begin {tabu}[t]{c}
   with no\\ line \\ commands \\ used
 \end {tabu}
 versus tables
 \begin {tabu}[t]{|c|}
   \firsthline \hline \hline \hline
    with some \\ line \\ commands \\
   \lasthline \hline \hline \hline
\end {tabu} used.
```



Bottom alignment

```
 Tables
 \begin {tabu}[t]{c}
   with no\\ line \\ commands \\ used
 \end {tabu}
 versus tables
 \begin {tabu}[b]{|c|}
   \firsthline \hline \hline \hline
   with some \\ line \\ commands \\
   \lasthline \hline \hline \hline
 \end {tabu} used.
```



`\firsthline \firsthline \firsthline` is equivalent to: `\firsthline \hline \hline`
and also to: `\firstline \hline \hline \hline`

But the optional argument must come in *first position*: `\firsthline [extratabsurround] ...`

The same for \lastline.

In yellow you can see the `\extratabsurround` strut, because \tracingtabu = 3 for this `tabu`

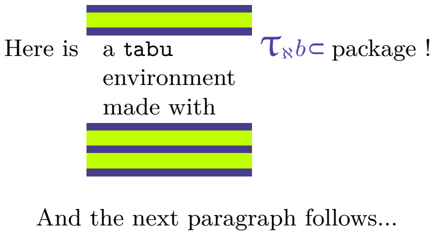
## 3.4 More style for lines

\taburulecolor {⟨*rule color*⟩}
\taburulecolor |⟨*double rule sep color*⟩|{⟨*rule color*⟩}

\taburulecolor sets (in a "locally-global" way) the color to be used for \hline, \firsthline, \lasthline and also vertical lines if the standard line style is used (the standard line style is active after \tabulinestyle {} or after \tabureset).

The optional parameter enclosed by vertical bars: |⟨*double rule sep color*⟩| is the color to set between two adjacents rules. If not specified, double (or triple...) rules are separated by a vertical space (\vskip).

```
\taburulecolor |lime|{DarkSlateBlue}
\arrayrulewidth=1mm \doublerulesep=2mm
Here is
\begin{tabu}spread 0pt {X[-1]}
                  \firsthline\hline
 a  tabu \\
 environment \\
 made with \\ \lasthline[5mm]\hline\hline
\end{tabu} \TabU package !\par
And the next paragraph follows...
```


Here is a tabu environment made with $\tau_{\aleph}b\subset$ package !

And the next paragraph follows...

\tabulinestyle {⟨*line style specification*⟩}

\tabulinestyle sets the line style for vertical (|) and horizontal lines (*ie.*\tabucline: \hline, \firsthline etc. are not modified by \tabulinestyle)

The line specification is of the form:

3pt rule color on 4pt dash color off 5pt gap color
rule color on 4pt dash color off 5pt gap color
on 4pt dash color off 6pt gap color
3pt rule color
on 4pt dash color
off 5pt
3pt
Named style defined by \newtabulinestyle

Well... any parameter is optional. Obviouly the rule color is the same as the dash color and the former overwrites the latter if both are given.

Your color names can contain spaces but:

- If the first character in the line specification is not a letter, then it is taken as a dimension: the thickness of the line. Otherwise, the default thikness is used *ie.*\arrayrulewidth.
- Your color names must not contain any series of characters that match one the patterns:
  on? off?

  where ? is a character of category 12, different from ! and possibly preceded by spaces. I don't think this is a real limitation...

\newtabulinestyle {⟨*style=line spec., style=line spec., ...*⟩}babel

This command defines a line style to be used in the first optional argument of \tabucline (horizal lines) or the optional argument of |(vertical lines) or with \tabulinestyle (locally-global style).

Style names and color names are babel-protected.

\tabucline [style or spec.]{start-end}

\tabucline is an attempt to give a versatile command to make horizontal lines:

- \tabucline is pretty good with vertical lines even if the thickness of the line grows up,

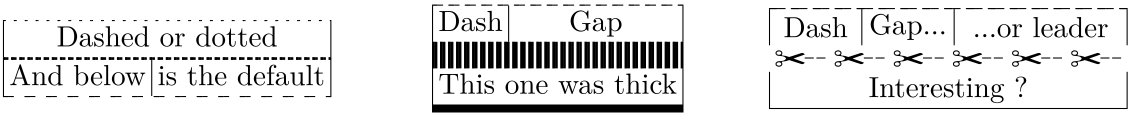
- `\tabucline` takes care of `\extrarowheight`,
- `\tabucline` can make horizontal dashed lines, with a pgf/TikZ syntax:
  `\tabucline` [⟨*width*⟩ on⟨*dash*⟩ off⟨*gap*⟩]{⟨*first column*⟩-⟨*last column*⟩}
- alternatively, you can give `\tabucline` a `\hbox` to make a leader with it: The ⟨*spec.*⟩ must then begin with `\hbox`, `\box` or `\copy`,
- finally you can give `\tabucline` a color *name*, after the line specification.

Any parameter can be omitted.

| | | |
|---|---|---|
| [1pt on 1.5pt off 2pt] | `\tabucline [1pt on 1.5pt off 2pt]{1-4}` | draws a horizontal dashed line of width 1pt. Dashes are 1.5pt long and gap width is 2pt. The line is drawn between columns 1 and 4. Here there are only 2 columns and the line stops at column 2. |
| [1.5pt] | `\tabucline [1.5pt]{-}` | draws a horizontal solid line of width 1.5pt between the first and the last column. |
| default | `\tabucline {2-}` | draws a horizontal solid line of width `\arrayrulewidth` between the second column and the last one. |
| [on 2pt red] | `\tabucline [on 2pt red]{-5}` | draws a horizontal dashed line between columns 1 and 5 of width `\arrayrulewidth`. Dashed are 2pt long and gap width is 4pt (the default). |

| **Define the line style** | **Use the line style** |
|---|---|
| `\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}` | `\tabucline [myline]{-}` |

Or use a leader or a box to make a leader with it
directly in the argument of `\tabucline`
`\tabucline [\hbox {$\scriptstyle \star $}]{1-3}`



| Dashed or dotted | |
|---|---|
| And below | is the default |

| Dash | Gap |
|---|---|
| This one was thick | |

| Dash | Gap... | ...or leader |
|---|---|---|
| Interesting ? | | |

## 3.5 Automatic horizontal lines and row colors

`\everyrow {code}`

`\everyrow` can be used to insert horizontal lines automatically:

```
\begin{tabu}to .5\linewidth{cX[2mc]X} \tabucline[1pt]-
                    \everyrow{\tabucline[on 2pt]-}
This is      &a small example &of a \texttt{tabu}              \\
which        &automatically   &inserts                        \\
a horizontal &line after      &each of its row \everyrow{} \\ \tabucline[1
    pt]-
\end{tabu}
```

| | | |
|---|---|---|
| This is | a small example | of a tabu |
| which | automatically | inserts |
| a horizontal | line after | each of its row |

`\everyrow` can be used in `longtabu` as well. The syntax is like `\everycr`: a token-like syntax, and braces are mandatory: `\everyrow {argument}`.

### \taburowcolors [first line]⟨*number*⟩{first .. last}

\taburowcolors sets the alternate colors to be used on every row of the tabular. The command can be used before a `tabu` environment or inside it, at the end of a row.

The optional parameter [first line] tells the first row from which background colors are starting – this optional parameter has no effect when \taburowcolors is used at the end of a row: background are starting immediately in this case.

⟨*number*⟩ is the number of colors in the color series. If not specified, it defaults to 2 (for alternate rows color).

Finally ⟨*first*⟩ and ⟨*last*⟩ are the first and the last colors in the colorseries.

Example:

```
\taburowcolors[2] 3{Crimson!30 ..
                         ForestGreen!40}
\taburulecolor |GreenYellow|{OrangeRed}
\arrayrulewidth=1pt \doublerulesep=1.5pt
\everyrow{\hline\hline}
\begin{tabu} {X[-1]X}
This is      &just a test          \\
and i think  &it will              \\
look         &rather bad           \\
for          &i've not             \\
chosen       &the colors           \\
with care.   &i can't              \\
say          &less...              \\
\taburowcolors 2{Crimson .. ForestGreen}
1            &This is Crimson      \\
2            &This is ForestGreen  \\
3            &This is Crimson      \\
4            &This is ForestGreen \\
\end{tabu}
```

| | |
|---|---|
| This is | just a test |
| and i think | it will |
| look | rather bad |
| for | i've not |
| chosen | the colors |
| with care. | i can't |
| say | less... |
| 1 | This is Crimson |
| 2 | This is ForestGreen |
| 3 | This is Crimson |
| 4 | This is ForestGreen |

### \tabureset

To go back to "standard" parameters, τℵb⊂ provides the command \tabureset which basically does:

\tabulinesep = $0pt$
\tabulinestyle {}
\taburowcolors {}
\extrarowsep = $0pt$
\everyrow {}
\extratabsurround = $0pt$
\taburulecolor ||{}

## 4 Modifying the font and the alignment in one row: \rowfont

### \rowfont [alignment]{font specification}

Inside a `tabu` environment, you can modify the font for each cell in a row. \rowfont has priority over column font specification, exactly like \rowcolor (package colortbl) has priority over \columncolor.

The alignment of each cell in one row can also be changed to:

l = left    or for ragged2e settings: L
c = center    C
r = right    R
j = justify    J

Any other value for the optional ⟨*alignment*⟩ parameter is silently ignored. If ragged2e is not loaded, `L R C` and `J` are synonymous with the lowercase equivalent.

```
\begin{tabu}{|X|X[-1]|}          \tabucline-
  \rowfont[c]\bfseries
   This               &Is                   \\\tabucline[on 2pt,blue]-
  \xpackage{tabu}     &package              \\\tabucline[off 2pt blue]-
```

```
    \rowfont[r]\itshape
    for                     &\texttt{tabu} and \texttt{longtabu} \\\tabucline-
  \end{tabu}
```

| This | Is |
|---|---|
| tabu | package |
| *for* | *tabu and longtabu* |

# 5 Saving and restoring a tabu

### `\savetabu` {⟨*user-name*⟩}

The command `\savetabu` can be used at the end of any line of a `tabu` environment to save the parameters of a `tabu` environment. The saving is always global. This allows to easily make tabulars which share exactly the same shape throughout your document. This can also be used as a kind of `tabbing` environment which is able to remember the tabs positions...

If the ⟨*user-name*⟩ has been used before, an info is displayed in the .log file and the previous settings are overwritten.

With the `\tracingtabu` $> 0$, informations about the saved parameters are reported in the .log file.

Recalling saved parameters are done with `\usetabu` (complete recovery) or `\preamble` (partial recovery of the preamble only).

### `\usetabu` {⟨*user-name*⟩}

`\usetabu` is the complement of `\savetabu`: it can be put alone in the `tabu` preamble instead of the usual columns specifications to restore any previous settings saved with `\savetabu`.

The ⟨*user-name*⟩ must exist otherwise, you get an error.

`\usetabu` is a help to **make several tabulars of exactly the same shape, same target, same preamble.** The only parameter that can be changed is the optional vertical position parameter for the whole tabular.

`\usetabu` does not work with longtabu .

`\usetabu` locally restores:

- the preamble[7].
- the vertical position `[c]`, `[b]` or `[t]`, unless another position is specified.
- the target width of the `tabu` in points: the saved target width does not contain any control sequence: it is fixed and stored in points.
- the width of `tabu` `X` columns: those widths are not calculated any more – even in the case of negativ coefficients – and `X` columns are directly transformed into `p`, `m` or `b` columns of the same widths as the ones that where calculated at the time of `\savetabu`
- `\tabcolsep` (or `\arraycolsep` in math mode) `\extrarowheight`, `\extrarowdepth`, `\arraystretch` and `\extratabsurround`
- `\arrayrulewidth`, `\doublerulesep` and the parameters for `\everyrow` `\taburulecolor`, `\tabulinestyle`, and `\taburowcolors`
- `\minrowclearance`, (package colortbl)

`\abovetabulinesep` and `\belowtabulinesep` are not restored, because they are related to the content of the tabular rather than to its shape.

Example:

```
\tabcolsep=12pt \extrarowsep=1mm
\tabulinestyle{on 1pt ForestGreen}
```

---

[7] The complete `\halign`-preamble is restored.

```
\begin{tabu}to .7\linewidth{|XXX|X[c]|} \savetabu{mytabu} \tabucline-
 This & is & tabu & package                            \\ \tabucline-
\end{tabu}
```

| | | | |
|---|---|---|---|
| This | is | tabu | package |

```
\tabureset
\begin{tabu}{\usetabu{mytabu}}            \tabucline-
\multicolumn3{|c}{This is tabu} & package \\ \tabucline-
\tabuphantomline
\end{tabu}
```

| | |
|---|---|
| This is tabu | package |

If one day you use `tabu`, you will have the idea to restore a `tabu` while modifying its target, or adding new columns... \savetabu and \usetabu have not been thought for this purpose, and you may have unexpected results.

**\preamble {⟨*user-name*⟩}**

\preamble can also be used after \savetabu. This is a variant of \usetabu that locally restores:

- the `tabu` (or `longtabu`) preamble.
- the vertical position [c], [b] or [t] (or [c], [l] or [r] for `longtabu`), unless another position is specified.
- the `tabu` / `longtabu` target width, unless another target is specified.

Any other tabular parameter is not restored.

Put \preamble {⟨*user-name*⟩} alone inside the `tabu` (or `longtabu`) preamble in place of the usual columns specifications.

\preamble works exactly as if you defined a custom environment for `tabu`.

\preamble works with longtabu .

Example (continued...):

```
\tabulinestyle{1pt off1pt}
\begin{tabu} to\linewidth{\preamble{mytabu}} \tabucline-
This       &is &tabu &package         \\ \tabucline-
\end{tabu}
```

| | | | |
|---|---|---|---|
| This | is | tabu | package |

`\tabcolsep`, rule colors etc. are not restored from \savetabu: the only `tabu` preamble is restored.

# 6 Some other features

## 6.1 Printing numbers inside tabu with numprint and siunitx

### \tabudecimal

τℵb⊂ provides a *facility* to print numbers inside columns. This facility is not implemented to replace `siunitx` `S` and `s` columns or `numprint` `n` and `N` columns or other packages that provide alignment such as `warpcol`, `dcolumn` or `rccol`. It just make easy to apply a macro you get already on each number in a column of a `tabu`.

\tabudecimal has been developped mainly because it makes possible to align numbers inside `tabu` `X` columns.

### \tabudecimal {⟨*user-macro*⟩}

\tabudecimal can be used in the preamble of a `tabu` before a column specification. The ⟨*user-macro*⟩ is a macro with one parameter that has to be defined before.

Example with **\numprint**:

```
\def\usermacro#1{\numprint[\officialeuro]{\zap@space #1 \@empty}}
\nprounddigits{2} \npprintnull \npthousandsep{\,} \npunitseparator{~}
```

```
 \rowfont [c]{\bf } January & February \\
 12.324    & 745.32  \\
 21.13     & 0       \\
 213.3245  & 12.342  \\
 2143.12 & 324.325 \\
\end {tabu}
```

| **January** | **February** | **...** |
|---:|---:|---|
| 12,32 € | 745,32 € | ... |
| 21,13 € | 0,00 € | ... |
| 213,32 € | 12,34 € | ... |
| 2 143,12 € | 324,33 € | ... |

Example with **\SI**:

```
\def\usermacro#1{\SI[group-four-digits=true,           % thousand separator
                     round-mode=places,                 % round numbers
                     round-precision=2,                 % with 2 decimal digits
                     round-integer-to-decimal=true,     % add trailing 0 if necessary
                     per-mode=symbol]{#1}{\officialeuro\per\kilo\gram}}
\begin{tabu}spread 0pt{|[GreenYellow]*2{>{\tabudecimal \usermacro}X[r]|[GreenYellow]}} ...
```

| **January** | **February** | **...** |
|---:|---:|---|
| 12.32 €/kg | 745.32 €/kg | ... |
| 21.13 €/kg | 0.00 €/kg | ... |
| 213.32 €/kg | 12.34 €/kg | ... |
| 2 143.12 €/kg | 324.33 €/kg | ... |

As you can see, the columns widths are exactly the same, whatever their content.

Here \tabulinesep has been set to 3*pt*.

### You should know how it works...

Yes you should know how it works to avoid problems. `tabu` has a small scanner based on `\futurelet` to grab all numbers, blank spaces, commas and dots + and − sign and also the letter `e` and `E` for exponants. The scanner stops as soon as something else than a number, blank space, comma, dot, +, −, `e`, `E` is found, and even if it is a macro that contains a number.

This explains why there is `\zap@space` in the definition of `\usermacro`: because the scanner scans blank spaces and because `\numprint` does not allow blank spaces in its mandatory argument, quite strangely...

## 6.2 Paragraph indentation

`tabu` takes care of paragraph indentation when it is used with `X` columns and its default target, no matter if it has been loaded or not with the `linegoal` option. Example with LaTeX default: `\parindent` = 20pt.

| This is tabu with its default target in an indented paragraph. |
|---|

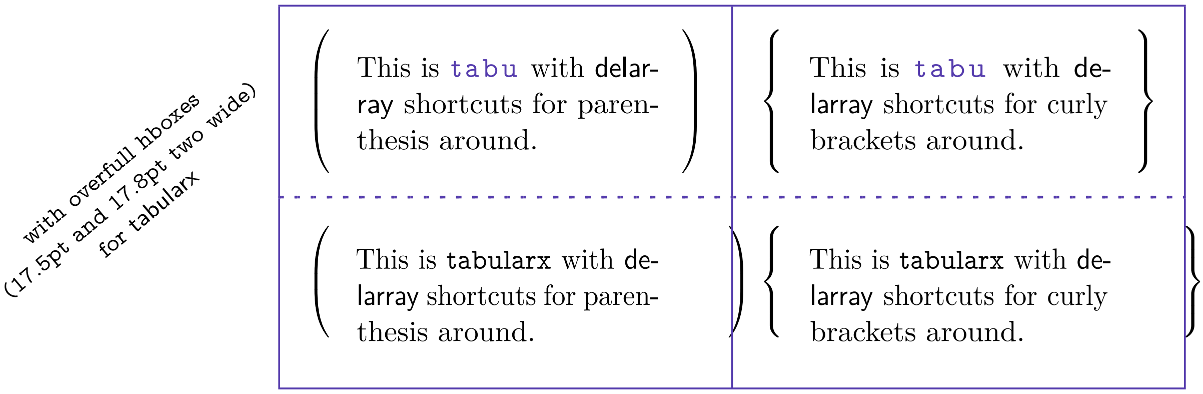
| This is tabu with its default target, preceded by `\noindent` |
|---|

| This is `tabularx` with target: `\linewidth` in an indented paragraph. |
|---|

| This is `tabularx` with target: `\linewidth`, preceded by `\noindent` |
|---|

## 6.3 delarray shortcuts

When you enclose your tabular with math delimiters using delarray shortcuts, tabu tries to reach its target for the whole: the tabular and the delimiter(s). You can see the difference:



| | |
|---|---|
| This is `tabu` with delarray shortcuts for parenthesis around. | This is `tabu` with delarray shortcuts for curly brackets around. |
| This is `tabularx` with delarray shortcuts for parenthesis around. | This is `tabularx` with delarray shortcuts for curly brackets around. |

Here `\tabulinesep` $= 3mm$

# 7 Differences between tabu, tabular, tabularx and longtable

## 7.1 Paragraph indentation

See Paragraph indentation

## 7.2 Custom environments

Unlike `tabularx`, it is possible to define your own environment using `tabu`:

```
\newenvironment{foo}
   {\begin{tabu}{X[1.2]||[1pt gray]X}}
   {\end{tabu}}
```

`tabu` environment, even when `X` columns are used, may appear in the definition of your custom `tabular` environment.

You can also use the commands `\savetabu` `\preamble` (or `\usetabu`) for this purpose.

## 7.3 Inversion of tokens

When you typeset the following `tabular`:

```
\begin{tabular}{|>{\bfseries}>{ before }l<{ one }<{ two }|}
   cell content
\end{tabular}
```

You get the following result: | before **cell content two one** |

⟶ The word *before* is not bold, and ***two*** comes before ***one***.

The reason is explained in the documentation of `array.sty`, and is related to the `array` environment in math mode when using `\newcolumntype`.

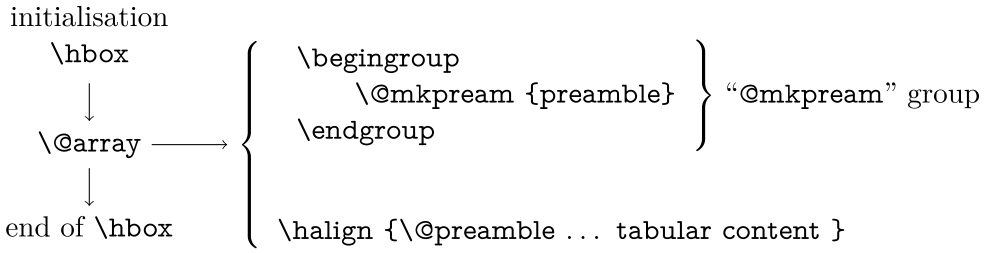
This rather strange inversion of tokens may be justified in math mode (otherwise, errors may occur) but not in text mode in our opinion. Inside a `tabu` environment, when not in math mode, the tokens are not reversed and you get the intuitively expected result:

| **before cell content one two** |

In math mode however, tokens are in the reverse order in the `tabu` environment like they are in the `array` environment.

## 7.4 Improved process for rewriting columns *(for keen readers)*

Any tabular that does not split accross pages is made with the following process:



For more details, see the Flow chart of expansion.

`\@mkpream` works in two times inside a (semi-simple) group:

*First the rewriting process:*

Each special column in the tabular preamble is transformed into one the columns defined by `array.sty`.

*Second the building of the* `\halign` *preamble:*

The "rewritten preamble" is parsed and transformed in a preamble for the TEX primitive `\halign`. The result is stored into the `\@preamble` macro.

Any special columns of `tabu` are defined only inside the "`@mkpream`" group.

In the following example, you get an error with `tabular` and no error with `tabu`. With `tabular`, and `siunitx` `S` column, the *rewriting process* is as follow:

```
\documentclass {minimal}
\usepackage {numprint,siunitx,xcolor}
\usepackage {tabu}
\begin {document}

\begin {tabular}{*2{S[color=green]}}
   123,45
\end {tabular}

\begin {tabu}{*2{S[color=green]}}
   123,45
\end {tabu}

\end {document}
```

Inside `tabular`:

1) Rewrite `S`: not found because inside `{...}`
2) Rewrite ∗
3) Rewrite `n` column defined by package `numprint`
   Then the 'n' in `green` is rewritten ⟶ problem

Inside `tabu`:

1) Rewrite ∗
2) Rewrite `|` (there is none here)

`go back`

3) Rewrite ∗
4) Rewrite `|`
5) Rewrite `S`
6) Rewrite `n` ⟶ not found because `S` was rewritten before, according to `siunitx` definition.

The process of rewriting columns is usually longer inside `tabu` than inside `tabular`, but conversely `tabu` with `X` columns is optimised compared to `tabularx`, because the preamble is built only once, and not rebuilt before each trial as `tabularx` does. Thus `tabu` is much quicker than `tabularx`.

The process of rewriting is very sensitiv to the order in which columns are actually rewritten. This becomes critical when columns are defined with an optional argument like `tabu` `X` and `|` columns or `siunitx` `S` column.

# 8 The package options

## 8.1 The debugshow package option

```
\tracingtabu
\tracingtabu  =1, 2, 3 or 4
```

The control sequence `\tracingtabu` has the same effect as the `debugshow` option:

- $\tau_{\aleph}b\subset$ will report the widths it computes at each attempt to read the target, when `X` columns are used.
- Saved informations on the `tabu` are reported in the .log file when `\savetabu` is used.

| | |
|---|---|
| `\tracingtabu` = 2 | gives more information on the measures of the natural widths. |
| `\tracingtabu` = 3 | shows the struts inserted inside the `tabu` environment and gives more information about the measures of the height and depth of every row. |
| `\tracingtabu` = 4 | displays information on the insertions made by `\tabucline`. |

Typical information in the .log file:

```
(tabu) Try      tabu X         tabu Width        Target       Coefs       Update
(tabu)   1)   386.67296pt    797.34592pt    386.67296pt    2.0pt    -205.33649pt
(tabu)   2)   181.33647pt    386.67294pt    386.67296pt    2.0pt      0.00002pt
(tabu)   2)  Target reached (hfuzz=0.1pt) ****************
```

What does it mean?

1) The first attempt was performed with `X`=386.67296pt
   The `tabu` width (797.34592pt) exceeded the target by 410.67296pt.
   Thus `X` has been updated: 410.67296pt /2 = 205.33649pt and then:
   `X` = 386.67296pt − 205.33649pt = 181.33647pt

2) The second attempt lead to a `tabu` width of 386.67294pt: the target is reached.
   The final width of each `X` column is the product of `tabu X` by its width coefficient.

## 8.2 The delarray package option

`delarray` option has the single effect to load `delarray.sty` for delimiters shortcuts around `tabu`. Delimiters shortcuts work both in math and text mode.

## 8.3 The linegoal package option

With the linegoal option, the default target for `tabu` with `X` columns is `\linegoal` instead of `\linewidth`. The linegoal package must be loaded and compilation must be done with pdfTEX, otherwise, a warning is displayed and the `linegoal` option has no effect: the default target remains `\linewidth`. `\linegoal` works with pdfTEX in `pdf` mode **and in** `dvi` mode.

If for some reason, you wish to turn down the linegoal option in your document, you can say (in a group for example): `\let\tabudefaulttarget=\linewidth`

In any case, specifying the target overwrites the default: `\begin {tabu} to\linewidth`

# 9 Corrections of some bugs *(available only inside* `tabu`*)*

## 9.1 Correction for colortbl and arydshln: compatibility with delarray

Both colortbl and arydshln forget the control sequence `\@arrayright` in their implementation, quite strangely because both of them take care of `\@arrayleft`. As a result, delarray shortcuts for delimiters around a tabular does not work if colortbl and/or arydshln are loaded.

Those control sequences are used by the delarray package to put variable size delimiters around the array:

```
\begin {tabu} \{{X}.                          \left \{\begin {tabu}{X}
  ...                              is like:     ...
\end {tabu}                                   \end {tabu} \right .
```

## 9.2 Correction for arydshln: @ columns

A bug in `\adl@xarraydashrule`: `!-arg` columns (class 1) and `@-arg` columns (class 5) should be treated the same as far as rules are concerned.

With this correction, the "known problem number 1" in arydshln documentation is solved.

# 10 To do for even better tabus

In decreasing order of priority:

- Make double `\tabucline` compatible with colortbl `\doublerulesepcolor`
- Multiple `\tabucline` between different columns: extended specs:
  `\tabucline [line spec]{start-stop, start-stop}[line spec]{start-stop}` ...
- Reimplement `\multicolumn` in order to allow the `X` token in `\multicolumn` preamble. Provide `\multicell` to allow spanning columns and rows at the same time.
- Presently, `longtabu` with `X` columns works only if `\LTchunksize` is greater than the number of rows. I compiled a `longtabu` of 56 pages on my PC with `\LTchunksize` = 2 000 without problem. Presently `\LTchunksize` is set to 10 000 during trials when `longtabu` contains `X columns`.
- Make `\tabucline` work with page breaks (one line on the top of the page, one line on the bottom of the previous).

# 11 Technical notice and Implementation

## 11.1 Drawing a tabular - The $\tau_{\aleph}b\subset$ approach

$\tau_{\aleph}b\subset$ has a different approach than almost any other package providing facilities for tabulars. colortbl and arydshln both put the cells contents into a box for measuring purpose, and then use the dimensions of each box to make their setups:

**colortbl** needs the dimensions of the box to put a rule in the background of the cell,

**arydshln** needs the dimensions to set the length of its leaders (dash lines).

This is achieved by modifying the macros defined in `array.sty` to insert columns inside the `\halign` preamble.

Instead, $\tau_{\aleph}b\subset$ proceeds as follow:

1. It first measures (if there are some negative width coefficients, or if `tabu spread` is used) the natural widths of the cells / the columns,
2. Then it always measures the height and depth of each cell / row,
3. Thereafter, the tabular is printed exactly as if `array.sty` was entitle to print it: no “extra” boxing of the cells material. The measurements have been stored and can be used to set the struts (only one per row) and the lengths of vertical leaders.
4. No macros of `array.sty` is modified at stage 3.

$\tau_{\aleph}b\subset$ material inserted in the tabular for vertical leaders, `\rowfont` etc. is put inside the special “free” tokens provided by `array.sty`:

- A vertical leader is put inside a `!` column: `!{vertical leader}`
- Changing font and alignment in one row requires some setup in `>` tokens: `>{rowfont material}`.

This way, the commands of `array.sty` that build each column definition (or preamble, in the sense of `\halign`) are never modified.

## 11.2 Algorithms

### tabu to target

The algorithm of `\tabu@arith` computes the desired widths to reach the target. In any case, only one measure of the tabular is required to get the widths for all columns. Here we describe the method with an example and some equations too to show that this handle all cases in generality.

**Notations and initialisation of `X`** In the case of `tabu to` the target $T = 300$ is given : it is the target specified by the user or the default `tabu` target which is `\linewidth` $-\langle$*parindent correction*$\rangle$ or `\linegoal`. Each `X` column has a width coefficient which is given too (or default to 1). The coefficients are: $c_1, c_2, \ldots c_n$.

$X$ is the main dimension that drives the widths of all columns with a non negative coefficient, and limit the widths of columns with a negative coefficient.

Then we have first:

| Coef $c_i$ | | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $\sum$ | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $-1$ | $-2$ | $-5$ | $-2$ | 2 | 3 | 15 | |
| Target $T$ | 300 | | | | | | | | |

Some coefficients are negative and we have to measure the natural widths of the corresponding columns, for columns always have a width:

$$\lambda_i = \begin{cases} c_i \cdot X & \text{if } c_i > 0 \\ \text{Min}\left(|c_i| \cdot X, \nu_i\right) & \text{if } c_i < 0 \quad \text{with} \quad \nu_i \leq T \quad \forall i \end{cases}$$

$\nu_i$ is the “natural width of the column” in the sense that it is the maximum of the natural widths of each cell in the *ith* `X` column, limited to the `tabu` target: $\nu_i \leq T \quad \forall i$.

The whole width of the tabular is always:

$$\mathrm{wd(table)} = \sum_i \lambda_i + \text{incompressible material} \begin{cases} \bullet\ \texttt{\textbackslash tabcolsep} \\ \bullet\ \text{vertical lines/leaders thickness} \\ \bullet\ \text{non } \texttt{X} \text{ columns} \end{cases}$$

and should finally be equal to $T$, by the correct computation of the $\lambda_i$.

So what is $X$ at first ? Columns that have a non negative coefficients always have a width equal to $\lambda_i = c_i \cdot X$ therefore, if we only have non negative coefficients, we can safely set:

$$X = \frac{T}{\sum\limits_i c_i}$$

then: $\sum_i \lambda_i = \sum_i c_i \cdot X \geq T$ at the first trial. But this is not the same if some coefficients are negative, because in this case the column width $\lambda_i$ can shrink until its natural width $\nu_i$ and may be until to $0pt$ ! And then if every column has a negative coefficient, one of them can have a width close to the target $T$. We have to ensure that the first measure of the natural widths does not limit them artificially:

$$\begin{aligned} \forall i \quad c_i < 0 &\implies |c_i| \cdot X \geq T \\ \exists\, c_i > 0 &\implies \sum_{\substack{i \\ c_i > 0}} c_i \cdot X \geq T \end{aligned}$$

And finally, for the measure: $X = \mathrm{Max}\left[\mathop{\mathrm{Max}}_{\substack{i \\ c_i<0}} \frac{T}{|c_i|} ; \frac{T}{\sum\limits_{\substack{i \\ c_i>0}} c_i}\right]$

| Coef $c_i$ | | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $\sum$ | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $-1$ | $-2$ | $-5$ | $-2$ | 2 | 3 | 15 | |
| Target $T$ | 300 | | | | | | | | |
| $X$ | 300 | | | | | | | | |
| $\nu_i$ | | 10 | 300 | 80 | 80 | | | | |
| $\lambda_i$ | | 10 | 300 | 80 | 80 | 600 | 900 | 1970 | 1800 |

**First step of the algorithm: reducing the width** After having measured the table we get: $\mathrm{wd(table)} = 2100$. The *incompressible material* is $2100 - 1970 = 130$ wide and the gap to the target is $\Delta = 2100 - 300 = 1800$.

We now choose a new value for $X$:

$$\sum_i \lambda_i = \sum_i \mathop{\mathrm{Min}}_{\substack{i \\ c_i<0}} (\nu_i ; c_i \cdot X) + \sum_{\substack{i \\ c_i>0}} c_i \cdot X \leq \sum_i |c_i| \cdot X$$

Let's try $X' = \frac{\sum\limits_i \lambda_i - \Delta}{\sum\limits_i |c_i|}$ so that $\sum\limits_i \lambda'_i \leq \sum\limits_i |c_i| \cdot X' \leq \sum\limits_i \lambda_i - \Delta$ :

| Coef $c_i$ | | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $\sum$ | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $-1$ | $-2$ | $-5$ | $-2$ | 2 | 3 | 15 | |
| Target $T$ | 300 | | | | | | | | |
| $X$ | 300 | | | | | | | | |
| $\nu_i$ | | 10 | 300 | 80 | 80 | | | | |
| $\lambda_i$ | | 10 | 300 | 80 | 80 | 600 | 900 | 1970 | 1800 |
| $X'$ | 11.33 | | | | | | | | |

$X' = \frac{1970 - 1800}{15} = \frac{170}{15} = 11.33$ Note that the computation of $X'$ does not involve any measurement.

| Coef $c_i$ | | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $\sum$ | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $-1$ | $-2$ | $-5$ | $-2$ | 2 | 3 | 15 | |
| Target $T$ | 300 | | | | | | | | |
| $X$ | 300 | | | | | | | | |
| $\nu_i$ | | 10 | 300 | 80 | 80 | | | | |
| $\lambda_i$ | | 10 | 300 | 80 | 80 | 600 | 900 | 1970 | 1800 |
| $X'$ | 11.33 | | | | | | | | |
| $\lambda'_i$ | | $10,00$ | $22,67$ | $56,67$ | $22,67$ | $22,67$ | $34,00$ | $168,67$ | $-1.33$ |

Here we are in the case where the table width:

$$\begin{aligned} \text{wd(table)} &= \sum_i \lambda_i + \text{incompressible material} = T + \Delta \\ \implies & \sum_i \lambda'_i + \text{incompressible material} \leq \sum_i \lambda_i - \Delta + I = T \end{aligned}$$

Without any measure, we can say that the final table width will be less than the target, if we choose $X'$. The free space to share among the `X` columns (computed with $X'$) is now $\Delta' = T - (\sum_i \lambda'_i + I) = 300 - (168.67 + 130) = -1.33$, where $I$ is the incompressible material.

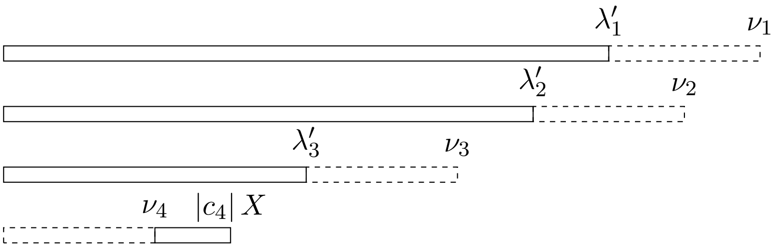
**Giving space** We say that a column is *saturated* (*ie.*full) if its natural width is greater than $|c_i| \cdot X$, or all the same that $\lambda_i < \nu_i$. We also will consider that the columns with $c_i > 0$ have a "natural width" which is always equal to $c_i \cdot X$: in other words, a column with a non negative coefficient is always *saturated.*

Giving space (or "refunding" space) to the columns must be done in priority to the *saturated* columns. If all columns are finally underfull, then we will distribute the extra space to each, according a distribution rule. But this case can only occur if $\forall i \quad c_i < 0$ because we first choosed $X$ so that:

$$X \geq \frac{T}{\sum\limits_{\substack{i \\ c_i > 0}} c_i}$$

and hence, the sum of the widths of the "non negative" columns exceeds the target.

Let's rank the columns widths:



we first give space
to the saturated columns
1, 2 and 3

Because of the saturation, the total amount of space to give: $|\Delta|$ shall be shared among the columns according to their widths coefficients. We shall not give too much space: the columns shall remain saturated. Let $0 < \epsilon \leq |\Delta|'$ the amount of space to give, then after the operation:

$$\begin{aligned} \lambda_1" + \lambda_2" + \lambda_3" &= \lambda'_1 + \lambda'_2 + \lambda'_3 + \epsilon \\ &= |c_1|\, X' + |c_2|\, X' + |c_3|\, X' + \epsilon \end{aligned}$$

Let's say $X" = X' + \dfrac{|\Delta|'}{\sum\limits_{\substack{i \\ c_i \text{saturated}}} |c_i|}$ then it's possible, without any measure, to compute:

$$\sum_{\substack{i \\ c_i \text{saturated}}} \lambda"_i + \nu_4 \leq \sum_{\substack{i \\ c_i \text{saturated}}} |c_i| \cdot X" + \nu_4 \leq \sum_i |c_i|\, X' + \Delta' = \sum_i \lambda'_i + \Delta' \leq T - I$$

Or for clarity: $\sum_i \lambda_i" + I = \mathrm{wd(table)} \leq T$ and the new free space to share is now :

$$\Delta" = \left| T - \left( \sum_i \lambda"_i + I \right) \right|$$

At each step of the computation, and without any measure but the first, $X$ grows, $\Delta$ decreases, and finally the target is reached for $X$ such that $\Delta \leq$ hfuzz.

| Coef $c_i$ | | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $\sum$ | $\Delta$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $-1$ | $-2$ | $-5$ | $-2$ | $2$ | $3$ | $15$ | |
| Target $T$ | 300 | | | | | | | | |
| $X$ | 300 | | | | | | | | |
| $\nu_i$ | | 10 | 300 | 80 | 80 | | | | |
| $\lambda_i$ | | 10 | 300 | 80 | 80 | 600 | 900 | 1970 | 1800 |
| $X'$ | 11.33 | | | | | | | | |
| $\lambda'_i$ | | $10,00$ | $22,67$ | $56,67$ | $22,67$ | $22,67$ | $34,00$ | $168,67$ | $-1.33$ |
| $X"$ | 11.43 | | | | | | | | |
| $\lambda"_i$ | | $10,00$ | $22,86$ | $57,14$ | $22,86$ | $22,86$ | $34,29$ | $170,00$ | 0 |

Now if the width of the table is less that the target, because 1) every column has a negative coefficient and 2) their natural widths are so small than the tabular don't fill the wanted horizontal space, the algorithm artificially raise the natural widths, according to a linear distribution:

$$\lambda'_i = \lambda_i + \Delta \cdot \frac{\lambda_i}{\sum_i \lambda_i} = \nu_i + \Delta \cdot \frac{\nu_i}{\sum_i \nu_i} = \nu_i \cdot \left( 1 + \frac{\Delta}{\sum_i \nu_i} \right)$$

### `tabu spread` dimen

The case of `tabu spread` is interesting and quite complex...

Here, the aim of the game is to give a target to the table, depending on its natural width. `tabu` has a default target (`\linewidth` in general, but it is possible to `\let \tabudefaulttarget` to another value... for example `\linegoal`) which is a maximum for the final target of `tabu spread`. The case where the spread is $0pt$ is not simpler nor more difficult.

If every column has a negative coefficient, it's rather easy because either the table exceeds the target, and then the new target will be the default target (the *maximum*), or the table width is less than the default target and we fix the new target to be that width + the spread, in the limit of the default target.

The condition that must hold on coefficient is not restritive if every column has a negative coefficient because if you say, for example: $X = \operatorname{Max}_i \frac{\nu_i}{|c_i|}$ then:

$$\sum_i \lambda_i = \sum_i \operatorname{Min}\left(\nu_i ; |c_i| \cdot X\right)$$

is true. It's always possible to find a $X$ such that the behaviour anounced in the documentation is observed !

Then let's get some non negative coefficients. The natural widths of such columns must be measured, but the natural width of the tabular is not the same, for the proportions between column widths – expressed by their positive coefficient $c_i$ – must be respected.

The real natural width of the tabular, which observe the proportions between columns with a non negative coefficient is:

$$\mathrm{wd(table)} + \operatorname*{Max}_{\substack{i \\ c_i>0}} \left( \frac{\nu_i}{c_i} \right) \times \sum_{\substack{i \\ c_i>0}} c_i - \sum_{\substack{i \\ c_i>0}} \nu_i > \mathrm{wd(table)}$$

This quantity is computed, Tabu adds the `spread` and fix the new target to the sum, in the limit of the default target.

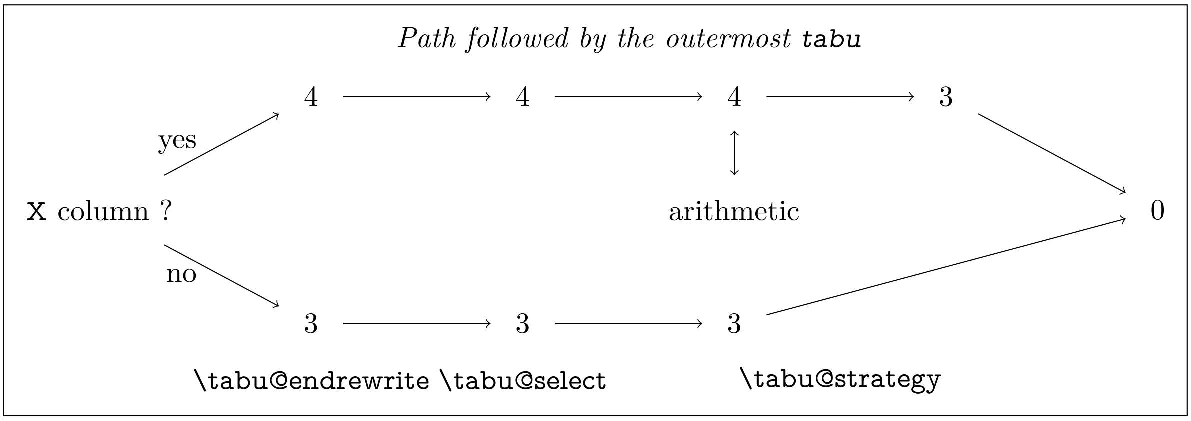
Then $X$ is initialized such that: $X = \mathrm{Max}\left[\mathop{\mathrm{Max}}_{\substack{i\\ c_i<0}} \frac{T}{|c_i|};\ \frac{T}{\sum\limits_{\substack{i\\ c_i>0}} c_i}\right]$

and the algorithm described in the former section works, without any new measurement of the tabular.

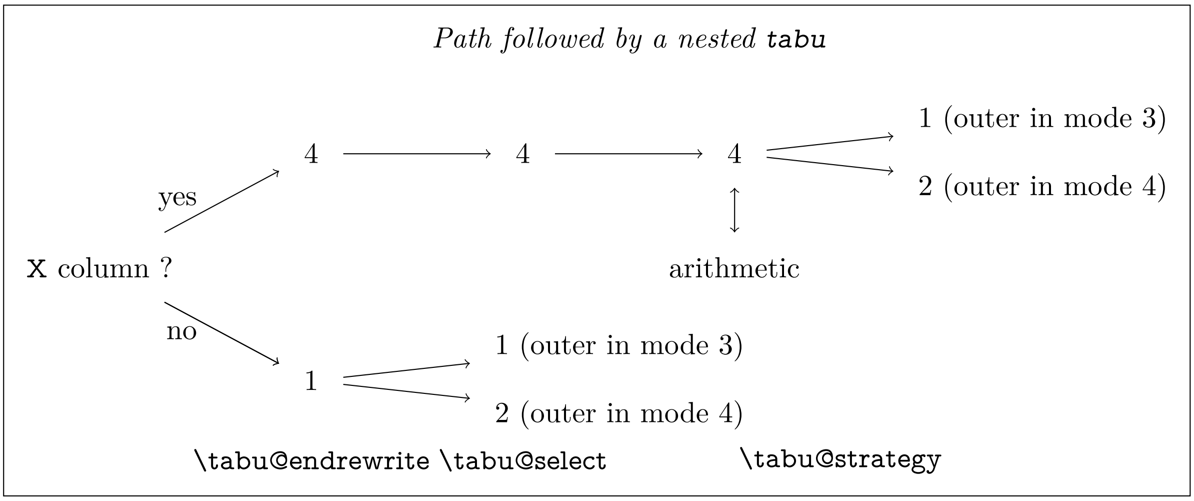
Unless this was not possible or deemed inconvenient for clarity, the code is presented in the same order it executes.

## 11.3 The tabu strategies

| | Not nested (outer) | | Nested | |
|---|---|---|---|---|
| | `\count@` | condition | `\count@` | condition |
| `\tabu@endrewrite` | | | 0 | outer is in mode 0 |
| | 3 | no X column | 1 | no X column |
| | 4 | X columns | 3 | X column |
| `\tabu@select` | | | 0 | outer in mode 0 ⇒ print |
| | | | 1 | outer in mode 3 |
| | | | 2 | from 1 in `\tabu@endrewrite` if outer in mode 4 |
| | 3 or 4 | needs trials | 3 or 4 | needs trials |
| `\tabu@strategy` | 0 | print out | | |
| | | | 1 | Exit in vertical measure (outer in mode 3) |
| | | | 2 | Exit with a rule (outer in mode 4) |
| | 3 | Vertical measure | | |
| | 4 | Horizontal measure | 4 | Horizontal measure (nested in coef< 0 or spread) |



*Path followed by the outermost* `tabu`



*Path followed by a nested* `tabu`

## 11.4 Identification and Requirements

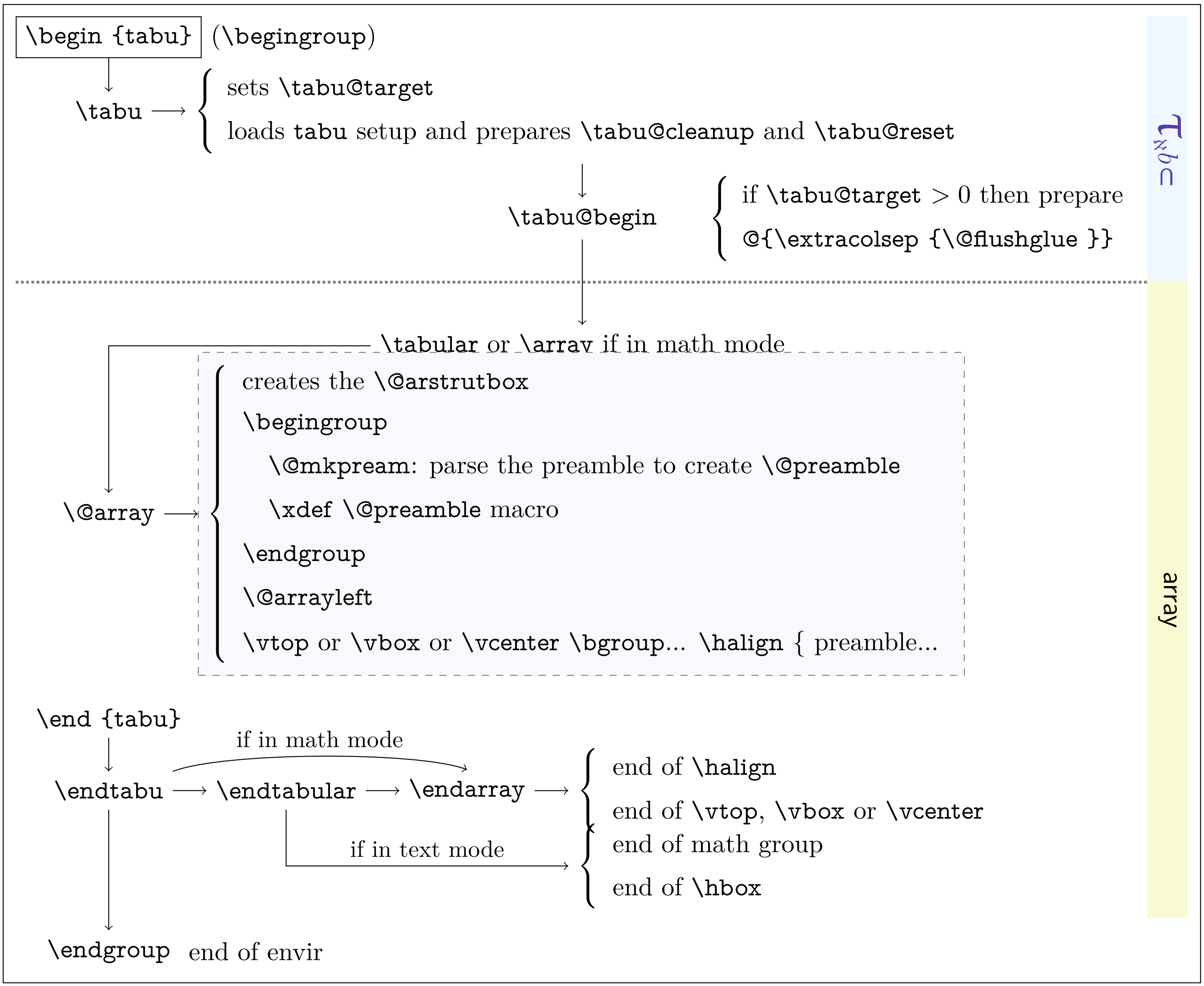
$\tau_{\aleph}b\subset$ requires **array.sty** and **varwidth.sty**. The package namespace is **tabu@**.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}[2005/12/01]
\ProvidesPackage{tabu}[2011/02/26 v2.8 - flexible LaTeX tabulars (FC)]
\RequirePackage{array}[2008/09/09]
\RequirePackage{varwidth}[2009/03/30]
```

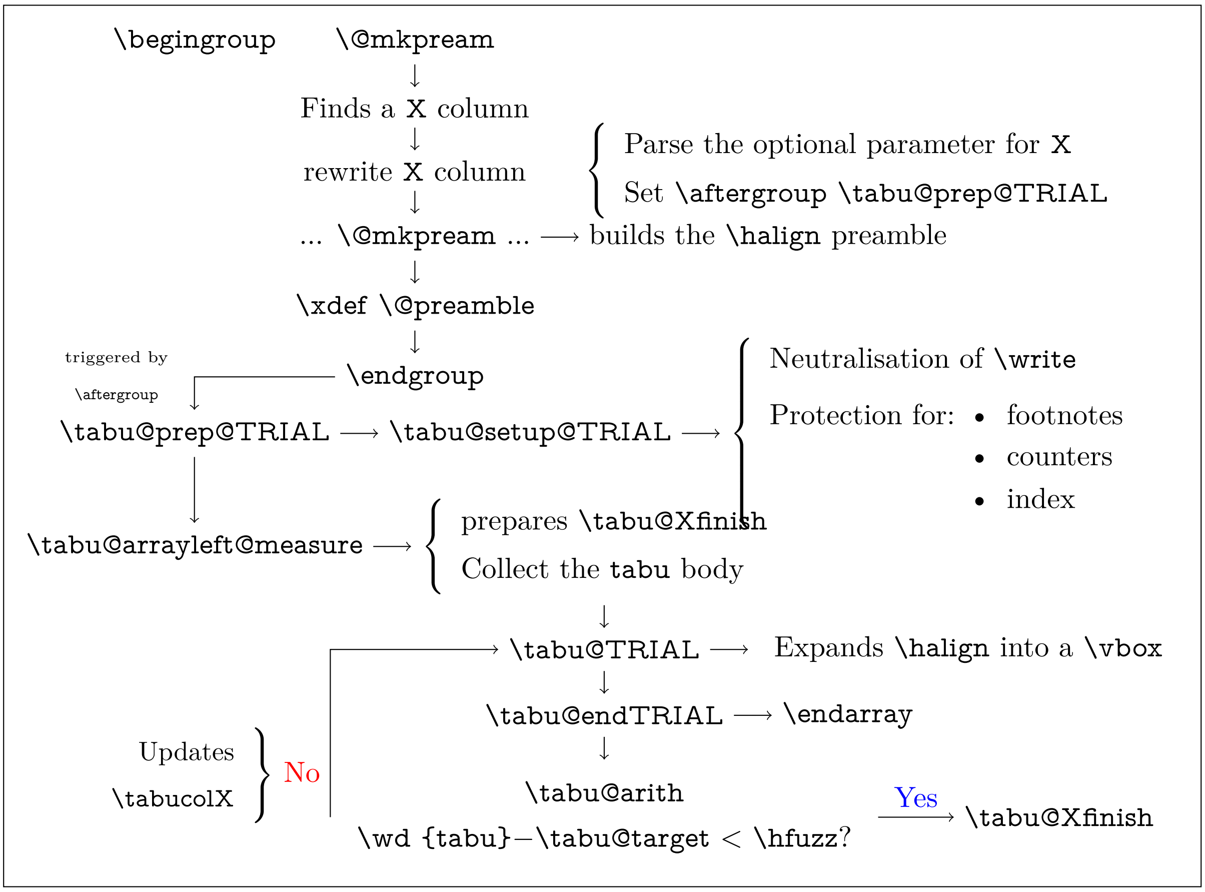
Minimal catcode acertaining for loading $\tau_{\aleph}b\subset$ in good conditions:

```
\AtEndOfPackage{\tabu@AtEnd \let\tabu@AtEnd \@undefined}
\let\tabu@AtEnd\@empty
\def\TMP@EnsureCode#1={%
    \edef\tabu@AtEnd{\tabu@AtEnd
                     \catcode#1 \the\catcode#1}%
    \catcode#1=%
}% \TMP@EnsureCode
\TMP@EnsureCode 33 = 12 % !
\TMP@EnsureCode 58 = 12 % : (for siunitx)
\TMP@EnsureCode124 = 12 % |
\TMP@EnsureCode 36 =  3 % $ = math shift
\TMP@EnsureCode 38 =  4 % & = tab alignmment character
\TMP@EnsureCode 32 = 10 % space
\TMP@EnsureCode 94 =  7 % ^
\TMP@EnsureCode 95 =  8 % _
```

## 11.5 Flow chart of expansion


\begin {tabu} (\begingroup)
\tabu
sets \tabu@target
loads tabu setup and prepares \tabu@cleanup and \tabu@reset
\tabu@begin
if \tabu@target > 0 then prepare
@{\extracolsep {\@flushglue }}
\tabular or \array if in math mode
\@array
creates the \@arstrutbox
\begingroup
\@mkpream: parse the preamble to create \@preamble
\xdef \@preamble macro
\endgroup
\@arrayleft
\vtop or \vbox or \vcenter \bgroup... \halign { preamble...
\end {tabu}
if in math mode
\endtabu
\endtabular
\endarray
end of \halign
end of \vtop, \vbox or \vcenter
end of math group
end of \hbox
if in text mode
\endgroup end of envir
$\tau_{\aleph}b\subset$
array

**tabu to with X column** The important part of the job is made inside the dashed box above: `\@mkpream` expands the columns definitions, which can be user defined. Hopefully, it does its job inside a group, therefore a user-column can set a macro to be expanded `\aftergroup`. This implementation allows much modifications in the tabular preparation, without any change in the macros of `array.sty`.



## tabu spread with X column

In the case of "`tabu spread`" with `X` columns, the process is the same as the one described for "`tabu to`" with `X` columns. However, the first trial is different because we have first to measure the *natural width* of the tabular. The process is the following:

- `\tabu@target` is first set to `\linewidth` (or `\linegoal` with the `linegoal` package option).
- The `X` column corresponds to a `\vbox` with `\hsize` fixed to `\tabu@target`.
- Inside this `\vbox` the cell content is written into a `\hbox` whose width is limited to `\tabu@target`. This `\hbox` is captured into the box register `\tabu@box`.
- At the end of the cell, the `\badness` of the `\hbox` is checked:
  - if the `\badness` is > 1000 then the text is too long and "`tabu spread`" is useless: `tabu to \tabu@target` give the same result.
  - Otherwise, we get the natural width of the cell content by:
    `\setbox \tabu@box \hbox {\unhbox \tabu @box}`
- At the end of the first trial, `\tabu@spreadarith` checks if:

$$\text{width(tabular)} + \text{spread} < \texttt{\textbackslash linewidth}\ (\text{or } \texttt{\textbackslash linegoal})$$

  - if not, then `tabu to\tabu@target` give the same result
  - Otherwise, the target for `tabu to` will be:

$$\text{width(tabular)} + \text{spread} - \sum_i \text{natural widths } X_i + \underbrace{\text{Max}_i\left(\frac{\text{natural width } X_i}{\text{coef}_i}\right) \times \sum_i \text{coef}_i}_{\substack{\text{minimal natural width that can be obtained}\\ \text{with the given coefs}}}$$

And the next trial will be done as if the user called "`tabu to`" with this target.

## 11.6 Some constants

Here we define the constants used by Tℵb⊂: TeX registers and a few *helper* macros.

When working inside a tabular (*ie.*`\halign`) each cell is a TeX group. Probably the most important property of each register defined here is whether it is global or not. A *local* register does not suffer, never, any global assignment.

### TeX registers

**taburow** LaTeX counter that globally stores the value of the current row. It is updated at `\everyrow`, rather than at `\everycr`[8]. `\thetaburow` expands to the (arabic) number.

This counter can be read by the user, but she **must not change its value** because it is used internally to store the height/depth of every row, for vertical spacing adjustment (and vertical leaders).

**\tabu@nbcols** TeX counter that – locally – saves the total number of columns of the `tabu`. Special `@` and `!` columns are not counted (they are not *real* columns for `\halign`, but only insertions into the preamble).

The value is used by `\tabucline` to ensure that the leader does not jut out over the last column...

**\tabu@cnt** TeX counter that – locally – stores the number of trials. Incidentally, it is also temporarily used to parse the width coefficient for `X` columns, during the rewriting process.

**\tabu@Xcol** TeX counter that – locally – stores the number of `tabu` `X` columns. Defined while rewriting the `X` token, it is used in the specification of the width of the column (`\tabu@hsize {Rank of the X column}{coef}`).

It is also used to store the natural width of `X` columns (in the cases of a negativ coefficient or if `tabu spread` is used).

**\tabu@alloc**
**\tabu@nested** A global counter whose initial value $(-1)$ is incremented for each nested tabular. The end of the outermost tabular globally resets the value to $-1$. `\tabu@nested` stores locally the value of `\tabu@alloc` and is therefore the “index” of the current tabular (the one that is actually in construction).

This influences the initialisation process (cf. `\tabu@setup` and `\tabu@init`).

**\tabu@start**
**\tabu@stop** They are used locally by `\tabucline` and `\everyrow` while parsing the parameters: this is, for clarity, the local name for `\@tempcnta` and `\@tempcntb`.

```
\newcount  \c@taburow        \def\thetaburow {\number\c@taburow}
\newcount  \tabu@nbcols
\newcount  \tabu@cnt
\newcount  \tabu@Xcol
\let\tabu@start  \@tempcnta
\let\tabu@stop   \@tempcntb
\newcount  \tabu@alloc  \tabu@alloc=\m@ne
\newcount  \tabu@nested
\def\tabu@alloc@{\global\advance\tabu@alloc \@ne \tabu@nested\tabu@alloc}
```

**\tabu@target** TeX dimen that – locally – stores the `tabu` target (either “`to`” or “`spread`”).

**\tabu@spreadtarget** TeX dimen that – locally – stores the `tabu` spread given by the user.

**\tabu@naturalX** TeX dimen that – globally – stores the total natural widths of the `X` columns, in the cases of negativ coefficients and/or `tabu spread`. The value is reset to *0pt* at `\everyrow`, and *maxima/minima* are stored into the macro `\tabu@naturalXmin` and `\tabu@naturalXmax`: those are required for the algorithm of `tabu spread` (`\tabu@spreadarith`).

**\tabucolX** TeX dimen that – locally – stores the width corresponding to the preamble token `X`[1]: the standard width of `X` columns.

**\tabu@DELTA** This is for clarity, the local name of `\@tempdimc` in `\tabu@arith`.

**\tabu@thick**
**\tabu@on**
**\tabu@off** They are used locally by `\tabu@getline`, while parsing the parameters for a line specification. This is for clarity, the local name for `\@tempdima`, `\@tempdimb` and `\@tempdimc`.

---

[8] Package xcolor defines the `\rownum` TeX counter, which is globally updated at `\everycr`. Hence this `\rownum` counter is not reliable in case the user invokes `\cline` or `\cmidrule` for example...

```
\newdimen \tabu@target
\newdimen \tabu@spreadtarget
\newdimen \tabu@naturalX
\newdimen \tabucolX
\let\tabu@DELTA \@tempdimc
\let\tabu@thick \@tempdima
\let\tabu@on     \@tempdimb
\let\tabu@off    \@tempdimc
```

**\tabu@Xsum** TeX dimen that – locally – stores the sum of all width coefficients for `X` columns. This is required to fix the initial value for `\tabucolX` and then in the algorithms (`\tabu@arith` and `\tabu@arithnegcoef`).

```
\newdimen \tabu@Xsum
```

**\extrarowdepth** array.sty defines `\extrarowheight` as a TeX dimen register: the extra height to be finally added to each row of a table. τℵb⊂ defines `\extrarowdepth` in addition: the *extra depth*. Though `\extrarowheight` and `\extrarowdepth` can be set by the user, the official interface is `\extrarowsep`.

**\abovetabulinesep**
**\belowtabulinesep** TeX dimensions `\abovetabulinesep` and `\belowtabulinesep` store the minimum allowed vertical space between the contents of the cells and their borders. Their values are ignored if non positive. Though they can be set by the user, the official interface is `\tabulinesep`.

The philosophy and the technics are similar to the one provided by the cellspace package. However, limitations of cellspace are lifted (nested `tabu` environments, use of colors... see the cellspace limitations in the revision history). τℵb⊂ inserts only one strut per line, whose name is `\@arstrut`.

**\tabustrutrule** The TeX dimen `\tabustrutrule` is here only for debugging purpose: its value must be $0pt$. It behaves mostly like TeX primitive `\overfullrule`, and allow to see the struts introduced in the tabular, and to control vertical spacing. Setting `\tabustrutrule` to a positive value has no effect unless `\tracingtabu` is $\geq 3$. The official interface is `\tracingtabu` $= 3$.

```
\newdimen \extrarowdepth
\newdimen \abovetabulinesep
\newdimen \belowtabulinesep
\newdimen \tabustrutrule      \tabustrutrule \z@
```

**\tabu@thebody** This token stores – locally – the collected content of the `tabu` environment during the measuring process.

**\tabu@footnotes** Token that globally stores the footnotes inside the `tabu` environment, for `\insert` does not work inside such a level of groupings...

```
\newtoks \tabu@thebody
\newtoks \tabu@footnotes
```

**\tabu@box** Stores – loally – the whole `tabu` when an attempt to adjust `X` columns is performed.

**\tabu@arstrutbox** While the `\@arstrutbox` may redefined globally at the end of each line (for vertical spacing adjustment), we define a new box and `\let \@arstrutbox` to be that box inside the `tabu` environment.

Hence, the `\@arstrutbox` used by other tabular environment does not suffer any modification.

**\tabu@hleads**
**\tabu@vleads** Those boxes are used to built horizontal and vertical leaders. In order not to rebuilt the boxes every time a leader is inserted, the box is globally defined if a line style is specified (*via* `|[line style]` or `\tabucline [line style]{...}` or `\tabulinestyle {line style}`.

```
\newsavebox \tabu@box
\newsavebox \tabu@arstrutbox
\newsavebox \tabu@hleads
\newsavebox \tabu@vleads
```

## Switches

**`\iftabu@colortbl`** The global switch `\iftabu@colortbl` is used by `\rowfont` when modifying the alignments, because colortbl changes the glues put inside the `\halign` preamble to make standard alignments. This switch is set At Begin Document.

**`\iftabu@siunitx`** Global switch set `\AtBeginDocument`. `true` if siunitx package is detected.

**`\iftabu@measuring`** This switch is somewhat *magic* in the sense that it has several meanings... It is temporarily set to `true` by `\tabu@arith` in the trial group, to say that the `tabu` did not reach its target yet. It is also set to `true` in the `\@mkpream` group when the first `X` column is encountered in the preamble. Finally, it is true in the trial**S** group when the outermost tabular is in strategy number 2 or number 3.

**`\iftabu@spread`** A switch whether "`tabu spread`" is used or not. A nested `tabu` inside a `X` column whose coefficient is negative has a default target set to `spread 0pt`.

**`\iftabu@negcoef`** A switch set to true in case of negativ coef (natural width if less than `X[coef]`).

**`\iftabu@everyrow`** A very important global switch: `true` when outside any `tabu` environment, `true` as well when inside a cell of a `tabu`, but globally set to `false` at `\everycr` and therefore inside any `\noalign` command. This allows to insert leaders (by `\omit \span \omit \cr \noalign {...}`) or first/last line corrections only once, even if `\everycr` is executed more than once.

**`\iftabu@long`** Finally the swith `\iftabu@long` is set to `true` inside `longtabu` and to false inside `tabu`. This is convenient because some setup are slightly different between `tabu` and `longtabu`.

```
\newif \iftabu@colortbl
\newif \iftabu@siunitx
\newif \iftabu@measuring
\newif \iftabu@spread
\newif \iftabu@negcoef
\newif \iftabu@everyrow
\def\tabu@everyrowtrue {\global\let\iftabu@everyrow \iftrue}
\def\tabu@everyrowfalse{\global\let\iftabu@everyrow \iffalse}
\newif \iftabu@long
```

**`\iftabuscantokens`**
**`\tabu@rescan`** `\iftabuscantokens` is the switch for whether or not `tabu` will use `\scantokens`. Though the user can set `\iftabuscantokens` to `\iftrue` or `\iffalse`, the official interface is `tabu*` .

It does not make sense to use `\scantokens` in a nested `tabu`: only the outermost `tabu` can use `\scantokens`, for the environment body must be collected with care !

`\tabu@rescan` is the helper macro for scanning tokens.

```
\newif \iftabuscantokens
\def\tabu@rescan {\tabu@verbatim \scantokens  }
```

## Some helper macros

**`\tabu@gobblespace`** Two macros which are needed when scanning tokens with `\futurelet`.

**`\tabu@gobbletoken`**
**`\tabu@gobbleX`** This gobbles the character number 10 in ASCII (`^^J` in TEX).

**`\tabu@ifenvir`** Checks if the current environment is `tabu` or `longtabu` (for `\multicolumn` inside `tabu`).

**`\tabu@modulo`** Computes the modulo (for `\taburowcolors`). The method is taken from H.O. intcalc package.

```
\def\tabu@gobblespace #1  {#1}
\def\tabu@gobbletoken #1#2{#1}
\def\tabu@gobbleX{\futurelet\@let@token \tabu@gobblex}
\def\tabu@gobblex{\if ^^J\noexpand\@let@token \expandafter\@gobble
                  \else\ifx \@sptoken\@let@token
                    \expandafter\tabu@gobblespace\expandafter\tabu@gobbleX
                  \fi\fi
}% \tabu@gobblex
\def\tabu@X{^^J}
```

```
{\obeyspaces
\global\let\tabu@spxiii= % saves an active space (for \ifx)
\gdef\tabu@@spxiii{ }}
\def\tabu@ifenvir {% only for \multicolumn
    \expandafter\tabu@if@nvir\csname\@currenvir\endcsname
}% \tabu@ifenvir
\def\tabu@if@nvir #1{\csname @\ifx\tabu#1first\else
                              \ifx\longtabu#1first\else
                              second\fi\fi oftwo\endcsname
}% \tabu@ifenvir
\def\tabu@modulo #1#2{\numexpr\ifnum\numexpr#1=\z@ 0\else #1-(#1-(#2-1)/2)/(#2)*(#2)\fi}
```

**\tabu@strtrim** Trimming spaces at low cost...

```
{\catcode`\&=3
\gdef\tabu@strtrim  #1{% #1 = control sequence to trim
    \ifodd 1\ifx #1\@empty \else \ifx #1\space \else 0\fi \fi
            \let\tabu@c@l@r \@empty       \let#1\@empty
    \else   \expandafter  \tabu@trimspaces  #1&#1\@nnil
    \fi
}% \tabu@strtrim
\gdef\tabu@trimspaces #1&#2\@nnil{\let\tabu@c@l@r=#2\tabu@firstspace .#1& &#2}%
\gdef\tabu@firstspace #1#2#3 &{\tabu@lastspace #2#3&}
\gdef\tabu@lastspace #1&#2&#3{\def #3{#1}%
    \ifx #3\tabu@c@l@r \def\tabu@c@l@r{\protect\color{#1}}\expandafter\remove@to@nnil \fi
    \tabu@trimspaces #1&#3\@nnil}
}% \catcode
```

**\tabu@sanitizearg** Sanitize an argument (babel compliant).

```
\def\tabu@sanitizearg #1#2{{%
    \csname \ifcsname if@safe@actives\endcsname         % <babel>
                      @safe@activestrue\else
                      relax\fi       \endcsname
    \edef#2{#1}\tabu@strtrim#2\@onelevel@sanitize#2%
    \expandafter}\expandafter\def\expandafter#2\expandafter{#2}%
}% \tabu@sanitizearg
```

**\tabu@textbar** The character `|` may have a special category code inside the document, depending on the language setting or for example, `|` can be the delimiter shortcut for verbatim. We use `\scantokens` to allow an `\ifx` test even if the category code of `|` changes along the compilation.

```
\def\tabu@textbar #1{\begingroup \endlinechar\m@ne \scantokens{\def\:{|}}%
    \expandafter\endgroup \expandafter#1\:% !!! semi simple group !!!
}% \tabu@textbar
```

**\tabu@everyrow@bgroup**
**\tabu@everyrow@egroup**

Commands like \everyrow, \taburulecolor, \tabulinestyle, \taburowcolors can be expanded either in a cell or outside a `tabu` environment or at the end of a row, inside a `\noalign` group.

To avoid the insertion of an empty math atom (equivalent to `\hbox` to0*pt*{}) we open a semi-simple group rather than a math group if not in `\noalign`. `\toks@` is used to define the local-to-the-TEX-group setting (post-fixed by `@L`).

```
\def\tabu@everyrow@bgroup{\iftabu@everyrow \begingroup \else \noalign{\ifnum0=`}\fi \fi}
\def\tabu@everyrow@egroup{%
    \iftabu@everyrow \expandafter \endgroup \the\toks@
    \else            \ifnum0=`{\fi}%
    \fi
}% \tabu@everyrow@egroup
```

## Rebuild the `\@arstrutbox`

**\tabu@arstrut**
**\tabu@rearstrut**

The macros rebuilds the `\@arstrutbox` (a `\hbox`). With the *debug* variants when `\tracingtabu` = 3 and `\tabustrutrule` > 0.

```
\def\tabu@arstrut {\global\setbox\@arstrutbox \hbox{\vrule
    height \arraystretch \dimexpr\ht\strutbox+\extrarowheight
    depth  \arraystretch \dimexpr\dp\strutbox+\extrarowdepth
    width  \z@}%
}% \tabu@arstrut
\def\tabu@rearstrut {%
    \@tempdima \arraystretch\dimexpr\ht\strutbox+\extrarowheight \relax
    \@tempdimb \arraystretch\dimexpr\dp\strutbox+\extrarowdepth  \relax
    \ifodd 1\ifdim \ht\@arstrutbox=\@tempdima
            \ifdim \dp\@arstrutbox=\@tempdimb 0 \fi\fi
        \tabu@mkarstrut
    \fi
}% \tabu@rearstrut
```

**\tabu@DBG@arstrut** This is the “debug” version of `\tabu@arstrut`: used when `\tracingtabu` = 3 or more to show the struts inserted in the tabular.

```
\def\tabu@@DBG #1{\ifdim\tabustrutrule>\z@ \color{#1}\fi}
\def\tabu@DBG@arstrut {\global\setbox\@arstrutbox
    \hbox to\z@{\hbox to\z@{\hss
    {\tabu@DBG{cyan}\vrule
    height \arraystretch \dimexpr\ht\strutbox+\extrarowheight
    depth  \z@
    width  \tabustrutrule}\kern-\tabustrutrule
    {\tabu@DBG{pink}\vrule
    height \z@
    depth  \arraystretch \dimexpr\dp\strutbox+\extrarowdepth
    width \tabustrutrule}}}%
}% \tabu@DBG@arstrut
```

**\tabu@save@decl** No inversion on tokens in the **tabu** preamble, when not in math mode.

```
\def\tabu@save@decl{\toks\count@ \expandafter{\the\toks\expandafter\count@
                                                              \@nextchar}}%
\def\tabu@savedecl{\ifcat$\d@llarend\else
       \let\save@decl \tabu@save@decl \fi % no inversion of tokens in text mode
}% \tabu@savedecl
```

**\tabu@finalstrut**

```
\def\tabu@finalstrut #1{\unskip\ifhmode\nobreak\fi\vrule height\z@ depth\z@ width\z@}
```

## Disable some commands during trials

**\tabuDisableCommands** Following the model of hyperref `\pdfstringdefDisableCommands`, `\tabuDisableCommands` allow the user to change the definition of some commands during the trial loops, by the mean of a hook to be expanded by `\tabu@setstrategy`.

```
\newcommand*\tabuDisableCommands {\g@addto@macro\tabu@trialh@@k }
\let\tabu@trialh@@k \@empty
```

**\tabu@nowrite** A trick (from the TEX-book) to forbidd `\write` when a trial is done on the `\halign`.

**\tabu@noxfootnotes** Disable footnotes during trials.

```
\def\tabu@nowrite #1#{{\afterassignment}\toks@}
\let\tabu@write\write
\let\tabu@immediate\immediate
\def\tabu@WRITE{\begingroup
   \def\immediate\write{\aftergroup\endgroup
```

```
                  \tabu@immediate\tabu@write}%
}% \tabu@WRITE
\expandafter\def\expandafter\tabu@GenericError\expandafter{%
                      \expandafter\tabu@WRITE\GenericError}
\def\tabu@warn{\tabu@WRITE\PackageWarning{tabu}}
\def\tabu@noxfootnote [#1]{\@gobble}
```

**\tabu@nocolor**
**\tabu@norowcolor**

For optimisation purpose, color changes are deactivated during trials, for they do not affect the measures.

```
\def\tabu@nocolor #1#{\@gobble}
\newcommand*\tabu@norowcolor[2][]{}
```

### siunitx S and s columns management

**\tabu@maybesiunitx**

A macro that encloses the definition of \tabu@celllalign, in order to check if the column is a siunitx S (or s) column, and neutralise the setup of \rowfont in this case, for siunitx provides its own key=value options to set fonts inside S (or s) columns.

```
\def\tabu@maybesiunitx #1{\def\tabu@temp{#1}%
                          \futurelet\@let@token \tabu@m@ybesiunitx}
\def\tabu@m@ybesiunitx #1{\def\tabu@m@ybesiunitx {%
    \ifx #1\@let@token \let\tabu@cellleft \@empty \let\tabu@cellright \@empty \fi
    \tabu@temp}% \tabu@m@ybesiunitx
}\expandafter\tabu@m@ybesiunitx \csname siunitx_table_collect_begin:Nn\endcsname
\def\tabu@celllalign@def #1{\def\tabu@celllalign{\tabu@maybesiunitx{#1}}}%
```

## 11.7 Rules, colors and vertical adjustment

### \extrarowsep and \tabulinesep

**\extrarowsep**

\extrarowsep makes the assignment for both \extrarowheight and \extrarowdepth.

The macro may be prefixed by \global.

```
\newcommand*\extrarowsep{\edef\tabu@C@extra{\the\numexpr\tabu@C@extra+1}%
    \iftabu@everyrow    \aftergroup\tabu@Gextra
    \else               \aftergroup\tabu@n@Gextra
    \fi
    \@ifnextchar={\tabu@gobbletoken\tabu@extra} \tabu@extra
}% \extrarowsep
\def\tabu@extra {\@ifnextchar_%
    {\tabu@gobbletoken{\tabu@setextra\extrarowheight \extrarowdepth}}
    {\ifx ^\@let@token \def\tabu@temp{%
            \tabu@gobbletoken{\tabu@setextra\extrarowdepth \extrarowheight}}%
    \else   \let\tabu@temp \@empty
            \afterassignment \tabu@setextrasep \extrarowdepth
    \fi \tabu@temp}%
}% \tabu@extra
\def\tabu@setextra #1#2{\def\tabu@temp{\tabu@extr@#1#2}\afterassignment\tabu@temp#2}
\def\tabu@extr@ #1#2{\@ifnextchar^%
    {\tabu@gobbletoken{\tabu@setextra\extrarowdepth \extrarowheight}}
    {\ifx _\@let@token \def\tabu@temp{%
            \tabu@gobbletoken{\tabu@setextra\extrarowheight \extrarowdepth}}%
    \else   \let\tabu@temp \@empty
            \tabu@Gsave \tabu@G@extra \tabu@C@extra \extrarowheight \extrarowdepth
    \fi \tabu@temp}%
}% \tabu@extr@
\def\tabu@setextrasep {\extrarowheight=\extrarowdepth
    \tabu@Gsave \tabu@G@extra \tabu@C@extra \extrarowheight \extrarowdepth
}% \tabu@setextrasep
\def\tabu@Gextra{\ifx \tabu@G@extra\@empty \else {\tabu@Rextra}\fi}
\def\tabu@n@Gextra{\ifx \tabu@G@extra\@empty \else \noalign{\tabu@Rextra}\fi}
```

```
\def\tabu@Rextra{\tabu@Grestore \tabu@G@extra \tabu@C@extra}
\let\tabu@C@extra \z@
\let\tabu@G@extra \@empty
```

**\tabulinesep**

\tabulinesep makes the assignment for both \abovetabulinesep and \belowtabulinesep. The macro may be prefixed by \global.

```
\newcommand*\tabulinesep{\edef\tabu@C@linesep{\the\numexpr\tabu@C@linesep+1}%
    \iftabu@everyrow    \aftergroup\tabu@Glinesep
    \else               \aftergroup\tabu@n@Glinesep
    \fi
    \@ifnextchar={\tabu@gobbletoken\tabu@linesep} \tabu@linesep
}% \tabulinesep
\def\tabu@linesep {\@ifnextchar_%
    {\tabu@gobbletoken{\tabu@setsep\abovetabulinesep \belowtabulinesep}}
    {\ifx ^\@let@token \def\tabu@temp{%
            \tabu@gobbletoken{\tabu@setsep\belowtabulinesep \abovetabulinesep}}%
     \else  \let\tabu@temp \@empty
            \afterassignment \tabu@setlinesep \abovetabulinesep
     \fi \tabu@temp}%
}% \tabu@linesep
\def\tabu@setsep #1#2{\def\tabu@temp{\tabu@sets@p#1#2}\afterassignment\tabu@temp#2}
\def\tabu@sets@p #1#2{\@ifnextchar^%
    {\tabu@gobbletoken{\tabu@setsep\belowtabulinesep \abovetabulinesep}}
    {\ifx _\@let@token \def\tabu@temp{%
            \tabu@gobbletoken{\tabu@setsep\abovetabulinesep \belowtabulinesep}}%
    \else   \let\tabu@temp \@empty
            \tabu@Gsave \tabu@G@linesep \tabu@C@linesep \abovetabulinesep \belowtabulinesep
    \fi \tabu@temp}%
}% \tabu@sets@p
\def\tabu@setlinesep {\belowtabulinesep=\abovetabulinesep
    \tabu@Gsave \tabu@G@linesep \tabu@C@linesep \abovetabulinesep \belowtabulinesep
}% \tabu@setlinesep
\def\tabu@Glinesep{\ifx \tabu@G@linesep\@empty \else {\tabu@Rlinesep}\fi}
\def\tabu@n@Glinesep{\ifx \tabu@G@linesep\@empty \else \noalign{\tabu@Rlinesep}\fi}
\def\tabu@Rlinesep{\tabu@Grestore \tabu@G@linesep \tabu@C@linesep}
\let\tabu@C@linesep \z@
\let\tabu@G@linesep \@empty
```

**\tabu@Gsave**
**\tabu@Grestore**

Utility macros to implement the possibility to prefix a macro by \global.

```
\def\tabu@Gsave #1#2#3#4{\xdef#1{#1%
    \toks#2{\toks\the\currentgrouplevel{\global#3\the#3\global#4\the#4}}}%
}% \tabu@Gsave
\def\tabu@Grestore#1#2{%
    \toks#2{}#1\toks\currentgrouplevel\expandafter{\expandafter}\the\toks#2\relax
    \ifcat$\the\toks\currentgrouplevel$\else
        \global\let#1\@empty \global\let#2\z@
        \the\toks\currentgrouplevel
    \fi
}% \tabu@Grestore
```

## Setting code for every row

**\everyrow** As long as `tabu` needs to execute some code at `\everycr`, it's not difficult to provide a command to give the user the opportunity to execute its own arbitrary code. However, `\everyrow` will be used almost only with `\hline` (or `\tabucline` or `\midrule`).

`\everyrow` can be changed anywhere inside the `tabu`: at the end of a row, or even inside a cell.

The rows LaTeX counter **taburow must not be changed by the user!**.

The settings are saved in a "locally-global" way...

```
\newcommand*\everyrow{\tabu@everyrow@bgroup
                      \tabu@start \z@ \tabu@stop \z@ \tabu@evrstartstop
}% \everyrow
\def\tabu@evrstartstop {\@ifnextchar^%
    {\afterassignment \tabu@evrstartstop \tabu@stop=}%
    {\ifx ^\@let@token
            \afterassignment\tabu@evrstartstop \tabu@start=%
     \else  \afterassignment\tabu@everyr@w     \toks@
     \fi}%
}% \tabu@evrstartstop
\def\tabu@everyr@w {%
    \xdef\tabu@everyrow{%
        \noexpand\tabu@everyrowfalse
        \let\noalign \relax
        \noexpand\tabu@rowfontreset
        \iftabu@colortbl \noexpand\tabu@rc@ \fi % \taburowcolors
        \let\noexpand\tabu@docline \noexpand\tabu@docline@evr
        \the\toks@
        \noexpand\tabu@evrh@@k
        \noexpand\tabu@rearstrut
        \global\advance\c@taburow \@ne}%
    \iftabu@everyrow \toks@\expandafter
        {\expandafter\def\expandafter\tabu@evr@L\expandafter{\the\toks@}\ignorespaces}%
    \else \xdef\tabu@evr@G{\the\toks@}%
    \fi
    \tabu@everyrow@egroup
}% \tabu@everyr@w
\def\tabu@evr {\def\tabu@evrh@@k}         % for internal use only
\tabu@evr{}
```

## Setting line styles and colors

**\newtabulinestyle** `\newtabulinestyle` {style=spec.,style=spec,style=spec}

All the job is done by `\tabu@getline`. New line style specification are always defined globally, and can be overwritten without warning...

```
\newcommand*\newtabulinestyle [1]{%
    {\@for \@tempa :=#1\do{\expandafter\tabu@newlinestyle \@tempa==\@nil}}%
}% \newtabulinestyle
\def\tabu@newlinestyle #1=#2=#3\@nil{\tabu@getline {#2}%
    \tabu@sanitizearg {#1}\@tempa
    \ifodd 1\ifx \@tempa\@empty \ifdefined\tabu@linestyle@ 0 \fi\fi
    \global\expandafter\let
        \csname tabu@linestyle@\@tempa \endcsname =\tabu@thestyle \fi
}% \tabu@newlinestyle
```

**\tabulinestyle** `\tabulinestyle` {style name} or `\tabulinestyle`line specs / leader

The job is done by `\tabu@getline`. The settings as usual, are stored in a "locally-global" way...

```
\newcommand*\tabulinestyle [1]{\tabu@everyrow@bgroup \tabu@getline{#1}%
```

```
    \iftabu@everyrow
        \toks@\expandafter{\expandafter \def \expandafter
                    \tabu@ls@L\expandafter{\tabu@thestyle}\ignorespaces}%
        \gdef\tabu@ls@{\tabu@ls@L}%
    \else
        \global\let\tabu@ls@G \tabu@thestyle
        \gdef\tabu@ls@{\tabu@ls@G}%
    \fi
    \tabu@everyrow@egroup
}% \tabulinestyle
```

**\taburulecolor** colortbl provides **\arrayrulecolor**, but the definition is global and must be restores manually after the table. \taburulecolor works with the same scheme as \everyrow: even if the definition of the rules colors must be global (because we it can be changed inside the tabular) the value is not restored globally at the end of the environment.

Instead, **\tabu@arc@L** stores locally the color definition (*ie.*its definition is relative to the group level before the entry inside the **tabu** environment).

This is the same for **\doublerulesepcolor** (which may be given as an optional argument to \taburulecolor): colortbl makes the definition global, while Tℵb⊂ keeps grouping level into mind (“locally-global” settings).

```
\newcommand*\taburulecolor{\tabu@everyrow@bgroup \tabu@textbar \tabu@rulecolor}
\def\tabu@rulecolor #1{\toks@{}%
    \def\tabu@temp #1##1#1{\tabu@ruledrsc{##1}}\@ifnextchar #1%
                                                    \tabu@temp
                                                 \tabu@rulearc
}% \tabu@rulecolor
\def\tabu@ruledrsc #1{\edef\tabu@temp{#1}\tabu@strtrim\tabu@temp
    \ifx \tabu@temp\@empty \def\tabu@temp{\tabu@rule@drsc@ {}{}}%
    \else \edef\tabu@temp{\noexpand\tabu@rule@drsc@ {}{\tabu@temp}}%
    \fi
    \tabu@temp
}% \tabu@ruledrsc@
\def\tabu@ruledrsc@   #1#{\tabu@rule@drsc@ {#1}}
\def\tabu@rule@drsc@ #1#2{%
    \iftabu@everyrow
        \ifx \\#1#2\\\toks@{\let\CT@drsc@ \relax}%
        \else        \toks@{\def\CT@drsc@{\color #1{#2}}}%
        \fi
    \else
        \ifx \\#1#2\\\global\let\CT@drsc@ \relax
        \else        \gdef\CT@drsc@{\color #1{#2}}%
        \fi
    \fi
    \tabu@rulearc
}% \tabu@rule@drsc@
\def\tabu@rulearc    #1#{\tabu@rule@arc@ {#1}}
\def\tabu@rule@arc@ #1#2{%
    \iftabu@everyrow
        \ifx \\#1#2\\\toks@\expandafter{\the\toks@ \def\CT@arc@{}}%
        \else        \toks@\expandafter{\the\toks@ \def\CT@arc@{\color #1{#2}}}%
        \fi
        \toks@\expandafter{\the\toks@
            \let\tabu@arc@L  \CT@arc@
            \let\tabu@drsc@L \CT@drsc@
            \ignorespaces}%
    \else
        \ifx \\#1#2\\\gdef\CT@arc@{}%
        \else        \gdef\CT@arc@{\color #1{#2}}%
        \fi
        \global\let\tabu@arc@G  \CT@arc@
        \global\let\tabu@drsc@G \CT@drsc@
    \fi
    \tabu@everyrow@egroup
}% \tabu@rule@arc@
```

**\taburowcolors** \taburowcolors {number}⟨*number*⟩{first color .. last color}

The aim of the game is to define the process that will be executed at \everyrow.

After that, the usual process for “locally-global” settings is plugged into `\tabu@cleanup` and `\tabu@reset`...

```
\def\taburowcolors {\tabu@everyrow@bgroup \@testopt \tabu@rowcolors 1}
\def\tabu@rowcolors [#1]#2#{\tabu@rowc@lors{#1}{#2}}
\def\tabu@rowc@lors #1#2#3{%
    \toks@{}\@defaultunits \count@      =\number0#2\relax \@nnil
            \@defaultunits \tabu@start  =\number0#1\relax \@nnil
    \ifnum \count@<\tw@ \count@=\tw@ \fi
    \advance\tabu@start \m@ne
    \ifnum \tabu@start<\z@ \tabu@start \z@ \fi
    \tabu@rowcolorseries #3\in@..\in@ \@nnil
}% \tabu@rowcolors
\def\tabu@rowcolorseries #1..#2\in@ #3\@nnil {%
    \ifx \in@#1\relax
        \iftabu@everyrow \toks@{\def\tabu@rc@{}\let\tabu@rc@L \tabu@rc@}%
        \else   \gdef\tabu@rc@{}\global\let\tabu@rc@G \tabu@rc@
        \fi
    \else
        \ifx \\#2\\\tabu@rowcolorserieserror \fi
        \tabu@sanitizearg{#1}\tabu@temp
        \tabu@sanitizearg{#2}\@tempa
        \advance\count@ \m@ne
    \iftabu@everyrow
        \def\tabu@rc@ ##1##2##3##4{\def\tabu@rc@{%
            \ifnum ##2=\c@taburow
                \definecolorseries{tabu@rcseries@\the\tabu@nested}{rgb}{last}{##3}{##4}\fi
            \ifnum \c@taburow<##2 \else
                \ifnum \tabu@modulo {\c@taburow-##2}{##1+1}=\z@
                    \resetcolorseries[{##1}]{tabu@rcseries@\the\tabu@nested}\fi
                \xglobal\colorlet{tabu@rc@\the\tabu@nested}{tabu@rcseries@\the\tabu@nested!!+}%
                \rowcolor{tabu@rc@\the\tabu@nested}\fi}%
        }\edef\x{\noexpand\tabu@rc@           {\the\count@}
                                          {\the\tabu@start}
                                               {\tabu@temp}
                                                  {\@tempa}%
                }\x
        \toks@\expandafter{\expandafter\def\expandafter\tabu@rc@\expandafter{\tabu@rc@}}%
        \toks@\expandafter{\the\toks@ \let\tabu@rc@L \tabu@rc@ \ignorespaces}%
    \else   % inside \noalign
        \definecolorseries{tabu@rcseries@\the\tabu@nested}{rgb}{last}{\tabu@temp}{\@tempa}%
        \expandafter\resetcolorseries\expandafter[\the\count@]{tabu@rcseries@\the\tabu@nested}%
        \xglobal\colorlet{tabu@rc@\the\tabu@nested}{tabu@rcseries@\the\tabu@nested!!+}%
        \let\noalign \relax \rowcolor{tabu@rc@\the\tabu@nested}%
        \def\tabu@rc@ ##1##2{\gdef\tabu@rc@{%
            \ifnum \tabu@modulo {\c@taburow-##2}{##1+1}=\@ne
                \resetcolorseries[{##1}]{tabu@rcseries@\the\tabu@nested}\fi
            \xglobal\colorlet{tabu@rc@\the\tabu@nested}{tabu@rcseries@\the\tabu@nested!!+}%
            \rowcolor{tabu@rc@\the\tabu@nested}}%
        }\edef\x{\noexpand\tabu@rc@{\the\count@}{\the\c@taburow}}\x
        \global\let\tabu@rc@G \tabu@rc@
    \fi
    \fi
    \tabu@everyrow@egroup
}% \tabu@rowcolorseries
\tabuDisableCommands {\let\tabu@rc@ \@empty }
\def\tabu@rowcolorserieserror {\PackageError{tabu}
    {Invalid syntax for \string\taburowcolors
    \MessageBreak Please look at the documentation!}\@ehd
}% \tabu@rowcolorserieserror
```

**\tabureset** Simply – and locally – reset the default values for \tabulinesep (0pt), \extrarowsep (0pt), \extratabsurround (0pt), \tabulinestyle {}, \everyrow {} and \taburulecolor ||{}.

```
\newcommand*\tabureset {%
    \tabulinesep=\z@ \extrarowsep=\z@ \extratabsurround=\z@
    \tabulinestyle{}\everyrow{}\taburulecolor||{}\taburowcolors{}%
}% \tabureset
```

### Parsing line styles

**\tabu@getline** This macro parses a line specification argument of the form:

3pt BlanchedAlmond on 4pt Crimsom off 2pt ForestGreen

Note that Crimson will overwrite BlanchedAlmond in this case: the color for the line dash may be specified after the line width or after the line dash length.

The process uses `\scantokens` on the argument given by the user, which is first expanded in a context where the babel switch `\if@save@actives` is set to **true**. Then `\scantokens` is used on the argument in a group where the letter "**o**" is active, and defined to be a macro which rewrites the line specification. Incidentally, the comma is active too, and expands to a space. This way the initial argument is "genetically modified", so that it becomes very easy to assign dimensions (thickness, dash length and gap length) and colors separately.

For example: 3pt BlanchedAlmond on 4pt Crimson will be expanded in a context where "**o**" is active (and equal to **\tabu@oxiii**, the **xiii** suffix means "active" *ie.*`\catcode` = 13).

Then the "**o**" in BlanchedAlmond is rewritten as follow:

1. "**o**" sees "**n**" after itself, then it expands **\tabu@onxiii**.
2. **\tabu@onxiii** sees a character whose catcode is not other, then the rewriting process is aborted, and "**ond**" is rewritten as "**ond**" where the "**o**" is not active but the usual letter "**o**".

The next "**o**" is rewritten as follow:

1. "**o**" sees "**n**" after itself, then it expands **\tabu@onxiii**.
2. **\tabu@onxiii** sees a space (which is active): it calls back itself again,
3. **\tabu@onxiii** sees a character whose catcode is other: then the sequence "**on␣3**" is rewritten as:
   "**\tabu@ \tabu@on** =4pt Crimson"

Finally the whole argument is rewritten as:

**\tabu@ \tabu@thick** =3pt BlanchedAlmond **\tabu@ \tabu@on** =4pt Crimson **\tabu@ \tabu@**

Define **\tabu@** as an appropriate macro which uses **\afterassignment** to:

1. Assign the corresponding dimension (thickness, dash length or gap length).
2. Collect the rest until the next **\tabu@**, trim spaces and check if the color exists.

Limitation: A color name must not contain a sequence that matches on of the patterns:

...**on**⟨*a character of category 12*⟩... or ...**off**⟨*a character of category 12*⟩...

But this "limitation" is not too heavy, I suppose...

The result is `\tabu@thestyle`: a `tabu` line style to be used to rewrite a `|` column, for `\tabucline`.

We use locally the LaTeX defined dimen registers `\@tempdima`, `\@tempdimb` and `\@tempdimc`. For clarity, their names are `\tabu@thick`, `\tabu@on` and `\tabu@off` here...

```
\def\tabu@getline #1{\begingroup
    \csname \ifcsname if@safe@actives\endcsname         % <babel>
                      @safe@activestrue\else
                      relax\fi       \endcsname
    \edef\tabu@temp{#1}\tabu@sanitizearg{#1}\@tempa
    \let\tabu@thestyle \relax
    \ifcsname tabu@linestyle@\@tempa \endcsname
            \edef\tabu@thestyle{\endgroup
                \def\tabu@thestyle{\expandafter\noexpand
                    \csname tabu@linestyle@\@tempa\endcsname}%
            }\tabu@thestyle
    \else   \expandafter\tabu@definestyle \tabu@temp \@nil
    \fi
}% \tabu@getline
```

**\tabu@definestyle** Here is the `\scantokens` stuff.

```
\def\tabu@definestyle #1#2\@nil {\endlinechar \m@ne \makeatletter
    \tabu@thick \maxdimen  \tabu@on \maxdimen    \tabu@off \maxdimen
    \let\tabu@c@lon \@undefined  \let\tabu@c@loff \@undefined
    \ifodd 1\ifcat .#1\else\ifcat\relax #1\else 0\fi\fi % catcode 12 or non expandable cs
            \def\tabu@temp{\tabu@getparam{thick}}%
    \else   \def\tabu@temp{\tabu@getparam{thick}\maxdimen}%
    \fi
    {%
        \let\tabu@ \relax
        \def\:{\obeyspaces \tabu@oXIII \tabu@commaXIII \edef\:}% (space active \: happy ;-))
        \scantokens{\:{\tabu@temp #1#2 \tabu@\tabu@}}%
                        \expandafter}\expandafter
                                \def\expandafter\:\expandafter{\:}% line spec rewritten now ;-)
    \def\;{\def\:}%
    \scantokens\expandafter{\expandafter\;\expandafter{\:}}% space is now inactive (catcode 10)
    \let\tabu@ \tabu@getcolor    \:%    all arguments are ready now ;-)
    \ifdefined\tabu@c@lon \else \let\tabu@c@lon\@empty \fi
    \ifx \tabu@c@lon\@empty \def\tabu@c@lon{\CT@arc@}\fi
    \ifdefined\tabu@c@loff \else \let\tabu@c@loff \@empty        \fi
    \ifdim \tabu@on=\maxdimen \ifdim \tabu@off<\maxdimen
                                  \tabu@on \tabulineon        \fi\fi
    \ifdim \tabu@off=\maxdimen \ifdim \tabu@on<\maxdimen
                                  \tabu@off \tabulineoff      \fi\fi
    \ifodd 1\ifdim \tabu@off=\maxdimen \ifdim \tabu@on=\maxdimen 0 \fi\fi
            \in@true    % <leaders>
    \else   \in@false   % <rule>
    \fi
    \ifdim\tabu@thick=\maxdimen \def\tabu@thick{\arrayrulewidth}%
    \else                       \edef\tabu@thick{\the\tabu@thick}%
    \fi
    \edef \tabu@thestyle ##1##2{\endgroup
        \def\tabu@thestyle{%
            \ifin@  \noexpand\tabu@leadersstyle {\tabu@thick}
                                                {\the\tabu@on}{##1}
                                                {\the\tabu@off}{##2}%
            \else   \noexpand\tabu@rulesstyle
                            {##1\vrule width \tabu@thick}%
                            {##1\leaders \hrule height \tabu@thick \hfil}%
            \fi}%
    }\expandafter \expandafter
        \expandafter \tabu@thestyle \expandafter
            \expandafter \expandafter
                {\expandafter\tabu@c@lon\expandafter}\expandafter{\tabu@c@loff}%
}% \tabu@definestyle
```

**\tabu@onxiii**
**\tabu@ofxiii**
**\tabu@offiii**

We have to define the active "`o`" character, which looks for the next tokens, trying to find a pattern like `on`⟨*category 12*⟩ or `off`⟨*category 12*⟩ (possibly with – active – spaces between `on` or `off` and the next character of catcode 12).

```
{\catcode`\O=\active \lccode`\O=`\o \catcode`\,=\active
    \lowercase{\gdef\tabu@oXIII {\catcode`\o=\active \let O=\tabu@oxiii}}
    \gdef\tabu@commaXIII {\catcode`\,=\active \let ,=\space}
}% \catcode
\def\tabu@oxiii #1{%
    \ifcase  \ifx n#1\z@ \else
             \ifx f#1\@ne\else
             \tw@       \fi\fi
          \expandafter\tabu@onxiii
    \or   \expandafter\tabu@ofxiii
    \else o%
    \fi#1}%
\def\tabu@onxiii #1#2{%
    \ifcase  \ifx  !#2\tw@          \else
             \ifcat.\noexpand#2\z@  \else
             \ifx \tabu@spxiii#2\@ne\else
             \tw@               \fi\fi\fi
          \tabu@getparam{on}#2\expandafter\@gobble
    \or   \expandafter\tabu@onxiii    % (space is active)
    \else o\expandafter\@firstofone
    \fi{#1#2}}%
\def\tabu@ofxiii #1#2{%
    \ifx #2f\expandafter\tabu@offxiii
    \else   o\expandafter\@firstofone
    \fi{#1#2}}
\def\tabu@offxiii #1#2{%
    \ifcase \ifx  !#2\tw@           \else
            \ifcat.\noexpand#2\z@   \else
            \ifx\tabu@spxiii#2\@ne  \else
            \tw@                \fi\fi\fi
          \tabu@getparam{off}#2\expandafter\@gobble
    \or   \expandafter\tabu@offxiii   % (space is active)
    \else o\expandafter\@firstofone
    \fi{#1#2}}
```

**\tabu@getparam**

The rewritten stuff.

```
\def\tabu@getparam #1{\tabu@ \csname tabu@#1\endcsname=}
```

**\tabu@getcolor**

`\tabu@ \tabu@on =`⟨*3pt*⟩ `Crimson\tabu@`

`\tabu@getcolor` first makes the assignment to `\tabu@on` and then looks for the color name which might have been placed before the next `\tabu@`.

```
\def\tabu@getcolor #1{% \tabu@ <- \tabu@getcolor after \edef
    \ifx \tabu@#1\else   % no more spec
        \let\tabu@theparam=#1\afterassignment \tabu@getc@l@r #1\fi
}% \tabu@getcolor
\def\tabu@getc@l@r #1\tabu@ {%
    \def\tabu@temp{#1}\tabu@strtrim \tabu@temp
    \ifx \tabu@temp\@empty
    \else%\ifcsname \string\color@\tabu@temp \endcsname  % if the color exists
        \ifx \tabu@theparam \tabu@off   \let\tabu@c@loff \tabu@c@l@r
        \else                           \let\tabu@c@lon  \tabu@c@l@r
        \fi
    %\else \tabu@warncolour{\tabu@temp}%
    \fi%\fi
    \tabu@ % next spec
}% \tabu@getc@l@r
\def\tabu@warncolour #1{\PackageWarning{tabu}
    {Color #1 is not defined. Default color used}%
}% \tabu@warncolour
```

**\tabu@leadersstyle**
**\tabu@rulesstyle**

When a style is executed, it expands either **\tabu@leadersstyle** or **\tabu@rulesstyle** depending on whether or not it contains leaders (dashed lines) or simple rules (solid lines): TEX internals allow to insert solid lines easily inside a tabular, while inserting leaders is more complex.

**\tabu@leadersstyle** eventually rebuilds the (horizontal and vertical) leaders boxes, and then define two macros: **\tabu@thevleaders** and **\tabu@thehleades**, suitable to draw vertical and horizontal lines respectively. Incidentally, **\tabu@leaders** is defined to be the parameters for the leaders.

**\tabu@rulesstyle** only defines the two macros **\tabu@thevrule** and **\tabu@thehrule**. The control sequence **\tabu@leaders** is undefined so that we know if the style contains a leader or a rule.

```
\def\tabu@leadersstyle #1#2#3#4#5{\def\tabu@leaders{{#1}{#2}{#3}{#4}{#5}}%
    \ifx \tabu@leaders\tabu@leaders@G \else
                \tabu@LEADERS{#1}{#2}{#3}{#4}{#5}\fi
}% \tabu@leadersstyle
\def\tabu@rulesstyle #1#2{\let\tabu@leaders \@undefined
        \gdef\tabu@thevrule{#1}\gdef\tabu@thehrule{#2}%
}% \tabu@rulesstyle
```

**\tabu@LEADERS**

Here the two leaders boxes **\tabu@hleads** and **\tabu@vleads** are built, as well as the leaders macros **\tabu@thehleaders** and **\tabu@thevleaders**.

```
\def\tabu@LEADERS #1#2#3#4#5{%% width, dash, dash color, gap, gap color
    {\let\color \tabu@color % => during trials ->  \color = \tabu@nocolor
    {%                      %    but the leaders boxes should have colors !
    \def\@therule{\vrule}\def\@thick{height}\def\@length{width}%
    \def\@box{\hbox}\def\@unbox{\unhbox}\def\@elt{\wd}%
    \def\@skip{\hskip}\def\@ss{\hss}\def\tabu@leads{\tabu@hleads}%
    \tabu@l@@d@rs {#1}{#2}{#3}{#4}{#5}%
    \global\let\tabu@thehleaders \tabu@theleaders
    }%
    {%
    \def\@therule{\hrule}\def\@thick{width}\def\@length{height}%
    \def\@box{\vbox}\def\@unbox{\unvbox}\def\@elt{\ht}%
    \def\@skip{\vskip}\def\@ss{\vss}\def\tabu@leads{\tabu@vleads}%
    \tabu@l@@d@rs {#1}{#2}{#3}{#4}{#5}%
    \global\let\tabu@thevleaders \tabu@theleaders
    }%
    \gdef\tabu@leaders@G{{#1}{#2}{#3}{#4}{#5}}%
    }%
}% \tabu@LEADERS
\def\tabu@therule #1#2{\@therule \@thick#1\@length\dimexpr#2/2 \@depth\z@}
\def\tabu@l@@d@rs #1#2#3#4#5{%% width, dash, dash color, gap, gap color
    \global\setbox \tabu@leads=\@box{%
        {#3\tabu@therule{#1}{#2}}%
        \ifx\\#5\\\@skip#4\else{#5\tabu@therule{#1}{#4*2}}\fi
        {#3\tabu@therule{#1}{#2}}}%
    \global\setbox\tabu@leads=\@box to\@elt\tabu@leads{\@ss
```

```
        {#3\tabu@therule{#1}{#2}}\@unbox\tabu@leads}%
    \edef\tabu@theleaders ##1{\def\noexpand\tabu@theleaders {%
        {##1\tabu@therule{#1}{#2}}%
        \xleaders \copy\tabu@leads \@ss
        \tabu@therule{0pt}{-#2}{##1\tabu@therule{#1}{#2}}}%
    }\tabu@theleaders{#3}%
}% \tabu@l@@d@rs
```

## 11.8 The entry inside tabu

### \tabu, \endtabu, \longtabu and \endlontabu

**\tabu** \tabu and \longtabu are the commands of the environments.

**\endtabu** \endtabu is \endtabular or \endarray in math mode.

```
\newcommand*\tabu {\tabu@longfalse
    \ifmmode \def\tabu@ {\array}\def\endtabu {\endarray}%
    \else  \def\tabu@ {\tabu@tabular}\def\endtabu {\endtabular}\fi
    \expandafter\let\csname tabu*\endcsname \tabu
    \expandafter\def\csname endtabu*\endcsname{\endtabu}%
    \tabu@spreadfalse \tabu@negcoeffalse \tabu@settarget
}% {tabu}
\let\tabu@tabular \tabular % <For LyX: some users redefine \tabular...>
\expandafter\def\csname tabu*\endcsname{\tabuscantokenstrue \tabu}
\newcommand*\longtabu {\tabu@longtrue
   \ifmmode\PackageError{tabu}{longtabu not allowed in math mode}\fi
   \def\tabu@{\longtable}\def\endlongtabu{\endlongtable}%
   \LTchunksize=\@M
   \expandafter\let\csname tabu*\endcsname \tabu
   \expandafter\def\csname endlongtabu*\endcsname{\endlongtabu}%
   \let\LT@startpbox \tabu@LT@startpbox % \everypar{ array struts }
   \tabu@spreadfalse \tabu@negcoeffalse \tabu@settarget
}% {longtabu}
\expandafter\def\csname longtabu*\endcsname{\tabuscantokenstrue \longtabu}
\def\tabu@nolongtabu{\PackageError{tabu}
   {longtabu requires the longtable package}\@ehd}
```

### Setting the tabu target

**\tabu@settarget** The macro sets \tabu@target (a dimen) to the value specified for “tabu to” or “tabu spread”.

**\tabu@begin**

```
\def\tabu@settarget {\futurelet\@let@token \tabu@sett@rget }
\def\tabu@sett@rget {\tabu@target \z@
   \ifcase \ifx \bgroup\@let@token   \z@  \else
           \ifx \@sptoken\@let@token \@ne \else
           \if t\@let@token          \tw@ \else
           \if s\@let@token          \thr@@\else
           \z@\fi\fi\fi\fi
         \expandafter\tabu@begin
   \or   \expandafter\tabu@gobblespace\expandafter\tabu@settarget
   \or   \expandafter\tabu@to
   \or   \expandafter\tabu@spread
   \fi
}% \tabu@sett@rget
\def\tabu@to to{\def\tabu@halignto{to}\tabu@gettarget}
\def\tabu@spread spread{\tabu@spreadtrue\def\tabu@halignto{spread}\tabu@gettarget}
\def\tabu@gettarget {\afterassignment\tabu@linegoaltarget \tabu@target }
\def\tabu@linegoaltarget {\futurelet\tabu@temp \tabu@linegoalt@rget }
\def\tabu@linegoalt@rget {%
```

```
   \ifx \tabu@temp\LNGL@setlinegoal
        \LNGL@setlinegoal \expandafter \@firstoftwo \fi % @gobbles \LNGL@setlinegoal
   \tabu@begin
}% \tabu@linegoalt@rget
\def\tabu@begin #1#{%
    \iftabu@measuring \expandafter\tabu@nestedmeasure \fi
    \ifdim \tabu@target=\z@ \let\tabu@halignto \@empty
    \else                   \edef\tabu@halignto{\tabu@halignto\the\tabu@target}%
    \fi
    \@testopt \tabu@tabu@ \tabu@aligndefault #1\@nil
}% \tabu@begin
\long\def\tabu@tabu@ [#1]#2\@nil #3{\tabu@setup
   \def\tabu@align {#1}\def\tabu@savedpream{\NC@find #3}%
   \tabu@ [\tabu@align ]#2{#3\tabu@rewritefirst }%
}% \tabu@tabu@
\def\tabu@nestedmeasure {%
    \ifodd 1\iftabu@spread \else \ifdim\tabu@target=\z@ \else 0 \fi\fi\relax
            \tabu@spreadtrue
    \else   \begingroup \iffalse{\fi \ifnum0=`}\fi
            \toks@{}\def\tabu@stack{b}%
            \expandafter\tabu@collectbody\expandafter\tabu@quickrule
                                         \expandafter\endgroup
    \fi
}% \tabu@nestedmeasure
\def\tabu@quickrule {\indent\vrule height\z@ depth\z@ width\tabu@target}
```

**\tabu@setup**
**\tabu@init**
**\tabu@indent**

**\tabu@init** is expanded only when **tabu** is not nested. In this case, and if **\parindent** > 0, and if **\tabudefaulttarget** =**\linewidth**, the correction of the default target for paragraph indentation is executed (see paragraph indentation).

```
\def\tabu@setup{\tabu@alloc@
    \ifcase \tabu@nested
        \ifmmode \else \iftabu@spread\else \ifdim\tabu@target=\z@
            \let\tabu@afterendpar \par
        \fi\fi\fi
        \def\tabu@aligndefault{c}\tabu@init \tabu@indent
    \else       % <nested tabu>
        \def\tabu@aligndefault{t}\let\tabudefaulttarget \linewidth
    \fi
    \let\tabu@thetarget \tabudefaulttarget \let\tabu@restored \@undefined
    \edef\tabu@NC@list{\the\NC@list}\NC@list{\NC@do \tabu@rewritefirst}%
    \everycr{}\let\@startpbox \tabu@startpbox % for nested tabu inside longtabu...
              \let\@endpbox   \tabu@endpbox   % idem "    "    "    "    "    "
              \let\@tabarray  \tabu@tabarray  % idem "    "    "    "    "    "
    \tabu@setcleanup \tabu@setreset
}% \tabu@setup
\def\tabu@init{\tabu@starttimer \tabu@measuringfalse
    \edef\tabu@hfuzz  {\the\dimexpr\hfuzz+1sp}\global\tabu@footnotes{}%
    \let\firsthline    \tabu@firsthline   \let\lasthline      \tabu@lasthline
    \let\firstline     \tabu@firstline    \let\lastline       \tabu@lastline
    \let\hline         \tabu@hline        \let\@xhline        \tabu@xhline
    \let\color         \tabu@color        \let\@arstrutbox    \tabu@arstrutbox
    \iftabu@colortbl\else\let\LT@@hline   \tabu@LT@@hline \fi
    \tabu@trivlist     %<restore \\=\@normalcr inside lists>
    \let\@footnotetext \tabu@footnotetext \let\@xfootnotetext \tabu@xfootnotetext
    \let\@xfootnote    \tabu@xfootnote    \let\centering      \tabu@centering
    \let\raggedright   \tabu@raggedright  \let\raggedleft     \tabu@raggedleft
    \let\tabudecimal   \tabu@tabudecimal  \let\Centering      \tabu@Centering
    \let\RaggedRight   \tabu@RaggedRight  \let\RaggedLeft     \tabu@RaggedLeft
    \let\justifying    \tabu@justifying   \let\rowfont         \tabu@rowfont
    \let\fbox          \tabu@fbox         \let\color@b@x       \tabu@color@b@x
    \let\tabu@@everycr \everycr           \let\tabu@@everypar \everypar
    \let\tabu@prepnext@tokORI \prepnext@tok\let\prepnext@tok  \tabu@prepnext@tok
    \let\tabu@multicolumnORI\multicolumn  \let\multicolumn    \tabu@multicolumn
    \let\tabu@startpbox \@startpbox      % for nested tabu inside longtabu pfff !!!
    \let\tabu@endpbox   \@endpbox        % idem  "    "    "    "    "    "    "
    \let\tabu@tabarray  \@tabarray       % idem  "    "    "    "    "    "    "
    \tabu@adl@fix     \let\endarray      \tabu@endarray % <fix> colortbl & arydshln (delarray)
    \iftabu@colortbl\CT@everycr\expandafter{\expandafter\iftabu@everyrow \the\CT@everycr \fi}\fi
}% \tabu@init
\def\tabu@indent{% correction for indentation
    \ifdim \parindent>\z@\ifx \linewidth\tabudefaulttarget
%
    \everypar\expandafter{%
        \the\everypar\everypar\expandafter{\the\everypar}%
            \setbox\z@=\lastbox
            \ifdim\wd\z@>\z@ \edef\tabu@thetarget
                {\the\dimexpr -\wd\z@+\tabudefaulttarget}\fi
            \box\z@}%
    \fi\fi
}% \tabu@indent
```

**`\tabu@setcleanup`** We have to save locally (in the group of the environment) the current value of the last global assignments to \CT@arc@, \CT@drsc@, \tabu@ls@ etc.

**`\tabu@cleanup`** Restoration will be done globally after the box that contains the tabular by `\tabu@cleanup`.

```
\def\tabu@setcleanup {% saves last global assignments
    \ifodd 1\ifmmode \else \iftabu@long \else 0\fi\fi\relax
        \def\tabu@aftergroupcleanup{%
                \def\tabu@aftergroupcleanup{\aftergroup\tabu@cleanup}}%
    \else
        \def\tabu@aftergroupcleanup{%
                \aftergroup\aftergroup\aftergroup\tabu@cleanup
                \let\tabu@aftergroupcleanup \relax}%
    \fi
    \let\tabu@arc@Gsave         \tabu@arc@G
    \let\tabu@arc@G             \tabu@arc@L   % <init>
    \let\tabu@drsc@Gsave        \tabu@drsc@G
    \let\tabu@drsc@G            \tabu@drsc@L  % <init>
    \let\tabu@ls@Gsave          \tabu@ls@G
    \let\tabu@ls@G              \tabu@ls@L    % <init>
    \let\tabu@rc@Gsave          \tabu@rc@G
    \let\tabu@rc@G              \tabu@rc@L    % <init>
    \let\tabu@evr@Gsave         \tabu@evr@G
    \let\tabu@evr@G             \tabu@evr@L   % <init>
    \let\tabu@celllalign@save   \tabu@celllalign
    \let\tabu@cellralign@save   \tabu@cellralign
    \let\tabu@cellleft@save     \tabu@cellleft
    \let\tabu@cellright@save    \tabu@cellright
    \let\tabu@@celllalign@save  \tabu@@celllalign
    \let\tabu@@cellralign@save  \tabu@@cellralign
    \let\tabu@@cellleft@save    \tabu@@cellleft
    \let\tabu@@cellright@save   \tabu@@cellright
    \let\tabu@rowfontreset@save \tabu@rowfontreset
    \let\tabu@@rowfontreset@save\tabu@@rowfontreset
    \let\tabu@rowfontreset      \@empty
    \edef\tabu@alloc@save      {\the\tabu@alloc}%   restore at \tabu@reset
    \edef\c@taburow@save       {\the\c@taburow}%
    \edef\tabu@naturalX@save   {\the\tabu@naturalX}%
    \let\tabu@naturalXmin@save  \tabu@naturalXmin
    \let\tabu@naturalXmax@save  \tabu@naturalXmax
    \let\tabu@mkarstrut@save    \tabu@mkarstrut
    \edef\tabu@clarstrut{%
        \extrarowheight \the\dimexpr \ht\@arstrutbox-\ht\strutbox \relax
        \extrarowdepth \the\dimexpr \dp\@arstrutbox-\dp\strutbox \relax
        \let\noexpand\@arraystretch \@ne \noexpand\tabu@rearstrut}%
}% \tabu@setcleanup
\def\tabu@cleanup {\begingroup
    \globaldefs\@ne         \tabu@everyrowtrue
    \let\tabu@arc@G         \tabu@arc@Gsave
    \let\CT@arc@            \tabu@arc@G
    \let\tabu@drsc@G        \tabu@drsc@Gsave
    \let\CT@drsc@           \tabu@drsc@G
    \let\tabu@ls@G          \tabu@ls@Gsave
    \let\tabu@ls@           \tabu@ls@G
    \let\tabu@rc@G          \tabu@rc@Gsave
    \let\tabu@rc@           \tabu@rc@G
    \let\CT@do@color        \relax
    \let\tabu@evr@G         \tabu@evr@Gsave
    \let\tabu@celllalign    \tabu@celllalign@save
    \let\tabu@cellralign    \tabu@cellralign@save
    \let\tabu@cellleft      \tabu@cellleft@save
    \let\tabu@cellright     \tabu@cellright@save
    \let\tabu@@celllalign   \tabu@@celllalign@save
    \let\tabu@@cellralign   \tabu@@cellralign@save
    \let\tabu@@cellleft     \tabu@@cellleft@save
    \let\tabu@@cellright    \tabu@@cellright@save
    \let\tabu@rowfontreset  \tabu@rowfontreset@save
    \let\tabu@@rowfontreset \tabu@@rowfontreset@save
    \tabu@naturalX         =\tabu@naturalX@save
    \let\tabu@naturalXmax   \tabu@naturalXmax@save
    \let\tabu@naturalXmin   \tabu@naturalXmin@save
    \let\tabu@mkarstrut     \tabu@mkarstrut@save
    \c@taburow             =\c@taburow@save
    \ifcase \tabu@nested    \tabu@alloc \m@ne\fi
    \endgroup               % <end of \globaldefs>
    \ifcase \tabu@nested
        \the\tabu@footnotes \global\tabu@footnotes{}%
        \tabu@afterendpar   \tabu@elapsedtime
    \fi
    \tabu@clarstrut
    \everyrow\expandafter   {\tabu@evr@G}%
}% \tabu@cleanup
\let\tabu@afterendpar \relax
```

**`\tabu@setreset`** At the beginning of each trial, we have to restore the current value that were active at the entry in the `tabu` environment (for they could have been globally overwritten inside the tabular).

The same must occur when using `\usetabu` as a preamble. Values are restored locally inside the `tabu` box.

**`\tabu@reset`** `\tabu@setreset` defines `\tabu@reset` to be expanded at the beginning of each trial and when `\usetabu` is used.

```
\def\tabu@setreset {%
    \edef\tabu@savedparams {%         \relax for \tabu@message@save
        \ifmmode \col@sep \the\arraycolsep
        \else    \col@sep \the\tabcolsep \fi    \relax
        \arrayrulewidth   \the\arrayrulewidth   \relax
        \doublerulesep    \the\doublerulesep    \relax
        \extratabsurround \the\extratabsurround \relax
        \extrarowheight   \the\extrarowheight   \relax
        \extrarowdepth    \the\extrarowdepth    \relax
        \abovetabulinesep \the\abovetabulinesep \relax
        \belowtabulinesep \the\belowtabulinesep \relax
        \def\noexpand\arraystretch{\arraystretch}%
        \ifdefined\minrowclearance \minrowclearance\the\minrowclearance\relax\fi}%
    \begingroup
        \@temptokena\expandafter{\tabu@savedparams}% => only for \savetabu / \usetabu
        \ifx \tabu@arc@L\relax  \else \tabu@setsave \tabu@arc@L \fi
        \ifx \tabu@drsc@L\relax \else \tabu@setsave \tabu@drsc@L \fi
        \tabu@setsave \tabu@ls@L      \tabu@setsave \tabu@evr@L
        \expandafter \endgroup \expandafter
            \def\expandafter\tabu@saved@ \expandafter{\the\@temptokena
                \let\tabu@arc@G  \tabu@arc@L
                \let\tabu@drsc@G \tabu@drsc@L
                \let\tabu@ls@G   \tabu@ls@L
                \let\tabu@rc@G   \tabu@rc@L
                \let\tabu@evr@G  \tabu@evr@L}%
    \def\tabu@reset{\tabu@savedparams
        \tabu@everyrowtrue  \c@taburow \z@
        \let\CT@arc@        \tabu@arc@L
        \let\CT@drsc@       \tabu@drsc@L
        \let\tabu@ls@       \tabu@ls@L
        \let\tabu@rc@       \tabu@rc@L
        \global\tabu@alloc  \tabu@alloc@save
        \everyrow\expandafter{\tabu@evr@L}}%
}% \tabu@reset
\def\tabu@setsave #1{\expandafter\tabu@sets@ve #1\@nil{#1}}
\long\def\tabu@sets@ve #1\@nil #2{\@temptokena\expandafter{\the\@temptokena \def#2{#1}}}
```

## 11.9 The rewriting process: inside the “\@mkpream group”

### New column types and private (new) column types

**\tabu@newcolumntype** A helper macro to create new column types for tabu.

The column types **are not appended** to \NC@list in order to keep them local to tabu.

```
\def\tabu@newcolumntype #1{%
    \expandafter\tabu@new@columntype
        \csname NC@find@\string#1\expandafter\endcsname
        \csname NC@rewrite@\string#1\endcsname
        {#1}%
}% \tabu@newcolumntype
\def\tabu@new@columntype #1#2#3{%
    \def#1##1#3{\NC@{##1}}%
    \let#2\relax \newcommand*#2%
}% \tabu@new@columntype
```

**\tabu@privatecolumntype** Columns types defined with \tabu@privatecolumntype are "mounted" only inside the \@mkpream group of tabu.

```
\def\tabu@privatecolumntype #1{%
    \expandafter\tabu@private@columntype
        \csname NC@find@\string#1\expandafter\endcsname
        \csname NC@rewrite@\string#1\expandafter\endcsname
        \csname tabu@NC@find@\string#1\expandafter\endcsname
        \csname tabu@NC@rewrite@\string#1\endcsname
        {#1}%
}% \tabu@privatecolumntype
\def\tabu@private@columntype#1#2#3#4{%
    \g@addto@macro\tabu@privatecolumns{\let#1#3\let#2#4}%
    \tabu@new@columntype#3#4%
}% \tabu@private@columntype
\let\tabu@privatecolumns \@empty
```

## High priority columns

**`\tabucolumn`** `\tabucolumn` puts a user-defined column in high priority in the `tabu` rewriting process.

```
\newcommand*\tabucolumn [1]{\expandafter \def \expandafter
    \tabu@highprioritycolumns\expandafter{\tabu@highprioritycolumns
                                          \NC@do #1}}%
\let\tabu@highprioritycolumns \@empty
```

## Rewriting vertical lines and leaders

**`| (private column type)`** This is the rewrite macro for the `|` column type inside `tabu` and `longtabu`.

Vertical lines are *simply rewritten* as special `!` columns.

```
\tabu@privatecolumntype |{\tabu@rewritevline}
\newcommand*\tabu@rewritevline[1][]{\tabu@vlinearg{#1}%
                \expandafter \NC@find \tabu@rewritten}
```

**`\tabu@lines`** The `|` token for vertical lines may have a special catcode. `array.sty` makes the test with `\if` and therefore, it is catcode insensitiv. Here, we use `\scantokens` and check if `|` is not an *other* character.

```
\def\tabu@lines #1{%
    \ifx|#1\else \tabu@privatecolumntype #1{\tabu@rewritevline}\fi
    \NC@list\expandafter{\the\NC@list \NC@do #1}%
}% \tabu@lines@
```

**`\tabu@vlinearg`** The macro that parses the optional argument of `|` vertical lines...

```
\def\tabu@vlinearg #1{%
    \ifx\\#1\\\def\tabu@thestyle {\tabu@ls@}%
    \else\tabu@getline {#1}%
    \fi
    \def\tabu@rewritten ##1{\def\tabu@rewritten{!{##1\tabu@thevline}}%
    }\expandafter\tabu@rewritten\expandafter{\tabu@thestyle}%
    \expandafter \tabu@keepls \tabu@thestyle \@nil
}% \tabu@vlinearg
\def\tabu@keepls #1\@nil{%
    \ifcat $\@cdr #1\@nil $%
    \ifx \relax#1\else
    \ifx \tabu@ls@#1\else
        \let#1\relax
        \xdef\tabu@mkpreambuffer{\tabu@mkpreambuffer
                \tabu@savels\noexpand#1}\fi\fi\fi
}% \tabu@keepls
\def\tabu@thevline {\begingroup
    \ifdefined\tabu@leaders
        \setbox\@tempboxa=\vtop to\dimexpr
                      \ht\@arstrutbox+\dp\@arstrutbox{{\tabu@thevleaders}}%
        \ht\@tempboxa=\ht\@arstrutbox \dp\@tempboxa=\dp\@arstrutbox
        \box\@tempboxa
    \else
                \tabu@thevrule
    \fi             \endgroup
}% \tabu@thevline
```

```
\def\tabu@savels #1{%
    \expandafter\let\csname\string#1\endcsname #1%
    \expandafter\def\expandafter\tabu@reset\expandafter{\tabu@reset
                                                    \tabu@resetls#1}}%
\def\tabu@resetls #1{\expandafter\let\expandafter#1\csname\string#1\endcsname}%
```

## Vertical lines and leaders in the `\multicolumn` preamble

**`\tabu@rewritemulticolumn`** A special rewrite to allow `|[...]` in `\multicolumn` preamble inside `tabu` environment.

As long as `\multicolumn` begins with `\omit` (via `\multispan`) special care has to be taken: everything shall be purely expandable until `\omit`.

`\multicolumn` is not an environment: no group is opened apart the `\@mkpream` group. We open a semi simple group for `\multicolumn` when inside `tabu`, in order for the setup to be local (in case a user would try to embed a `tabular` inside the argument of `\multicolumn`...)

```
\tabu@newcolumntype \tabu@rewritemulticolumn{%
    \aftergroup \tabu@endrewritemulticolumn % after \@mkpream group
    \NC@list{\NC@do *}\tabu@textbar \tabu@lines
    \tabu@savedecl
    \tabu@privatecolumns
    \NC@list\expandafter{\the\expandafter\NC@list \tabu@NC@list}%
    \let\tabu@savels \relax
    \NC@find
}% \tabu@rewritemulticolumn
\def\tabu@endrewritemulticolumn{\gdef\tabu@mkpreambuffer{}\endgroup}
\def\tabu@multicolumn{\tabu@ifenvir \tabu@multic@lumn \tabu@multicolumnORI}
\long\def\tabu@multic@lumn #1#2#3{\multispan{#1}\begingroup
    \tabu@everyrowtrue
    \NC@list{\NC@do \tabu@rewritemulticolumn}%
    \expandafter\@gobbletwo % gobbles \multispan{#1}
         \tabu@multicolumnORI{#1}{\tabu@rewritemulticolumn #2}%
                {\iftabuscantokens \tabu@rescan \else \expandafter\@firstofone \fi
                {#3}}%
}% \tabu@multic@lumn
```

## Rewriting `tabu` `X` columns

**`X (private column type)`** This is the rewrite macro for `tabu` `X` columns. Such a column has an optional argument: the width coefficient for the `tabu` `X` column whose default value is 1, and may be some alignments parameters. The coefficient is used in the expression: `p{\dimexpr` ⟨*coef*⟩`\tabucolX }`

```
\tabu@privatecolumntype X[1][]{\begingroup \tabu@siunitx{\endgroup \tabu@rewriteX {#1}}}
\def\tabu@nosiunitx #1{#1{}{}\expandafter \NC@find \tabu@rewritten }
\def\tabu@siunitx   #1{\@ifnextchar \bgroup
                     {\tabu@rewriteX@Ss{#1}}
                      {\tabu@nosiunitx{#1}}}
\def\tabu@rewriteX@Ss #1#2{\@temptokena{}%
    \@defaultunits \let\tabu@temp =#2\relax\@nnil
    \ifodd 1\ifx S\tabu@temp \else \ifx s\tabu@temp \else 0 \fi\fi
        \def\NC@find{\def\NC@find >####1####2<####3\relax{#1 {####1}{####3}%
            }\expandafter\NC@find \the\@temptokena \relax
        }\expandafter\NC@rewrite@S \@gobble #2\relax
    \else \tabu@siunitxerror
    \fi
    \expandafter \NC@find \tabu@rewritten
}% \tabu@rewriteX@Ss
\def\tabu@siunitxerror {\PackageError{tabu}{Not a S nor s column !
        \MessageBreak X column can only embed siunitx S or s columns}\@ehd
}% \tabu@siunitxerror
```

**\tabu@rewriteX** This macro is expanded by during the rewriting process in case a X column is found.

\tabu@Xsum (a dimen) stores the sum of the (absolute) width coefficients.

For the first X column found in the preamble, a special setup occurs:

- if the default target is used (no target specified or tabu spread with X columns), the target: \tabu@target is set to the default, with a message in the .log file.
- \@halignto is \let to \relax to avoid its expansion in \xdef \@preamble just after \@mkpream. Indeed as long as we have to measure the natural width of the tabular, \@halign must be empty for trial steps.
- The rest of the setup is made \aftergroup (*ie.*after \xdef \@preamble which occurs inside a group) by \tabu@prep@TRIAL.

```
\def\tabu@rewriteX #1#2#3{\tabu@Xarg {#1}{#2}{#3}%
    \iftabu@measuring
    \else \tabu@measuringtrue % first X column found in the preamble
        \let\@halignto \relax   \let\tabu@halignto \relax
        \iftabu@spread \tabu@spreadtarget \tabu@target \tabu@target \z@
        \else          \tabu@spreadtarget \z@ \fi
        \ifdim \tabu@target=\z@
                \setlength\tabu@target \tabu@thetarget
                \tabu@message{\tabu@message@defaulttarget}%
        \else   \tabu@message{\tabu@message@target}\fi
    \fi
}% \tabu@rewriteX
```

**\tabu@rewriteXrestore** This macro replaces \tabu@rewriteX in the case of \usetabu.

```
\def\tabu@rewriteXrestore #1#2#3{\let\@halignto \relax
                                \def\tabu@rewritten{l}}
```

**\tabu@Xarg**
**\tabu@Xparse** A tedious (and fastidious) macro to parse the optional argument of X columns. The aim is to built \tabu@rewritten which expands to the column specification:

>{alignment} p *or* m *or* b {\dimexpr coef \tabucolX \relax }

After that array.sty make it easy: \expandafter \NC@find \tabu@rewritten

```
\def\tabu@Xarg #1#2#3{%
   \advance\tabu@Xcol \@ne      \let\tabu@Xlcr  \@empty
   \let\tabu@Xdisp    \@empty   \let\tabu@Xmath \@empty
    \ifx\\#1\\%    <shortcut when no option>
         \def\tabu@rewritten{p}\tabucolX \p@        % <default coef = 1>
    \else
         \let\tabu@rewritten \@empty   \let\tabu@temp \@empty  \tabucolX \z@
         \tabu@Xparse {}#1\relax
    \fi
    \tabu@Xrewritten{#2}{#3}%
}% \tabu@Xarg
\def\tabu@Xparse #1{\futurelet\@let@token \tabu@Xtest}
\expandafter\def\expandafter\tabu@Xparsespace\space{\tabu@Xparse{}}
\def\tabu@Xtest{%
   \ifcase \ifx \relax\@let@token \z@ \else
           \if ,\@let@token \m@ne\else
           \if p\@let@token 1\else
           \if m\@let@token 2\else
           \if b\@let@token 3\else
           \if l\@let@token 4\else
           \if c\@let@token 5\else
           \if r\@let@token 6\else
           \if j\@let@token 7\else
           \if L\@let@token 8\else
           \if C\@let@token 9\else
           \if R\@let@token 10\else
           \if J\@let@token 11\else
           \ifx \@sptoken\@let@token 12\else
           \if .\@let@token 13\else
           \if -\@let@token 13\else
           \ifcat $\@let@token 14\else
           15\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\relax
   \or \tabu@Xtype {p}%
   \or \tabu@Xtype {m}%
   \or \tabu@Xtype {b}%
   \or \tabu@Xalign \raggedright\relax
   \or \tabu@Xalign \centering\relax
   \or \tabu@Xalign \raggedleft\relax
   \or \tabu@Xalign \tabu@justify\relax
   \or \tabu@Xalign \RaggedRight\raggedright
   \or \tabu@Xalign \Centering\centering
   \or \tabu@Xalign \RaggedLeft\raggedleft
   \or \tabu@Xalign \justifying\tabu@justify
   \or \expandafter \tabu@Xparsespace
   \or \expandafter \tabu@Xcoef
   \or \expandafter \tabu@Xm@th
   \or \tabu@Xcoef{}%
   \else\expandafter \tabu@Xparse
   \fi
}% \tabu@Xtest
\def\tabu@Xalign #1#2{%
    \ifx \tabu@Xlcr\@empty \else \PackageWarning{tabu}
       {Duplicate horizontal alignment specification}\fi
    \ifdefined#1\def\tabu@Xlcr{#1}\let#1\relax
    \else       \def\tabu@Xlcr{#2}\let#2\relax\fi
    \expandafter\tabu@Xparse
}% \tabu@Xalign
\def\tabu@Xtype #1{%
    \ifx \tabu@rewritten\@empty \else \PackageWarning{tabu}
            {Duplicate vertical alignment specification}\fi
    \def\tabu@rewritten{#1}\expandafter\tabu@Xparse
}% \tabu@Xtype
\def\tabu@Xcoef#1{\edef\tabu@temp{\tabu@temp#1}%
    \afterassignment\tabu@Xc@ef \tabu@cnt\number\if-#10\fi
}% \tabu@Xcoef
\def\tabu@Xc@ef{\advance\tabucolX \tabu@temp\the\tabu@cnt\p@
    \tabu@Xparse{}%
}% \tabu@Xc@ef
\def\tabu@Xm@th #1{\futurelet \@let@token \tabu@Xd@sp}
\def\tabu@Xd@sp{\let\tabu@Xmath=$%
    \ifx $\@let@token \def\tabu@Xdisp{\displaystyle}%
            \expandafter\tabu@Xparse
    \else   \expandafter\tabu@Xparse\expandafter{\expandafter}%
    \fi
}% \tabu@Xd@sp
```

**\tabu@Xrewritten** Final step: the whole optional argument has been read, then builds the rewritten column specification.

```
\def\tabu@Xrewritten {%
   \ifx  \tabu@rewritten\@empty \def\tabu@rewritten{p}\fi
   \ifdim \tabucolX<\z@         \tabu@negcoeftrue
   \else\ifdim \tabucolX=\z@    \tabucolX \p@
   \fi\fi
   \edef\tabu@temp{{\the\tabu@Xcol}{\tabu@strippt\tabucolX}}%
   \edef\tabu@Xcoefs{\tabu@Xcoefs    \tabu@      \tabu@temp}%
   \edef\tabu@rewritten ##1##2{\def\noexpand\tabu@rewritten{%
        >{\tabu@Xlcr \ifx$\tabu@Xmath$\tabu@Xdisp\fi ##1}%
                    \tabu@rewritten {\tabu@hsize \tabu@temp}%
        <{##2\ifx$\tabu@Xmath$\fi}}%
   }\tabu@rewritten
}% \tabu@Xrewritten
```

**`\tabu@hsize`** `\tabu@hsize {X column number}{X column width coefficient}`

Depending on the sign of the coefficient, and of the stored value for the natural width of the column the `X` cell belongs to, `\tabu@hsize` returns the wanted width for the *par-box* that contains the cell content.

```
\def\tabu@hsize #1#2{%
    \ifdim #2\p@<\z@
        \ifdim \tabucolX=\maxdimen \tabu@wd{#1}\else
        \ifdim \tabu@wd{#1}<-#2\tabucolX \tabu@wd{#1}\else -#2\tabucolX\fi
        \fi
   \else #2\tabucolX
   \fi
}% \tabu@hsize
```

## Rewritting \usetabu and \preamble

The rewritting process is very simple, when all the job has been done cleverly at the time of \savetabu!!

The \savetabu macro is a bit more complex...

**`\usetabu (private column type)`** \usetabu is defined as a `tabu` new column type: loaded only inside the `\@mkpream` group inside the `tabu` environment.

```
\tabu@privatecolumntype \usetabu [1]{%
    \ifx\\#1\\\tabu@saveerr{}\else
        \@ifundefined{tabu@saved@\string#1}
            {\tabu@saveerr{#1}}
            {\let\tabu@rewriteX \tabu@rewriteXrestore
             \csname tabu@saved@\string#1\expandafter\endcsname\expandafter\@ne}%
    \fi
}% \NC@rewrite@\usetabu
```

**`\preamble (private column type)`** \preamble is defined as a `tabu` new column type: loaded only inside the `\@mkpream` group inside the `tabu` environment.

```
\tabu@privatecolumntype \preamble [1]{%
    \ifx\\#1\\\tabu@saveerr{}\else
        \@ifundefined{tabu@saved@\string#1}
            {\tabu@saveerr{#1}}
            {\csname tabu@saved@\string#1\expandafter\endcsname\expandafter\z@}%
    \fi
}% \NC@rewrite@\preamble
```

## Controlling the rewritting process

**`\tabu@rewritefirst`** This new column type is not really a column type! It is always added to a `tabu` preamble in order to do some setup before any other column is rewritten by `\@mkpream`.

Thus, `\NC@list` is simply set to `{\NC@do \tabu@rewritefirst }`. The rewritting of `\tabu@rewritefirst` will restore the original list `\NC@list`.

This “column type” sets:

- `\tabu@select` to be expanded `\aftergroup` (after the closing of the `\@mkpream` group. All the thick is there: all information collected during the rewritting of `X` columns (and vertical lines or leaders) can be *reinjected* into the group below the `\@mkpream` group, by the mean of the

**\tabu@mkpreambuffer** (globally defined).

- The private columns types are loaded by **\tabu@rewritefirst**: they will be rewritten afterwards, during the rewritting loop. This way, `X` column definition for `tabu` are only available during the rewritting process of the `tabu` preamble, making it possible (and easy) to embed a `tabularx` inside a cell of a `tabu`.
- **\save@decl** is modified inside the **\@mkpream** group, if `tabu` is in text mode.

```
\tabu@newcolumntype \tabu@rewritefirst{%
    \iftabu@long    \aftergroup \tabu@longpream  % <the whole implementation is here !>
    \else           \aftergroup \tabu@pream
    \fi
    \let\tabu@          \relax      \let\tabu@hsize     \relax
    \let\tabu@Xcoefs    \@empty     \let\tabu@savels    \relax
    \tabu@Xcol          \z@         \tabu@cnt           \tw@
    \gdef\tabu@mkpreambuffer{\tabu@{}}\tabu@measuringfalse
    \global\setbox\@arstrutbox \box\@arstrutbox
    \NC@list{\NC@do *}\tabu@textbar \tabu@lines
    \NC@list\expandafter{\the\NC@list \NC@do X}%
    \iftabu@siunitx     % <siunitx S and s columns>
            \NC@list\expandafter{\the\NC@list \NC@do S\NC@do s}\fi
    \NC@list\expandafter{\the\expandafter\NC@list \tabu@highprioritycolumns}%
    \expandafter\def\expandafter\tabu@NC@list\expandafter{%
                    \the\expandafter\NC@list \tabu@NC@list}%    % * | X S <original>
    \NC@list\expandafter{\expandafter \NC@do \expandafter\usetabu
                         \expandafter \NC@do \expandafter\preamble
                         \the\NC@list \NC@do \tabu@rewritemiddle
                                      \NC@do \tabu@rewritelast}%
    \tabu@savedecl
    \tabu@privatecolumns
    \edef\tabu@prev{\the\@temptokena}\NC@find \tabu@rewritemiddle
}% NC@rewrite@\tabu@rewritefirst
```

**\tabu@rewritemiddle**
**\tabu@rewritelast**

This new column type is rewritten after `X` columns, because it is declared by when the column **\tabu@rewritefirst** is actually rewritten. In the case where **\tabu@target** is $> 0$ (either because of "`tabu to`" or "`tabu spread`" has been called) and if there is no `X` column, then **@{\extracolsep \@flushglue }** is added at the beginning of the preamble.

To avoid duplicate margin in the `tabu` we have to test the next token in the preamble. If the next token is **|** or **!** then no margin must be added and **@{\extracolsep \@flushglue }** can be inserted at the beginning of the preamble.

Otherwise, we must insert **!{\extracolsep \@flushglue }** in order to keep the margin.

**\tabu@rewritelast** column type is loaded by **\tabu@rewritefirst** column type, only inside the **\@mkpream** group inside the `tabu` environment.

```
\tabu@newcolumntype \tabu@rewritemiddle{%
    \edef\tabu@temp{\the\@temptokena}\NC@find \tabu@rewritelast
}% \NC@rewrite@\tabu@rewritemiddle
\tabu@newcolumntype \tabu@rewritelast{%
    \ifx \tabu@temp\tabu@prev   \advance\tabu@cnt \m@ne
            \NC@list\expandafter{\tabu@NC@list \NC@do \tabu@rewritemiddle
                                               \NC@do \tabu@rewritelast}%
    \else \let\tabu@prev\tabu@temp
    \fi
    \ifcase \tabu@cnt   \expandafter\tabu@endrewrite
    \else               \expandafter\NC@find \expandafter\tabu@rewritemiddle
    \fi
}% \NC@rewrite@\tabu@rewritelast
```

### The end of the rewritting process: determining the tabu strategy

**`\tabu@endrewrite`** Determines the strategy to be executed `\aftergroup` (at the closing of the `\@mkpream` group):

0) There is no real strategy: `tabu` behaves like `tabular`, there no `X` column, and no need to measure the vertical dimensions of the cells (no dynamic spacing, no vertical leader). In case a target has been given to `tabu`, it behaves like `tabular*` and a infinite stretchability is given to the column inter-space. This is done (if required) by `\tabu@extracolsep`.

1) Measuring natural width of some (or all) columns is compulsory for `tabu spread` of `X` columns with negativ coefficients. Thereafter, the strategy nr 2 will bring into play.

2) Measuring the natural width is not necessary, or has been done before. But `tabu` contains `X` columns and trials have to be performed to reach the desired target, adjusting the `\tabucolX` dimension accordingly. Then, the strategy nr 3 may bring into play, if vertical measure is required.

3) Vertical measure of the cells is required, for vertical spacing adjustment or vertical leaders. This step can be done only if the width are known.

$>3$ The `tabu` is finished and ready to be printed !!

```
\def\tabu@endrewrite {%
    \let\tabu@temp \NC@find
    \ifx \@arrayright\relax \let\@arrayright \@empty  \fi
    \count@=%
        \ifx \@finalstrut\tabu@finalstrut \z@  % outer in mode 0 print
             \iftabu@measuring
                \xdef\tabu@mkpreambuffer{\tabu@mkpreambuffer
                    \tabu@target        \csname tabu@\the\tabu@nested.T\endcsname
                    \tabucolX           \csname tabu@\the\tabu@nested.X\endcsname
                    \edef\@halignto {\ifx\@arrayright\@empty to\tabu@target\fi}}%
             \fi
        \else\iftabu@measuring        4       % X columns
                \xdef\tabu@mkpreambuffer{\tabu@{\tabu@mkpreambuffer
                    \tabu@target        \the\tabu@target
                    \tabu@spreadtarget  \the\tabu@spreadtarget}%
                    \def\noexpand\tabu@Xcoefs{\tabu@Xcoefs}%
                    \edef\tabu@halignto{\ifx \@arrayright\@empty to\tabu@target\fi}}%
                 \let\tabu@Xcoefs \relax
             \else\ifcase\tabu@nested \thr@@  % outer, no X
                                      \global\let\tabu@afterendpar \relax
                  \else               \@ne    % inner, no X, outer in mode 1 or 2
                  \fi
                  \ifdefined\tabu@usetabu
                  \else \ifdim\tabu@target=\z@
                  \else \let\tabu@temp \tabu@extracolsep
                  \fi\fi
             \fi
        \fi
    \xdef\tabu@mkpreambuffer{\count@ \the\count@ \tabu@mkpreambuffer}%
    \tabu@temp
}% \tabu@endrewrite
```

**`\tabu@extracolsep`** Inserts `\extracolsep` `\@flushglue` in front of the preamble, unless another value for `\extracolsep` has been specified.

`\@flushglue` is 0pt plus 1fil.

```
\def\tabu@extracolsep{\@defaultunits    \expandafter\let
    \expandafter\tabu@temp \expandafter=\the\@temptokena \relax\@nnil
    \ifx \tabu@temp\@sptoken
        \expandafter\tabu@gobblespace \expandafter\tabu@extracolsep
    \else
        \edef\tabu@temp{\noexpand\NC@find
            \if |\noexpand\tabu@temp        @%
            \else\if !\noexpand\tabu@temp   @%
            \else                           !%
            \fi\fi
            {\noexpand\extracolsep\noexpand\@flushglue}}%
    \fi
    \tabu@temp
}% \tabu@extrac@lsep
```

## 11.10 Implementing the strategy at the exit of the \@mkpream group

### \tabu@select

**\tabu@pream** Triggered **\aftergroup** by the rewritting of **\tabu@rewritefirst**.

The **\tabu@mkpreambuffer** macro is expanded twice: first it injects **\count@** (the strategy number) and **\tabu@nbcols**, and redefines itself.

Second – and only if measurements are necessary – it expands into the *trials group* to inject **\tabu@Xcoefs** (the coefficients of **X** columns), **\tabu@Xsum** (the sum of the absolute coefficients), **\tabu@target**, **\tabu@spreadtarget**, and **\tabu@vertical**, which is the number by which one have to increment the strategy number after step 2 (either 1: then a last measure is done for the vertical dimensions, or 255 then the strategy number is $> 3$ and **\tabu@strategy** orders to finish.)

**\tabu@longpream** This is the **long** version for **longtabu**: the material to collect until **\@preamble** is different !

```
\long\def\tabu@pream #1\@preamble {%
    \let\tabu@ \tabu@@  \tabu@mkpreambuffer     \tabu@aftergroupcleanup
    \NC@list\expandafter {\tabu@NC@list}%    in case of nesting...
    \ifdefined\tabu@usetabu \tabu@usetabu \tabu@target \z@ \fi
    \let\tabu@savedpreamble \@preamble
    \global\let\tabu@elapsedtime \relax
    \tabu@thebody ={#1\tabu@aftergroupcleanup}%
    \tabu@thebody =\expandafter{\the\expandafter\tabu@thebody
                                                \@preamble}%
    \edef\tabuthepreamble {\the\tabu@thebody}% ( no @ allowed for \scantokens )
    \tabu@select
}% \tabu@pream
\long\def\tabu@longpream #1\LT@bchunk #2\LT@bchunk{%
    \let\tabu@ \tabu@@  \tabu@mkpreambuffer     \tabu@aftergroupcleanup
    \NC@list\expandafter {\tabu@NC@list}%    in case of nesting...
    \let\tabu@savedpreamble \@preamble
    \global\let\tabu@elapsedtime \relax
    \tabu@thebody ={#1\LT@bchunk #2\tabu@aftergroupcleanup \LT@bchunk}%
    \edef\tabuthepreamble {\the\tabu@thebody}% ( no @ allowed for \scantokens )
    \tabu@select
}% \tabu@longpream
```

**\tabu@select** Here we check if trials are required or not: depending on the value of **\count@** (set at **\tabu@endrewrite**, and *injected* here by **\tabu@mkpreambuffer**), on **\iftabu@measuring** (nested trials).

When trials are required, **\tabu@select** give control to **\tabu@setstrategy** (to prepare the neutralisation of commands, save counters etc).

When trials are not required, we just have to expand **\tabuthepreamble**, after having set up the \everyrow stuff properly (for vertical adjustment or vertical measure, if needed).

```
\def\tabu@select {%
    \ifnum\tabu@nested>\z@ \tabuscantokensfalse \fi
    \ifnum \count@=\@ne \iftabu@measuring \count@=\tw@ \fi\fi
    \ifcase \count@
```

```
        \global\let\tabu@elapsedtime \relax
        \tabu@seteverycr
        \expandafter \tabuthepreamble       % vertical adjustment (inheritated from outer)
    \or      % exit in vertical measure + struts per cell because no X and outer in mode 3
        \tabu@evr{\tabu@verticalinit}\tabu@celllalign@def{\tabu@verticalmeasure}%
        \def\tabu@cellralign{\tabu@verticalspacing}%
        \tabu@seteverycr
        \expandafter \tabuthepreamble
    \or                            % exit without measure because no X and outer in mode 4
        \tabu@evr{}\tabu@celllalign@def{}\let\tabu@cellralign \@empty
        \tabu@seteverycr
        \expandafter \tabuthepreamble
    \else                                   % needs trials
        \tabu@evr{}\tabu@celllalign@def{}\let\tabu@cellralign \@empty
        \tabu@savecounters
        \expandafter \tabu@setstrategy
    \fi
}% \tabu@select
\def\tabu@@ {\gdef\tabu@mkpreambuffer}
```

## General setup for trials: neutralisation of \write etc.

**\tabu@setstrategy** This is the general setup for trials: the **tabu** will be expanded more than once, thus some protections are set: the value of global counters are saved, footnotes have a special setup, **\hbadness** and **\hfuzz** are neutralised etc.

The initial value for **\tabucolX** is computed with the coefficients stored into **\tabu@Wvoefs**:

**\tabu@ {coef1} \tabu@ {coef2} \tabu@ {coef3}** etc.

is very suitable for loops on the column width coefficients (without the need of **\@for** or whatsoever).

```
\def\tabu@setstrategy {\begingroup  % <trials group>
    \tabu@trialh@@k    \tabu@cnt    \z@  % number of trials
    \hbadness          \@M          \let\hbadness          \@tempcnta
    \hfuzz             \maxdimen    \let\hfuzz             \@tempdima
    \let\write         \tabu@nowrite\let\GenericError      \tabu@GenericError
    \let\savetabu      \@gobble     \let\tabudefaulttarget \linewidth
    \let\@footnotetext \@gobble     \let\@xfootnote        \tabu@xfootnote
    \let\color         \tabu@nocolor\let\rowcolor          \tabu@norowcolor
    \let\tabu@aftergroupcleanup \relax % only after the last trial
    \tabu@mkpreambuffer
    \ifnum \count@>\thr@@ \let\@halignto \@empty  \tabucolX@init
                          \def\tabu@lasttry{\m@ne\p@}\fi
    \begingroup \iffalse{\fi \ifnum0=`}\fi
        \toks@{}\def\tabu@stack{b}\iftabuscantokens \endlinechar=10 \obeyspaces \fi %
                                  \tabu@collectbody \tabu@strategy %
}% \tabu@setstrategy
\def\tabu@savecounters{%
    \def\@elt ##1{\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \edef\tabu@clckpt {\begingroup \globaldefs=\@ne \cl@@ckpt \endgroup}\let\@elt \relax
}% \tabu@savecounters
\def\tabucolX@init {%  \tabucolX <= \tabu@target / (sum coefs > 0)
    \dimen@ \z@ \tabu@Xsum \z@ \tabucolX \z@ \let\tabu@ \tabu@Xinit \tabu@Xcoefs
    \ifdim \dimen@>\z@
        \@tempdima \dimexpr \tabu@target *\p@/\dimen@ + \tabu@hfuzz\relax
        \ifdim \tabucolX<\@tempdima \tabucolX \@tempdima \fi
    \fi
}% \tabucolX@init
\def\tabu@Xinit #1#2{\tabu@Xcol #1 \advance \tabu@Xsum
    \ifdim #2\p@>\z@ #2\p@  \advance\dimen@ #2\p@
    \else           -#2\p@  \tabu@negcoeftrue
                            \@tempdima \dimexpr \tabu@target*\p@/\dimexpr-#2\p@\relax \relax
                            \ifdim \tabucolX<\@tempdima \tabucolX \@tempdima \fi
                            \tabu@wddef{#1}{0pt}%
    \fi
}% \tabu@Xinit
```

### Collecting the tabu body

The macro collect the stuff inside `\@array`: depending on the global vertical alignment parameter for the whole tabular, the tabular is built inside a `\vbox`, `\vtop` or `\vcenter` (the default – unless `tabu` is nested).

At this time, we define `\tabu@trial` (which inherits from the `\vbox`, `\vtop` or `\vcenter`) and `\tabu@Xfinish` as well.

**\tabu@collectbody**
**\tabu@endofcollect**

The mechanism is the same as $\mathcal{AMS}$-`\collect@body` (also defined in `environ.sty`). The content of the tabular is captured inside `\toks@`, expanded by `\tabu@trial`.

```
\long\def\tabu@collectbody #1#2\end #3{%
    \edef\tabu@stack{\tabu@pushbegins #2\begin\end\expandafter\@gobble\tabu@stack}%
    \ifx \tabu@stack\@empty
        \toks@\expandafter{\expandafter\tabu@thebody\expandafter{\the\toks@ #2}%
                \def\tabu@end@envir{\end{#3}}%
                \iftabuscantokens
                    \iftabu@long \def\tabu@endenvir {\end{#3}\tabu@gobbleX}%
                    \else        \def\tabu@endenvir {\let\endarray \@empty
                                                     \end{#3}\tabu@gobbleX}%
                    \fi
                \else           \def\tabu@endenvir  {\end{#3}}\fi}%
        \let\tabu@collectbody \tabu@endofcollect
    \else\def\tabu@temp{#3}%
        \ifx \tabu@temp\@empty \toks@\expandafter{\the\toks@ #2\end }%
        \else \ifx\tabu@temp\tabu@@spxiii \toks@\expandafter{\the\toks@ #2\end #3}%
        \else \ifx\tabu@temp\tabu@X \toks@\expandafter{\the\toks@ #2\end #3}%
        \else \toks@\expandafter{\the\toks@ #2\end{#3}}%
        \fi\fi\fi
    \fi
    \tabu@collectbody{#1}%
}% \tabu@collectbody
\long\def\tabu@pushbegins#1\begin#2{\ifx\end#2\else b\expandafter\tabu@pushbegins\fi}%
\def\tabu@endofcollect #1{\ifnum0=`{}\fi
                          \expandafter\endgroup \the\toks@  #1%
}% \tabu@endofcollect
```

## 11.11 One trial after the other (\tabu@strategy)

### Switching between the strategies

**\tabu@strategy**

This macro does some specific setup depending on the strategy (1, 2 or 3), and orders to finish when all measurements are done.

This consists in a switch (`\ifcase`) which is done before the trials by `\tabu@strategy`, and after the trials by `\tabu@endtrial`.

```
\def\tabu@strategy {\relax  % stops \count@ assignment !
    \ifcase\count@          % case 0 = print with vertical adjustment (outer is finished)
        \expandafter \tabu@endoftrials
    \or                     % case 1 = exit in vertical measure (outer in mode 3)
        \expandafter\xdef\csname tabu@\the\tabu@nested.T\endcsname{\the\tabu@target}%
        \expandafter\xdef\csname tabu@\the\tabu@nested.X\endcsname{\the\tabucolX}%
        \expandafter \tabu@endoftrials
    \or                     % case 2 = exit with a rule replacing the table (outer in mode 4)
```

```
        \expandafter \tabu@quickend
    \or                     % case 3 = outer is in mode 3 because of no X
        \begingroup
            \tabu@evr{\tabu@verticalinit}\tabu@celllalign@def{\tabu@verticalmeasure}%
            \def\tabu@cellralign{\tabu@verticalspacing}%
            \expandafter \tabu@measuring
    \else                   % case 4 = horizontal measure
        \begingroup
            \global\let\tabu@elapsedtime \tabu@message@etime
            \long\def\multicolumn##1##2##3{\multispan{##1}}%
            \let\tabu@startpboxORI \@startpbox
            \iftabu@spread
                    \def\tabu@naturalXmax {\z@}%
                    \let\tabu@naturalXmin \tabu@naturalXmax
                    \tabu@evr{\global\tabu@naturalX \z@}%
                    \let\@startpbox \tabu@startpboxmeasure
            \else\iftabu@negcoef
                    \let\@startpbox \tabu@startpboxmeasure
            \else   \let\@startpbox \tabu@startpboxquick
            \fi\fi
            \expandafter \tabu@measuring
    \fi
}% \tabu@strategy
```

**`\tabu@measuring`** Expands `\tabu@trial` with the whole content of the environment stored in `\toks@` by `\tabu@collectbody`.

At the end of the trial, `\count@` will be reassigned to the value it had before the trial. Then `\tabu@endtrial` will choose the algorithm depending on the strategy number, and set the new strategy number (into `\count@` again) for the next step.

**`\tabu@trial`** This is the starting point of trials: `\halign` is expanded here.

**`\tabu@longtrial`** This is the long version of `\tabu@trial` for `longtabu`. Almost the same apart for the math group and the end (a `longtable` environment does not finish with `\endarray`).

```
\def\tabu@measuring{\expandafter \tabu@trial \expandafter
                                                \count@ \the\count@ \tabu@endtrial
}% \tabu@measuring
\def\tabu@trial{\iftabu@long \tabu@longtrial \else \tabu@shorttrial \fi}
\def\tabu@shorttrial {\setbox\tabu@box \hbox\bgroup \tabu@seteverycr
    \ifx \tabu@savecounters\relax \else
                \let\tabu@savecounters \relax \tabu@clckpt \fi
    $\iftabuscantokens \tabu@rescan \else \expandafter\@secondoftwo \fi
       \expandafter{\expandafter \tabuthepreamble
                         \the\tabu@thebody
                         \csname tabu@adl@endtrial\endcsname
                         \endarray}$\egroup             % got \tabu@box
}% \tabu@shorttrial
\def\tabu@longtrial {\setbox\tabu@box \hbox\bgroup \tabu@seteverycr
    \ifx \tabu@savecounters\relax \else
                \let\tabu@savecounters \relax \tabu@clckpt \fi
    \iftabuscantokens \tabu@rescan \else \expandafter\@secondoftwo \fi
       \expandafter{\expandafter \tabuthepreamble
                         \the\tabu@thebody
                         \tabuendlongtrial}\egroup       % got \tabu@box
}% \tabu@longtrial
\def\tabuendlongtrial{% no @ allowed for \scantokens
    \LT@echunk  \global\setbox\@ne \hbox{\unhbox\@ne}\kern\wd\@ne
                \LT@get@widths
}% \tabuendlongtrial
```

```
\def\tabu@adl@endtrial{% <arydshln in nested trials - problem for global column counters!>
    \crcr \noalign{\global\adl@ncol \tabu@nbcols}}% anything global is crap, junky and fails !
```

**\tabu@seteverycr** **\ialign** resets **\everycr** to an empty token. This macro sets **\everycr** for the **tabu** environment : a *bridge* around **\ialign** is built: **\everycr** redefines itself **\afterassignment**!

```
\def\tabu@seteverycr {\tabu@reset
    \everycr \expandafter{\the\everycr  \tabu@everycr}%
    \let\everycr \tabu@noeverycr                        % <for ialign>
}% \tabu@seteverycr
\def\tabu@noeverycr{{\aftergroup\tabu@restoreeverycr \afterassignment}\toks@}
\def\tabu@restoreeverycr {\let\everycr \tabu@@everycr}
\def\tabu@everycr {\iftabu@everyrow \noalign{\tabu@everyrow}\fi}
```

**\tabu@endoftrials** When the algorithm said the tabular was ready to be printed, **\tabu@endoftrials** closes the trials group and prints the tabular...

The required values (column widths, struts etc.) are *injected* into the group by the mean of the buffer **\tabu@bufferX** (locally defined).

**\tabu@closetrialsgroup** This closes the group in which all the trials are done.

```
\def\tabu@endoftrials {%
    \iftabuscantokens    \expandafter\@firstoftwo
    \else                \expandafter\@secondoftwo
    \fi
        {\expandafter \tabu@closetrialsgroup \expandafter
         \tabu@rescan \expandafter{%
                     \expandafter\tabuthepreamble
                         \the\expandafter\tabu@thebody
                                     \iftabu@long \else \endarray \fi}}
        {\expandafter\tabu@closetrialsgroup \expandafter
                     \tabuthepreamble
                         \the\tabu@thebody}%
                                         \tabu@endenvir       % Finish !
}% \tabu@endoftrials
\def\tabu@closetrialsgroup {%
    \toks@\expandafter{\tabu@endenvir}%
    \edef\tabu@bufferX{\endgroup
        \tabucolX        \the\tabucolX
        \tabu@target     \the\tabu@target
        \tabu@cnt        \the\tabu@cnt
        \def\noexpand\tabu@endenvir{\the\toks@}%
        %Quid de \@halignto = \tabu@halignto ??
    }% \tabu@bufferX
    \tabu@bufferX
    \ifcase\tabu@nested % print out (outer in mode 0)
        \global\tabu@cnt \tabu@cnt
        \tabu@evr{\tabu@verticaldynamicadjustment}%
        \tabu@celllalign@def{\everypar{}}\let\tabu@cellralign \@empty
        \let\@finalstrut \tabu@finalstrut
    \else                % vertical measure of nested tabu
        \tabu@evr{\tabu@verticalinit}%
        \tabu@celllalign@def{\tabu@verticalmeasure}%
        \def\tabu@cellralign{\tabu@verticalspacing}%
    \fi
    \tabu@clckpt \let\@halignto \tabu@halignto
    \let\@halignto \@empty
    \tabu@seteverycr
    \ifdim \tabustrutrule>\z@ \ifnum\tabu@nested=\z@
        \setbox\@arstrutbox \box\voidb@x % force \@arstrutbox to be rebuilt (visible struts)
    \fi\fi
}% \tabu@closetrialsgroup
```

**\tabu@quickend** Quick exit after having measuring the natural width of a nested **tabu**.

```
\def\tabu@quickend {\expandafter \endgroup \expandafter
                        \tabu@target \the\tabu@target \tabu@quickrule
                        \let\endarray \relax \tabu@endenvir
}% \tabu@quickend
```

**\tabu@endtrial** Depending on the strategy that was just applied, **\tabu@endtrial** chooses the algorithm and determines the number of the strategy for the next step.

```
\def\tabu@endtrial {\relax      % stops \count@ assignment !
    \ifcase \count@ \tabu@err   % case 0 = impossible here
    \or             \tabu@err   % case 1 = impossible here
    \or             \tabu@err   % case 2 = impossible here
    \or                         % case 3 = outer goes into mode 0
        \def\tabu@bufferX{\endgroup}\count@ \z@
    \else                       % case 4 = outer goes into mode 3
        \iftabu@spread  \tabu@spreadarith % inner into mode 1 (outer in mode 3)
        \else           \tabu@arith       %               or 2 (outer in mode 4)
        \fi
        \count@=%
            \ifcase\tabu@nested     \thr@@  % outer goes into mode 3
            \else\iftabu@measuring  \tw@    % outer is in mode 4
            \else                   \@ne    % outer is in mode 3
            \fi\fi
        \edef\tabu@bufferX{\endgroup
                           \tabucolX        \the\tabucolX
                           \tabu@target     \the\tabu@target}%
    \fi
    \expandafter \tabu@bufferX \expandafter
                                    \count@ \the\count@  \tabu@strategy
}% \tabu@endtrial
\def\tabu@err{\errmessage{(tabu) Internal impossible error! (\count@=\the\count@)}}
```

## 11.12 The algorithms: Measuring the tabu box

At the end of each trial, we call **\tabu@arith** (or **\tabu@spreadarith**) to computes the widths and update the values.

At the exit, **\iftabu@measuring** is set to **\iftrue**: a further trial is necessary, or **\iffalse**: the target width is reached.

### The arithmetic of X columns: the tabu to case

**\tabu@arithnegcoef** This is a loop against the width coefficients. There is no **\@for** or **\@whiles** because **\tabu@Xcoefs** stores the series in the form: `\tabu@ {coef1} \tabu@ {coef2} \tabu@ {coef3}`.

Thus, just **\let \tabu@** to be **\tabu@arith@negcoef** and expand **\tabu@Xcoefs**!

The aim of the game is to *neutralize* some X columns: when their natural width are less than coef×**\tabucolX**.

```
\def\tabu@arithnegcoef {%
    \@tempdima \z@ \dimen@ \z@ \let\tabu@ \tabu@arith@negcoef \tabu@Xcoefs
}% \tabu@arithnegcoef
\def\tabu@arith@negcoef #1#2{%
    \ifdim #2\p@>\z@    \advance\dimen@     #2\p@       % saturated by definition
                        \advance\@tempdima  #2\tabucolX
    \else
        \ifdim -#2\tabucolX <\tabu@wd{#1}% c_i X < natural width <= \tabu@target-> saturated
                        \advance\dimen@     -#2\p@
                        \advance\@tempdima  -#2\tabucolX
        \else
                        \advance\@tempdima \tabu@wd{#1}% natural width <= c_i X => neutralised
                        \ifdim \tabu@wd{#1}<\tabu@target \else % neutralised
                        \advance\dimen@     -#2\p@ % saturated (natural width = tabu@target)
                        \fi
        \fi
    \fi
}% \tabu@arith@negcoef
```

**`\tabu@arith`** General algorithms for **tabu to** with **X** columns.

```
\def\tabu@givespace #1#2{% here \tabu@DELTA < \z@
    \ifdim \@tempdima=\z@
        \tabu@wddef{#1}{\the\dimexpr -\tabu@DELTA*\p@/\tabu@Xsum}%
    \else
        \tabu@wddef{#1}{\the\dimexpr \tabu@hsize{#1}{#2}
                    *(\p@ -\tabu@DELTA*\p@/\@tempdima)/\p@\relax}%
    \fi
}% \tabu@givespace
\def\tabu@arith {\advance\tabu@cnt \@ne
    \ifnum \tabu@cnt=\@ne \tabu@message{\tabu@titles}\fi
    \tabu@arithnegcoef
    \@tempdimb \dimexpr \wd\tabu@box -\@tempdima \relax % <incompressible material>
    \tabu@DELTA = \dimexpr \wd\tabu@box - \tabu@target \relax
    \tabu@message{\tabu@message@arith}%
    \ifdim \tabu@DELTA <\tabu@hfuzz
        \ifdim \tabu@DELTA<\z@          % wd (tabu)<\tabu@target ?
            \let\tabu@ \tabu@givespace \tabu@Xcoefs
            \advance\@tempdima \@tempdimb \advance\@tempdima -\tabu@DELTA % for message
        \else   % already converged: nothing to do but nearly impossible...
        \fi
        \tabucolX \maxdimen
        \tabu@measuringfalse
    \else                               % need for narrower X columns
        \tabucolX =\dimexpr (\@tempdima -\tabu@DELTA) *\p@/\tabu@Xsum \relax
        \tabu@measuringtrue
        \@whilesw \iftabu@measuring\fi {%
            \advance\tabu@cnt \@ne
            \tabu@arithnegcoef
            \tabu@DELTA =\dimexpr \@tempdima+\@tempdimb -\tabu@target \relax % always < 0 here
            \tabu@message{\tabu@header
                \tabu@msgalign \tabucolX { }{ }{ }{ }{ }\@@
                \tabu@msgalign \@tempdima+\@tempdimb { }{ }{ }{ }{ }\@@
                \tabu@msgalign \tabu@target { }{ }{ }{ }{ }\@@
                \tabu@msgalign@PT \dimen@ { }{}{}{}{}{}{}\@@
                \ifdim -\tabu@DELTA<\tabu@hfuzz \tabu@spaces target ok\else
                \tabu@msgalign \dimexpr -\tabu@DELTA *\p@/\dimen@ {}{}{}{}{}\@@
                \fi}%
            \ifdim -\tabu@DELTA<\tabu@hfuzz
                \advance\@tempdima \@tempdimb % for message
                \tabu@measuringfalse
            \else
                \advance\tabucolX \dimexpr -\tabu@DELTA *\p@/\dimen@ \relax
            \fi
        }%
    \fi
    \tabu@message{\tabu@message@reached}%
    \edef\tabu@bufferX{\endgroup \tabu@cnt    \the\tabu@cnt
```

```
                                      \tabucolX    \the\tabucolX
                                      \tabu@target \the\tabu@target}%
}% \tabu@arith
```

## The arithmetic of X columns for tabu spread

**\tabu@spreadarith** Algorithm for tabu spread with X columns: the aim of the game is to compute the target (relative to the natural width of the tabular) and go to **\tabu@arith** afterwards.

```
\def\tabu@spreadarith {%
    \dimen@ \z@ \@tempdima \tabu@naturalXmax \let\tabu@ \tabu@spread@arith \tabu@Xcoefs
    \edef\tabu@naturalXmin {\the\dimexpr\tabu@naturalXmin*\dimen@/\p@}%
    \@tempdimc =\dimexpr \wd\tabu@box -\tabu@naturalXmax+\tabu@naturalXmin \relax
    \iftabu@measuring
        \tabu@target =\dimexpr \@tempdimc+\tabu@spreadtarget \relax
        \edef\tabu@bufferX{\endgroup \tabucolX \the\tabucolX \tabu@target\the\tabu@target}%
    \else
        \tabu@message{\tabu@message@spreadarith}%
        \ifdim \dimexpr \@tempdimc+\tabu@spreadtarget >\tabu@target
            \tabu@message{(tabu) spread
                \ifdim \@tempdimc>\tabu@target useless here: default target used%
                \else too large: reduced to fit default target\fi.}%
        \else
            \tabu@target =\dimexpr \@tempdimc+\tabu@spreadtarget \relax
            \tabu@message{(tabu) spread: New target set to \the\tabu@target^^J}%
        \fi
        \begingroup \let\tabu@wddef \@gobbletwo
            \@tempdimb \@tempdima
            \tabucolX@init
            \tabu@arithnegcoef
            \wd\tabu@box =\dimexpr \wd\tabu@box +\@tempdima-\@tempdimb \relax
        \expandafter\endgroup \expandafter\tabucolX \the\tabucolX
        \tabu@arith
    \fi
}% \tabu@spreadarith
\def\tabu@spread@arith #1#2{%
    \ifdim #2\p@>\z@ \advance\dimen@ #2\p@
    \else            \advance\@tempdima \tabu@wd{#1}\relax
    \fi
}% \tabu@spread@arith
```

## Reporting in the .log file (debugshow option)

**\tabu@message@defaulttarget**

```
\def\tabu@message@defaulttarget{%
    \ifnum\tabu@nested=\z@^^J(tabu) Default target:
    \ifx\tabudefaulttarget\linewidth    \string\linewidth
        \ifdim \tabu@thetarget=\linewidth \else
            -\the\dimexpr\linewidth-\tabu@thetarget\fi  =
    \else\ifx\tabudefaulttarget\linegoal\string\linegoal=
    \fi\fi
    \else (tabu) Default target (nested): \fi
    \the\tabu@target \on@line
    \ifnum\tabu@nested=\z@ , page \the\c@page\fi}
\def\tabu@message@target {^^J(tabu) Target specified:
   \the\tabu@target \on@line, page \the\c@page}
```

**\tabu@message@arith**

```
\def\tabu@message@arith {\tabu@header
```

```
    \tabu@msgalign \tabucolX { }{ }{ }{ }{ }\@@
    \tabu@msgalign \wd\tabu@box { }{ }{ }{ }{ }\@@
    \tabu@msgalign \tabu@target { }{ }{ }{ }{ }\@@
    \tabu@msgalign@PT \dimen@ { }{}{}{}{}{}{}\@@
    \ifdim \tabu@DELTA<\tabu@hfuzz giving space\else
    \tabu@msgalign \dimexpr (\@tempdima-\tabu@DELTA) *\p@/\tabu@Xsum -\tabucolX {}{}{}{}{}\@@
    \fi
}% \tabu@message@arith
```

**\tabu@message@spreadarith**

```
\def\tabu@message@spreadarith {\tabu@spreadheader
   \tabu@msgalign \tabu@spreadtarget { }{ }{ }{ }{}\@@
   \tabu@msgalign \wd\tabu@box { }{ }{ }{ }{}\@@
   \tabu@msgalign -\tabu@naturalXmax { }{}{}{}{}\@@
   \tabu@msgalign \tabu@naturalXmin { }{ }{ }{ }{}\@@
   \tabu@msgalign \ifdim \dimexpr\@tempdimc>\tabu@target \tabu@target
                  \else  \@tempdimc+\tabu@spreadtarget \fi
                  {}{}{}{}{}\@@}
```

**\tabu@message@negcoef**

```
\def\tabu@message@negcoef #1#2{
    \tabu@spaces\tabu@spaces\space * #1. X[\rem@pt#2]:
    \space width = \tabu@wd {#1}
        \expandafter\string\csname tabu@\the\tabu@nested.W\number#1\endcsname
    \ifdim -\tabu@pt#2\tabucolX<\tabu@target
    < \number-\rem@pt#2 X
    = \the\dimexpr -\tabu@pt#2\tabucolX \relax
    \else
    <= \the\tabu@target\space < \number-\rem@pt#2 X\fi}
```

**\tabu@message@reached**

```
\def\tabu@message@reached{\tabu@header
     ******* Reached Target:
            hfuzz = \tabu@hfuzz\on@line\space *******}
```

**\tabu@message@etime**

```
\def\tabu@message@etime{\edef\tabu@stoptime{\the\pdfelapsedtime}%
    \tabu@message{(tabu)\tabu@spaces Time elapsed during measure:
    \the\numexpr(\tabu@stoptime-\tabu@starttime-32767)/65536\relax sec
    \the\numexpr\numexpr(\tabu@stoptime-\tabu@starttime)
    -\numexpr(\tabu@stoptime-\tabu@starttime-32767)/65536\relax*65536\relax
    *1000/65536\relax ms \tabu@spaces(\the\tabu@cnt\space
                                        cycle\ifnum\tabu@cnt>\@ne s\fi)^^J^^J}}
```

**\tabu@message@verticalsp**

```
\def\tabu@message@verticalsp {%
    \ifdim \@tempdima>\tabu@ht
        \ifdim \@tempdimb>\tabu@dp
        \expandafter\expandafter\expandafter\string\tabu@ht =
            \tabu@msgalign \@tempdima { }{ }{ }{ }{ }\@@
        \expandafter\expandafter\expandafter\string\tabu@dp =
            \tabu@msgalign \@tempdimb { }{ }{ }{ }{ }\@@^^J%
        \else
        \expandafter\expandafter\expandafter\string\tabu@ht =
            \tabu@msgalign \@tempdima { }{ }{ }{ }{ }\@@^^J%
        \fi
    \else\ifdim \@tempdimb>\tabu@dp
        \tabu@spaces\tabu@spaces\tabu@spaces
        \expandafter\expandafter\expandafter\string\tabu@dp =
            \tabu@msgalign \@tempdimb { }{ }{ }{ }{ }\@@^^J\fi
    \fi
}% \tabu@message@verticalsp
```

**\tabu@message@save**

```
\edef\tabu@spaces{\@spaces}
\def\tabu@strippt{\expandafter\tabu@pt\the}
{\@makeother\P \@makeother\T\lowercase{\gdef\tabu@pt #1PT{#1}}}
\def\tabu@msgalign{\expandafter\tabu@msg@align\the\dimexpr}
\def\tabu@msgalign@PT{\expandafter\tabu@msg@align\romannumeral-`\0\tabu@strippt}
\def\do #1{%
    \def\tabu@msg@align##1.##2##3##4##5##6##7##8##9\@@{%
    \ifnum##1<10 #1 #1\else
    \ifnum##1<100 #1 \else
    \ifnum##1<\@m #1\fi\fi\fi
    ##1.##2##3##4##5##6##7##8#1}%
    \def\tabu@header{(tabu) \ifnum\tabu@cnt<10 #1\fi\the\tabu@cnt) }%
    \def\tabu@titles{\ifnum \tabu@nested=\z@
      (tabu) Try#1 #1 tabu X #1 #1 #1tabu Width #1 #1 Target
                  #1 #1 #1 Coefs #1 #1 #1 Update^^J\fi}%
    \def\tabu@spreadheader{%
      (tabu) Try#1 #1 Spread #1 #1 tabu Width #1 #1 #1 Nat. X #1 #1 #1 #1Nat. Min.
                                                       #1 New Target^^J%
      (tabu) sprd}
    \def\tabu@message@save {\begingroup
        \def\x ####1{\tabu@msg@align ####1{ }{ }{ }{ }{}\@@}
        \def\z ####1{\expandafter\x\expandafter{\romannumeral-`\0\tabu@strippt
                                                      \dimexpr####1\p@{ }{ }}}%
        \let\color \relax \def\tabu@rulesstyle ####1####2{\detokenize{####1}}%
        \let\CT@arc@ \relax \let\@preamble \@gobble
        \let\tabu@savedpream  \@firstofone
        \let\tabu@savedparams \@firstofone
        \def\tabu@target ####1\relax    {(tabu) target #1 #1 #1 #1 #1 = \x{####1}^^J}%
        \def\tabucolX ####1\relax      {(tabu) X columns width#1 = \x{####1}^^J}%
        \def\tabu@nbcols ####1\relax    {(tabu) Number of columns: \z{####1}^^J}%
        \def\tabu@aligndefault    ####1{(tabu) Default alignment: #1 #1 ####1^^J}%
        \def\col@sep ####1\relax       {(tabu) column sep #1 #1 #1 = \x{####1}^^J}%
        \def\arrayrulewidth ####1\relax{(tabu) arrayrulewidth #1 = \x{####1}}%
        \def\doublerulesep ####1\relax { doublerulesep = \x{####1}^^J}%
        \def\extratabsurround####1\relax{(tabu) extratabsurround = \x{####1}^^J}%
        \def\extrarowheight ####1\relax{(tabu) extrarowheight #1 = \x{####1}}%
        \def\extrarowdepth ####1\relax {extrarowdepth = \x{####1}^^J}%
        \def\abovetabulinesep####1\relax{(tabu) abovetabulinesep=\x{####1} }%
        \def\belowtabulinesep####1\relax{ belowtabulinesep=\x{####1}^^J}%
        \def\arraystretch         ####1{(tabu) arraystretch #1 #1 = \z{####1}^^J}%
        \def\minrowclearance####1\relax{(tabu) minrowclearance #1 = \x{####1}^^J}%
        \def\tabu@arc@L           ####1{(tabu) taburulecolor #1 #1 = ####1^^J}%
        \def\tabu@drsc@L          ####1{(tabu) tabudoublerulecolor=  ####1^^J}%
        \def\tabu@evr@L          ####1{(tabu) everyrow #1 #1 #1 #1 = \detokenize{####1}^^J}%
        \def\tabu@ls@L            ####1{(tabu) line style = \detokenize{####1}^^J}%
        \def\NC@find ####1\@nil{(tabu) tabu preamble#1 #1 = \detokenize{####1}^^J}%
        \def\tabu@wddef####1####2{(tabu) Natural width ####1 = \x{####2}^^J}%
        \let\edef \@gobbletwo \let\def \@empty \let\let \@gobbletwo
        \tabu@message{%
         (tabu) \string\savetabu{\tabu@temp}: \on@line^^J%
         \tabu@usetabu \@nil^^J}%
        \endgroup}
```

```
}\do{ }
```

## 11.13 Measuring the natural width of columns (**varwidth** code from D. Arseneau)

**`\tabu@startpboxmeasure`** The important job is done at the end: by `\tabu@endpboxmeasure`.

When "`tabu spread`" is used with `X` columns, the first trial must measure the natural width of the columns. When `X` columns have negativ coefficient, the natural is computed after the target has been reached, with the absolute coefficients.

Nested trials may occur (`tabu spread` inside a `X` column with negativ coefficient for example).

For the furthur trials, the standard scheme for `X` column is used: the natural width is measured only once.

pdfTEX font expansion is disabled inside the `varwidth` environment (we set `\pdfadjustspacing` to `0`).

```
\def\tabu@startpboxmeasure #1{\bgroup   % entering \vtop
    \edef\tabu@temp{\expandafter\@secondoftwo \ifx\tabu@hsize #1\else\relax\fi}%
    \ifodd 1\ifx \tabu@temp\@empty 0 \else      % starts with \tabu@hsize ?
            \iftabu@spread           \else      % if spread -> measure
            \ifdim \tabu@temp\p@>\z@ 0 \fi\fi\fi% if coef>0 -> do not measure
        \let\@startpbox \tabu@startpboxORI      % restore immediately (nesting)
        \tabu@measuringtrue                     % for the quick option...
        \tabu@Xcol =\expandafter\@firstoftwo\ifx\tabu@hsize #1\fi
        \ifdim \tabu@temp\p@>\z@ \ifdim \tabu@temp\tabucolX<\tabu@target
                                        \tabu@target=\tabu@temp\tabucolX \fi\fi
        \setbox\tabu@box  \hbox \bgroup
            \begin{varwidth}\tabu@target
                \let\FV@ListProcessLine \tabu@FV@ListProcessLine  % \hbox to natural width...
                \narrowragged \arraybackslash \parfillskip \@flushglue
                \ifdefined\pdfadjustspacing \pdfadjustspacing\z@ \fi
                \bgroup \aftergroup\tabu@endpboxmeasure
                \ifdefined \cellspacetoplimit \tabu@cellspacepatch \fi
    \else \expandafter\@gobble
                            \tabu@startpboxquick{#1}% \@gobble \bgroup
    \fi
}% \tabu@startpboxmeasure
\def\tabu@cellspacepatch{\def\bcolumn##1\@nil{}\let\ecolumn\@empty
                                          \bgroup\color@begingroup}
```

**`\tabu@endpboxmeasure`** The cell has been built inside a box: we have to get its dimensions, and update `\tabu@naturalX`, `\tabu@naturalXmin` and `\tabu@naturalXmax` accordingly (for `tabu spread`), and even store (globally) each column width: the column width is the maximum width of the cells it contains.

```
\def\tabu@endpboxmeasure {%
    \@finalstrut \@arstrutbox
                    \end{varwidth}\egroup    % <got my \tabu@box>
    \ifdim \tabu@temp\p@ <\z@   % neg coef
        \ifdim \tabu@wd\tabu@Xcol <\wd\tabu@box
            \tabu@wddef\tabu@Xcol {\the\wd\tabu@box}%
            \tabu@debug{\tabu@message@endpboxmeasure}%
        \fi
    \else                       % spread coef>0
        \global\advance \tabu@naturalX \wd\tabu@box
        \@tempdima =\dimexpr \wd\tabu@box *\p@/\dimexpr \tabu@temp\p@\relax \relax
        \ifdim \tabu@naturalXmax <\tabu@naturalX
            \xdef\tabu@naturalXmax {\the\tabu@naturalX}\fi
        \ifdim \tabu@naturalXmin <\@tempdima
            \xdef\tabu@naturalXmin {\the\@tempdima}\fi
    \fi
   \box\tabu@box \egroup % end of \vtop (measure) restore \tabu@target
}% \tabu@endpboxmeasure
\def\tabu@wddef #1{\expandafter\xdef
                   \csname tabu@\the\tabu@nested.W\number#1\endcsname}
\def\tabu@wd    #1{\csname tabu@\the\tabu@nested.W\number#1\endcsname}
\def\tabu@message@endpboxmeasure{\tabu@spaces\tabu@spaces<-> % <-> save natural wd
    \the\tabu@Xcol. X[\tabu@temp]:
    target = \the\tabucolX \space
    \expandafter\expandafter\expandafter\string\tabu@wd\tabu@Xcol
    =\tabu@wd\tabu@Xcol
}% \tabu@message@endpboxmeasure
```

**\tabu@startpboxquick** Contents of paragraph columns are not built during trials in strategy number 4.

```
\def\tabu@startpboxquick {\bgroup
    \let\@startpbox \tabu@startpboxORI  % restore immediately
    \let\tabu \tabu@quick               % \begin is expanded before...
    \expandafter\@gobble \@startpbox    % gobbles \bgroup
}% \tabu@startpboxquick
\def\tabu@quick {\begingroup \iffalse{\fi \ifnum0=`}\fi
    \toks@{}\def\tabu@stack{b}\tabu@collectbody \tabu@endquick
}% \tabu@quick
\def\tabu@endquick {%
    \ifodd 1\ifx\tabu@end@envir\tabu@endtabu  \else
            \ifx\tabu@end@envir\tabu@endtabus \else 0\fi\fi\relax
            \endgroup
    \else   \let\endtabu \relax
            \tabu@end@envir
    \fi
}% \tabu@quick
\def\tabu@endtabu   {\end{tabu}}
\def\tabu@endtabus  {\end{tabu*}}
```

## 11.14 Measuring the height and depths of rows

**\tabu@verticalmeasure** Starting point for vertical measure of every cell. Only the maxima/minima are stored, for Tℵb⊂ must know the height/depth of every row.

```
\def\tabu@verticalmeasure{\everypar{}%
    \ifnum \currentgrouptype>12         % 14=semi-simple, 15=math shift group
        \setbox\tabu@box =\hbox\bgroup
            \let\tabu@verticalspacing \tabu@verticalsp@lcr
            \d@llarbegin                % after \hbox ...
    \else
        \edef\tabu@temp{\ifnum\currentgrouptype=5\vtop
                        \else\ifnum\currentgrouptype=12\vcenter
                        \else\vbox\fi\fi}%
        \setbox\tabu@box \hbox\bgroup$\tabu@temp \bgroup
            \let\tabu@verticalspacing \tabu@verticalsp@pmb
    \fi
}% \tabu@verticalmeasure
```

**\tabu@verticalsp@lcr** Vertical spacing adjustment for standard `l`, `c`, `r` columns.

```
\def\tabu@verticalsp@lcr{%
    \d@llarend \egroup        % <got my \tabu@box>
    \@tempdima \dimexpr \ht\tabu@box+\abovetabulinesep
    \@tempdimb \dimexpr \dp\tabu@box+\belowtabulinesep \relax
        \ifdim\tabustrutrule>\z@ \tabu@debug{\tabu@message@verticalsp}\fi
    \ifdim \tabu@ht<\@tempdima     \tabu@htdef{\the\@tempdima}\fi
    \ifdim \tabu@dp<\@tempdimb     \tabu@dpdef{\the\@tempdimb}\fi
    \noindent\vrule height\@tempdima depth\@tempdimb
}% \tabu@verticalsp@lcr
```

**\tabu@verticalsp@pmb** Vertical spacing adjustment with struts for `p`, `m`, or `b` columns.

```
\def\tabu@verticalsp@pmb{% inserts struts as needed
    \par \expandafter\egroup
            \expandafter$\expandafter
                    \egroup \expandafter
                            \@tempdimc \the\prevdepth
    \@tempdima \dimexpr \ht\tabu@box+\abovetabulinesep
    \@tempdimb \dimexpr \dp\tabu@box+\belowtabulinesep \relax
        \ifdim\tabustrutrule>\z@ \tabu@debug{\tabu@message@verticalsp}\fi
    \ifdim \tabu@ht<\@tempdima    \tabu@htdef{\the\@tempdima}\fi
    \ifdim \tabu@dp<\@tempdimb    \tabu@dpdef{\the\@tempdimb}\fi
    \let\@finalstrut \@gobble
    \hrule height\@tempdima depth\@tempdimb width\hsize
%%    \box\tabu@box
}% \tabu@verticalsp@pmb
```

**\tabu@verticalinit** Initialisation of `\tabu@ht` and `\tabu@dp`. Done at \everyrow.

```
\def\tabu@verticalinit{%
    \ifnum \c@taburow=\z@ \tabu@rearstrut \fi       % after \tabu@reset !
    \advance\c@taburow \@ne
    \tabu@htdef{\the\ht\@arstrutbox}\tabu@dpdef{\the\dp\@arstrutbox}%
    \advance\c@taburow \m@ne
}% \tabu@verticalinit
\def\tabu@htdef {\expandafter\xdef \csname tabu@\the\tabu@nested.H\the\c@taburow\endcsname}
\def\tabu@ht                      {\csname tabu@\the\tabu@nested.H\the\c@taburow\endcsname}
\def\tabu@dpdef {\expandafter\xdef \csname tabu@\the\tabu@nested.D\the\c@taburow\endcsname}
\def\tabu@dp                      {\csname tabu@\the\tabu@nested.D\the\c@taburow\endcsname}
```

**\tabu@verticaldynamicadjustment** This updates the `\@arstrutbox` at \everyrow (*ie.*\everycr) in order to adjust the vertical spacing of cells.

```
\def\tabu@verticaldynamicadjustment {%
    \advance\c@taburow \@ne
        \extrarowheight \dimexpr\tabu@ht - \ht\strutbox
        \extrarowdepth  \dimexpr\tabu@dp - \dp\strutbox
        \let\arraystretch \@empty
    \advance\c@taburow \m@ne
}% \tabu@verticaldynamicadjustment
```

## 11.15 \tabuphantomline

**\tabuphantomline** This macro inserts a phantom line in front of a `tabu`. This is necessary when you use \usetabu with `tabu` `X` column, with a single line containing `\multicolumn`...

```
\def\tabuphantomline{\crcr \noalign{%
    {\globaldefs \@ne
        \setbox\@arstrutbox     \box\voidb@x
        \let\tabu@@celllalign   \tabu@celllalign
        \let\tabu@@cellralign   \tabu@cellralign
        \let\tabu@@cellleft     \tabu@cellleft
        \let\tabu@@cellright    \tabu@cellright
        \let\tabu@@thevline     \tabu@thevline
        \let\tabu@celllalign    \@empty
        \let\tabu@cellralign    \@empty
        \let\tabu@cellright     \@empty
        \let\tabu@cellleft      \@empty
        \let\tabu@thevline      \relax}%
    \edef\tabu@temp{\tabu@multispan \tabu@nbcols{\noindent &}}%
    \toks@\expandafter{\tabu@temp \noindent\tabu@everyrowfalse \cr
        \noalign{\tabu@rearstrut
            {\globaldefs\@ne
                \let\tabu@celllalign \tabu@@celllalign
                \let\tabu@cellralign \tabu@@cellralign
                \let\tabu@cellleft   \tabu@@cellleft
                \let\tabu@cellright  \tabu@@cellright
                \let\tabu@thevline   \tabu@@thevline}}}%
    \expandafter}\the\toks@
}% \tabuphantomline
```

## 11.16 Horizontal lines inside tabu: \tabucline, \firsthline and \lasthline

### Horizontal lines: multiple \firsthline / \lasthline

**\tabu@firstline**
**\tabu@lastline**
**\tabu@firsthline**
**\tabu@lasthline**

\firsthline and \lasthline are \let to \tabu@firsthline and \tabu@lasthline inside the tabu environment. This allows to duplicate horizontal lines, while keeping the alignement: \firsthline \firsthline \firsthline is allowed inside tabu and is the same as: \firsthline \hline \hline.

```
\def\tabu@firstline {\tabu@hlineAZ  \tabu@firsthlinecorrection     {}}
\def\tabu@firsthline{\tabu@hlineAZ  \tabu@firsthlinecorrection \hline}
\def\tabu@lastline  {\tabu@hlineAZ  \tabu@lasthlinecorrection      {}}
\def\tabu@lasthline {\tabu@hlineAZ  \tabu@lasthlinecorrection  \hline}
\def\tabu@hline {% replaces \hline if no colortbl (see \AtBeginDocument)
    \noalign{\ifnum0=`}\fi
    {\CT@arc@\hrule height\arrayrulewidth}%
    \futurelet \tabu@temp \tabu@xhline
}% \tabu@hline
\def\tabu@xhline{%
    \ifx \tabu@temp \hline
        {\ifx \CT@drsc@\relax \vskip
         \else\ifx \CT@drsc@\@empty \vskip
         \else \CT@drsc@\hrule height
         \fi\fi
         \doublerulesep}%
    \fi
    \ifnum0=`{\fi}%
}% \tabu@xhline
```

**\tabu@hlineAZ**
**\tabu@nexthlineAZ**
**\tabu@xhlineAZ**

Here we go, inside a \noalign group, we collect the next tokens:

1. first the option,
2. and then the next tokens if they are \hline or \firsthline.

The code to be executed at the end of the \noalign group is built into \toks@.

```
\def\tabu@hlineAZ #1#2{\noalign{\ifnum0=`}\fi \dimen@ \z@ \count@ \z@
    \toks@{}\def\tabu@hlinecorrection{#1}\def\tabu@temp{#2}%
    \tabu@hlineAZsurround
}% \tabu@hlineAZ
\newcommand*\tabu@hlineAZsurround[1][\extratabsurround]{%
    \extratabsurround #1\let\tabucline \tabucline@scan
    \let\hline     \tabu@hlinescan \let\firsthline \hline
    \let\cline     \tabu@clinescan \let\lasthline  \hline
    \expandafter \futurelet \expandafter \tabu@temp
                \expandafter \tabu@nexthlineAZ \tabu@temp
}% \tabu@hlineAZsurround
\def\tabu@hlinescan   {\tabu@thick \arrayrulewidth \tabu@xhlineAZ \hline}
```

```
\def\tabu@clinescan #1{\tabu@thick \arrayrulewidth \tabu@xhlineAZ {\cline{#1}}}
\def\tabucline@scan{\@testopt \tabucline@sc@n {}}
\def\tabucline@sc@n #1[#2]{\tabu@xhlineAZ {\tabucline[{#1}]{#2}}}
\def\tabu@nexthlineAZ{%
    \ifx \tabu@temp\hline \else
    \ifx \tabu@temp\cline \else
    \ifx \tabu@temp\tabucline \else
         \tabu@hlinecorrection
    \fi\fi\fi
}% \tabu@nexthlineAZ
\def\tabu@xhlineAZ #1{%
    \toks@\expandafter{\the\toks@ #1}%
    \@tempdimc \tabu@thick                  % The last line width
    \ifcase\count@ \@tempdimb \tabu@thick   % The first line width
    \else \advance\dimen@ \dimexpr \tabu@thick+\doublerulesep \relax
    \fi
    \advance\count@ \@ne    \futurelet \tabu@temp \tabu@nexthlineAZ
}% \tabu@xhlineAZ
```

**\tabu@firsthlinecorrection** This is the "correction macro" for \firsthline, *ie.*a strut and a skip are inserted **before** the first \hline.

```
\def\tabu@firsthlinecorrection{% \count@ = number of \hline -1
    \@tempdima \dimexpr \ht\@arstrutbox+\dimen@
    \edef\firsthline{%      <local in \noalign>
        \omit \hbox to\z@{\hss{\noexpand\tabu@DBG{yellow}\vrule
                    height \the\dimexpr\@tempdima+\extratabsurround
                    depth  \dp\@arstrutbox
                    width  \tabustrutrule}\hss}\cr
        \noalign{\vskip -\the\dimexpr   \@tempdima+\@tempdimb
                                        +\dp\@arstrutbox \relax}%
        \the\toks@
   }\ifnum0=`{\fi
            \expandafter}\firsthline % we are then !
}% \tabu@firsthlinecorrection
```

**\tabu@lasthlinecorrection** This is the "correction macro" for \lasthline, *ie.*a strut and a skip are inserted **after** the last \hline.

```
\def\tabu@lasthlinecorrection{%
    \@tempdima \dimexpr  \dp\@arstrutbox+\dimen@+\@tempdimb+\@tempdimc
    \edef\lasthline{%   <local in \noalign>
        \the\toks@
        \noalign{\vskip -\the\dimexpr\dimen@+\@tempdimb+\dp\@arstrutbox}%
        \omit \hbox to\z@{\hss{\noexpand\tabu@DBG{yellow}\vrule
                    depth \the\dimexpr \dp\@arstrutbox+\@tempdimb+\dimen@
                                       +\extratabsurround-\@tempdimc
                    height \z@
                    width \tabustrutrule}\hss}\cr
    }\ifnum0=`{\fi
            \expandafter}\lasthline % we are then !
}% \tabu@lasthlinecorrection
```

**\tabu@LT@@hline** Allowing colored rules even if colortbl is not loaded.

```
\def\tabu@LT@@hline{%
    \ifx\LT@next\hline
        \global\let\LT@next \@gobble
        \ifx \CT@drsc@\relax
            \gdef\CT@LT@sep{%
                \noalign{\penalty-\@medpenalty\vskip\doublerulesep}}%
        \else
            \gdef\CT@LT@sep{%
                \multispan\LT@cols{%
                \CT@drsc@\leaders\hrule\@height\doublerulesep\hfill}\cr}%
        \fi
    \else
        \global\let\LT@next\empty
        \gdef\CT@LT@sep{%
            \noalign{\penalty-\@lowpenalty\vskip-\arrayrulewidth}}%
    \fi
    \ifnum0=`{\fi}%
    \multispan\LT@cols
        {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
    \CT@LT@sep
    \multispan\LT@cols
        {\CT@arc@\leaders\hrule\@height\arrayrulewidth\hfill}\cr
    \noalign{\penalty\@M}%
    \LT@next
}% \tabu@LT@@hline
```

## Horizontal lines: `\tabucline`

**`\tabucline`** `\tabucline` [style or spec.]{start-end}

`\tabucline` appears only at the end of a line: this is the place where we can insert a **\noalign** group. The line to be inserted inside the **tabu** is build inside this **\noalign** group.

**\tabu@start** and **\tabu@stop** store the limits for the line: they are, for clarity, the local name of **\@tempcnta** and **\@tempcntb**.

```
\let\tabu@start \@tempcnta
\let\tabu@stop  \@tempcntb
\newcommand*\tabucline{\noalign{\ifnum0=`}\fi \tabu@cline}
\newcommand*\tabu@cline[2][]{\tabu@startstop{#2}%
   \ifnum \tabu@stop<\z@   \toks@{}%
   \else \tabu@clinearg{#1}\tabu@thestyle
        \edef\tabucline{\toks@{%
            \ifnum \tabu@start>\z@ \omit
                  \tabu@multispan\tabu@start {\span\omit}&\fi
            \omit \tabu@multispan\tabu@stop {\span\omit}%
                                        \tabu@thehline\cr
        }}\tabucline
        \tabu@tracinglines{(tabu:tabucline) Style: #1^^J\the\toks@^^J^^J}%
    \fi
    \futurelet \tabu@temp \tabu@xcline
}% \tabu@cline
\def\tabu@clinearg #1{%
    \ifx\\#1\\\let\tabu@thestyle \tabu@ls@
    \else \@defaultunits \expandafter\let\expandafter\@tempa
                                    \romannumeral-`\0#1\relax \@nnil
        \ifx \hbox\@tempa           \tabu@clinebox{#1}%
        \else\ifx \box\@tempa       \tabu@clinebox{#1}%
        \else\ifx \vbox\@tempa      \tabu@clinebox{#1}%
        \else\ifx \vtop\@tempa      \tabu@clinebox{#1}%
        \else\ifx \copy\@tempa      \tabu@clinebox{#1}%
        \else\ifx \leaders\@tempa   \tabu@clineleads{#1}%
        \else\ifx \cleaders\@tempa  \tabu@clineleads{#1}%
        \else\ifx \xleaders\@tempa  \tabu@clineleads{#1}%
        \else\tabu@getline {#1}%
        \fi\fi\fi\fi\fi\fi\fi\fi
    \fi
}% \tabu@clinearg
\def\tabu@clinebox #1{\tabu@clineleads{\xleaders#1\hss}}
\def\tabu@clineleads #1{%
    \let\tabu@thestyle \relax \let\tabu@leaders \@undefined
    \gdef\tabu@thehrule{#1}}
\def\tabu@thehline{\begingroup
    \ifdefined\tabu@leaders
            \noexpand\tabu@thehleaders
    \else   \noexpand\tabu@thehrule
    \fi            \endgroup
}% \tabu@thehline
\def\tabu@xcline{%
    \ifx \tabu@temp\tabucline
        \toks@\expandafter{\the\toks@ \noalign
        {\ifx\CT@drsc@\relax \vskip
         \else \CT@drsc@\hrule height
         \fi
         \doublerulesep}}%
    \fi
    \tabu@docline
}% \tabu@xcline
\def\tabu@docline {\ifnum0=`{\fi \expandafter}\the\toks@}
\def\tabu@docline@evr {\xdef\tabu@doclineafter{\the\toks@}%
              \ifnum0=`{\fi}\aftergroup\tabu@doclineafter}
\def\tabu@multispan #1#2{%
    \ifnum\numexpr#1>\@ne #2\expandafter\tabu@multispan
    \else                   \expandafter\@gobbletwo
    \fi  {#1-1}{#2}%
}% \tabu@multispan
```

**\tabu@startstop** This macro parses the mandatory argument of \tabucline: start-column and end-column of the \cline.

```
\def\tabu@startstop #1{\tabu@start@stop #1\relax 1-\tabu@nbcols \@nnil}
\def\tabu@start@stop #1-#2\@nnil{%
   \@defaultunits   \tabu@start\number 0#1\relax    \@nnil
   \@defaultunits   \tabu@stop \number 0#2\relax    \@nnil
   \tabu@stop   \ifnum \tabu@start>\tabu@nbcols     \m@ne
                \else\ifnum \tabu@stop=\z@          \tabu@nbcols
                \else\ifnum \tabu@stop>\tabu@nbcols \tabu@nbcols
                \else                               \tabu@stop
                \fi\fi\fi
   \advance\tabu@start \m@ne
   \ifnum \tabu@start>\z@ \advance\tabu@stop -\tabu@start \fi
}% \tabu@start@stop
```

## 11.17 Numbers in tabu

### \tabudecimal

**\tabudecimal** \tabu@tabudecimal is \tabudecimal inside the tabu environment.

```
\def\tabu@tabudecimal #1{%
    \def\tabu@decimal{#1}\@temptokena{}%
    \let\tabu@getdecimal@ \tabu@getdecimal@ignorespaces
    \tabu@scandecimal
}% \tabu@tabudecimal
\def\tabu@scandecimal{\futurelet \tabu@temp \tabu@getdecimal@}
\def\tabu@skipdecimal#1{#1\tabu@scandecimal}
\def\tabu@getdecimal@ignorespaces{%
```

```
    \ifcase 0\ifx\tabu@temp\ignorespaces\else
             \ifx\tabu@temp\@sptoken1\else
             2\fi\fi\relax
            \let\tabu@getdecimal@ \tabu@getdecimal
            \expandafter\tabu@skipdecimal
    \or     \expandafter\tabu@gobblespace\expandafter\tabu@scandecimal
    \else   \expandafter\tabu@skipdecimal
    \fi
}% \tabu@getdecimal@ignorespaces
\def\tabu@get@decimal#1{\@temptokena\expandafter{\the\@temptokena #1}%
                        \tabu@scandecimal}
\def\do#1{%
    \def\tabu@get@decimalspace#1{%
        \@temptokena\expandafter{\the\@temptokena #1}\tabu@scandecimal}%
}\do{ }
\let\tabu@@tabudecimal \tabu@tabudecimal
```

**\tabu@getdecimal**

```
\def\tabu@getdecimal{%
   \ifcase    0\ifx 0\tabu@temp\else
               \ifx 1\tabu@temp\else
               \ifx 2\tabu@temp\else
               \ifx 3\tabu@temp\else
               \ifx 4\tabu@temp\else
               \ifx 5\tabu@temp\else
               \ifx 6\tabu@temp\else
               \ifx 7\tabu@temp\else
               \ifx 8\tabu@temp\else
               \ifx 9\tabu@temp\else
               \ifx .\tabu@temp\else
               \ifx ,\tabu@temp\else
               \ifx -\tabu@temp\else
               \ifx +\tabu@temp\else
               \ifx e\tabu@temp\else
               \ifx E\tabu@temp\else
               \ifx\tabu@cellleft\tabu@temp1\else
               \ifx\ignorespaces\tabu@temp1\else
               \ifx\@sptoken\tabu@temp2\else
            3\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\relax
       \expandafter\tabu@get@decimal
   \or \expandafter\tabu@skipdecimal
   \or \expandafter\tabu@get@decimalspace
   \else\expandafter\tabu@printdecimal
   \fi
}% \tabu@getdecimal
\def\tabu@printdecimal{%
    \edef\tabu@temp{\the\@temptokena}%
    \ifx\tabu@temp\@empty\else
    \ifx\tabu@temp\space\else
        \expandafter\tabu@decimal\expandafter{\the\@temptokena}%
    \fi\fi
}% \tabu@printdecimal
```

## 11.18 Verbatim inside tabu with X columns

**`\tabu@verbatim`** Setup to be done before `\scantokens` to allow verbatim inside the `tabu` environment.

```
\def\tabu@verbatim{%
    \let\verb \tabu@verb
    \let\FV@DefineCheckEnd \tabu@FV@DefineCheckEnd
}% \tabu@verbatim
```

### Compatibility with LaTeX's kernel `\verb` command

**`\tabu@verb`** The `\verb` macro from the latex kernel expands `\@ifstar` in a context where the space token: ␣ has a category code of 12.

This is not compatible with `\scantokens` since `\scantokens` adds a space after each control sequence, including `\verb`:

`\verb` +some verbatim text+ becomes:

`\verb` ␣+some verbatim text+

and thus, the space token ␣ is set as the `\verb` delimiter.

We therefore use (a silly) `\@ifstar` in order to gobble the possible space token.

```
\let\tabu@ltx@verb \verb
\def\tabu@verb{\@ifstar {\tabu@ltx@verb*} \tabu@ltx@verb}
```

### Compatibility with the **fancyvrb** package

**`\tabu@FV@DefineCheckEnd`** This is quite the same issue as for LaTeX `\verb` command: a space is inserted after each control sequence scanned by `\scantoken`.

This leads to a break in the macro that checks the end of a `Verbatim` environment, since this macro basically checks for a line that conforms to the pattern:

**`#1\end {#2}#3`**

while with `\scantokens`, such a line becomes:

**`#1\end ␣{#2}#3`**

in a context where the space token is not of category 10 (space).

Thus we replace the end-check for the `Verbatim` environment by a check on the detokenized-line (with ε-TeX `\detokenize`):

```
\def\tabu@fancyvrb {%
    \def\tabu@FV@DefineCheckEnd ##1{%
        \def\tabu@FV@DefineCheckEnd{%
            ##1% <original definition (if fancyvrb is loaded)>
            \let\FV@CheckEnd     \tabu@FV@CheckEnd
            \let\FV@@CheckEnd    \tabu@FV@@CheckEnd
            \let\FV@@@CheckEnd   \tabu@FV@@@CheckEnd
            \edef\FV@EndScanning{%
            \def\noexpand\next{\noexpand\end{\FV@EnvironName}}%
                \global\let\noexpand\FV@EnvironName\relax
                \noexpand\next}%
            \xdef\FV@EnvironName{\detokenize\expandafter{\FV@EnvironName}}}%
    }\expandafter\tabu@FV@DefineCheckEnd\expandafter{\FV@DefineCheckEnd}
}% \tabu@fancyvrb
\def\tabu@FV@CheckEnd  #1{\expandafter\FV@@CheckEnd \detokenize{#1\end{}}\@nil}
\edef\tabu@FV@@@CheckEnd {\detokenize{\end{}}}
\begingroup
\catcode`\[1      \catcode`\]2
\@makeother\{     \@makeother\}
   \edef\x[\endgroup
```

```
      \def\noexpand\tabu@FV@@CheckEnd ##1\detokenize[\end{]##2\detokenize[}]##3%
   ]\x               \@nil{\def\@tempa{#2}\def\@tempb{#3}}
```

**`\tabu@FV@ListProcessLine`** This macro replaces `\FV@ListProcessLine` when measuring the natural width of a `Verbatim` environment (see `\tabu@startpboxmeasure`)

```
\def\tabu@FV@ListProcessLine #1{%
  \hbox {%to \hsize{%
    \kern\leftmargin
    \hbox {%to \linewidth{%
      \FV@LeftListNumber
      \FV@LeftListFrame
      \FancyVerbFormatLine{#1}\hss
%% DG/SR modification begin - Jan. 28, 1998 (for numbers=right add-on)
%%      \FV@RightListFrame}%
      \FV@RightListFrame
      \FV@RightListNumber}%
%% DG/SR modification end
    \hss}}
```

## 11.19 `\savetabu`

**`\savetabu`** When this command is called by the user, the `tabu` preamble and target are globally stored into a macro `\tabu@saved@`⟨*user-name*⟩.

```
\newcommand*\savetabu[1]{\noalign{%
    \tabu@sanitizearg{#1}\tabu@temp
    \ifx \tabu@temp\@empty  \tabu@savewarn{}{The tabu will not be saved}\else
        \@ifundefined{tabu@saved@\tabu@temp}{}{\tabu@savewarn{#1}{Overwritting}}%
        \ifdefined\tabu@restored \expandafter\let
            \csname tabu@saved@\tabu@temp \endcsname \tabu@restored
        \else {\tabu@save}%
        \fi
    \fi}%
}% \savetabu
\def\tabu@save {%
    \toks0\expandafter{\tabu@saved@}%
    \iftabu@negcoef
        \let\tabu@wddef \relax \let\tabu@ \tabu@savewd \edef\tabu@savewd{\tabu@Xcoefs}%
        \toks0\expandafter{\the\toks\expandafter0\tabu@savewd}\fi
    \toks1\expandafter{\tabu@savedpream}%
    \toks2\expandafter{\tabu@savedpreamble}%
    \let\@preamble \relax
    \let\tabu@savedpream \relax \let\tabu@savedparams \relax
    \edef\tabu@preamble{%
        \def\noexpand\tabu@aligndefault{\tabu@align}%
        \def\tabu@savedparams {\noexpand\the\toks0}%
        \def\tabu@savedpream  {\noexpand\the\toks1}}%
    \edef\tabu@usetabu{%
        \def\@preamble {\noexpand\the\toks2}%
        \tabu@target \the\tabu@target \relax
        \tabucolX    \the\tabucolX    \relax
        \tabu@nbcols \the\tabu@nbcols \relax
        \def\noexpand\tabu@aligndefault{\tabu@align}%
        \def\tabu@savedparams {\noexpand\the\toks0}%
        \def\tabu@savedpream  {\noexpand\the\toks1}}%
    \let\tabu@aligndefault \relax \let\@sharp \relax
    \edef\@tempa{\noexpand\tabu@s@ved
                          {\tabu@usetabu}
                         {\tabu@preamble}
```

```
                            {\the\toks1}}\@tempa
    \tabu@message@save
}% \tabu@save
\long\def\tabu@s@ved #1#2#3{%
    \def\tabu@usetabu{#1}% <for \tabu@message@save>
    \expandafter\gdef\csname tabu@saved@\tabu@temp\endcsname ##1{%
        \ifodd ##1%     \usetabu
            \tabu@measuringfalse \tabu@spreadfalse  % Just in case...
            \gdef\tabu@usetabu {%
                \ifdim \tabu@target>\z@ \tabu@warn@usetabu \fi
                \global\let\tabu@usetabu \@undefined
                \def\@halignto {to\tabu@target}%
                #1%
                \ifx \tabu@align\tabu@aligndefault@text
                \ifnum \tabu@nested=\z@
                       \let\tabu@align \tabu@aligndefault \fi\fi}%
        \else     %     \preamble
            \gdef\tabu@preamble {%
                \global\let\tabu@preamble \@undefined
                #2%
                \ifx \tabu@align\tabu@aligndefault@text
                \ifnum \tabu@nested=\z@
                       \let\tabu@align \tabu@aligndefault \fi\fi}%
        \fi
        #3}%
}% \tabu@s@ved
\def\tabu@aligndefault@text {\tabu@aligndefault}%
\def\tabu@warn@usetabu {\PackageWarning{tabu}
    {Specifying a target with \string\usetabu\space is useless
    \MessageBreak The target cannot be changed!}}
\def\tabu@savewd #1#2{\ifdim #2\p@<\z@ \tabu@wddef{#1}{\tabu@wd{#1}}\fi}
```

**\tabu@savewarn** Info for overwritting when \savetabu is used.

**\tabu@saveerr** Error if \usetabu is called with an unknown argument.

```
\def\tabu@savewarn#1#2{\PackageInfo{tabu}
    {User-name ‘#1’ already used for \string\savetabu
    \MessageBreak #2}}%
\def\tabu@saveerr#1{\PackageError{tabu}
    {User-name ‘#1’ is unknown for \string\usetabu
    \MessageBreak I cannot restore an unknown preamble!}\@ehd}
```

## 11.20 \rowfont

### Setting font and alignment specification

**\rowfont** \rowfont uses the control sequences `\tabu@celllalign`, `\tabu@cellleft`, `\tabu@cellright` and `\tabu@cellralign` which have been placed on purpose into the user-defined tokens inserted in any preamble by the `array` package.

`\tabu@celllalign` and `\tabu@cellralign` are used to modify the alignment. If the optional `[alignment]` parameter of \rowfont is not specified, then those control sequences expand to `\@empty`.

`\tabu@cellleft` contains the font-modification information.

Placement of those control sequences into the user-tokens that are inserted in the preamble by the `array` package is explained below under the macro `\tabu@prepnext@tok`.

```
\newskip \tabu@cellskip
\def\tabu@rowfont{\ifdim \baselineskip=\z@\noalign\fi
                    {\ifnum0=`}\fi    \tabu@row@font}
```

```
\newcommand*\tabu@row@font[2][]{%
    \ifnum7=\currentgrouptype
        \global\let\tabu@@cellleft    \tabu@cellleft
        \global\let\tabu@@cellright   \tabu@cellright
        \global\let\tabu@@celllalign  \tabu@celllalign
        \global\let\tabu@@cellralign  \tabu@cellralign
        \global\let\tabu@@rowfontreset\tabu@rowfontreset
    \fi
    \global\let\tabu@rowfontreset \tabu@rowfont@reset
    \expandafter\gdef\expandafter\tabu@cellleft\expandafter{\tabu@cellleft #2}%
    \ifcsname tabu@cell@#1\endcsname       % row alignment
            \csname tabu@cell@#1\endcsname \fi
    \ifnum0=`{\fi}% end of group / noalign group
}% \rowfont
\def\tabu@ifcolorleavevmode #1{\let\color \tabu@leavevmodecolor #1\let\color\tabu@color}%
```

**\tabu@rowfont@reset** This macro restores `\tabu@celllalign`, `\tabu@cellleft`, `\tabu@cellright`, and `\tabu@cellralign` to the value they had before the expansion of `\rowfont`.

It expands when a new row is inserted into the tabular or array.

```
\def\tabu@rowfont@reset{%
    \global\let\tabu@rowfontreset \tabu@@rowfontreset
    \global\let\tabu@cellleft     \tabu@@cellleft
    \global\let\tabu@cellright    \tabu@@cellright
    \global\let\tabu@cellfont     \@empty
    \global\let\tabu@celllalign   \tabu@@celllalign
    \global\let\tabu@cellralign   \tabu@@cellralign
}% \tabu@@rowfontreset
\let\tabu@rowfontreset \@empty     % overwritten \AtBeginDocument if colortbl
```

## Preparing stuff to be able to use \rowfont

**\tabu@prepnext@tok** `\tabu@prepnext@tok` will replace `\prepnext@tok` defined in array.sty (only inside a tabu environment). its purpose is to count the number of columns, and to insert the control sequences `\tabu@celllalign`, `\tabu@cellleft`, `\tabu@cellright` and `\tabu@cellralign` at the edge of each cell of the tabular. This is done by putting them inside the user-tokens placed around each column by the array package.

`\prepnext@tok` in array.sty initialises each user-token to an empty one, each time there is a need for a new one ! The macro has a very simple definition, but it expansion is the occasion to look carefully at the counters `\count@` and `\@tempcnta` which gives us all information required to decide is the token in preparation will be finally placed on the left or on the right of a column.

$$\underbrace{\texttt{>\{\textbackslash bfseries \textbackslash color \{red\}\}}}_{\texttt{\textbackslash toks}\ <i>}\ \mathrm{r}\ \underbrace{\texttt{<\{\textbackslash color \{black\}\textbackslash , \textbackslash \$ \}}}_{\texttt{\textbackslash toks}\ <i+1>}$$

When a column is inserted in the tabular preamble (`\@preamble`), the TeX counter `\count@` is equal to $i+1$ (*ie.*the right token) and the counter `\@tempcnta` is equal to $i$ (*ie.*the left token). If the column is special (*ie.*`@` or `!`) `\@tempcnta` is not updated.

Thus, when a new token is "prepared" by `\prepnext@tok`:

**either: i =\count @ =\@ tempcnta** : the token to prepare (*ie.*`\toks` $< i+1 >$) is the right one of a "normal" column. The switch `\iftabu@cellright` is set to `true`.
The *previous* token (`\toks` $< i >$=`\toks \count@`) is necessarily the left one of this "normal" column: we prepend `\tabu@celllalign` and append `\tabu@cellleft` to this token (`\toks` $< i >$). This token is finished and will not change afterwards.

**or: i =\count @ =\@ tempcnta**+1 : the token to prepare (`\toks` $< i+1 >$) is either the left one of a normal column, or the single one of a special `@` or `!` column.
If the switch `\iftabu@cellright` is true, then the *previous* token `\toks` $< i >$ is the right one of the last inserted column (which was a "normal" column, thus):, `\tabu@cellright \tabu@cellralign` is appended to it, and the switch `\ittabu@cellright` is reset to `false`. May be `\prepnext@tok` will be

expanded again (by `\save@decl`): if it happens, then again `\count@` $=$`\@tempcnta` $+1$ (same case) but `\iftabu@cellright` is `false` and nothing is changed.

**else:** The token to prepare (which is `\toks` $<i+1>=$`\toks` `\count@` $+1$), cannot be the right one of a “normal” column: `\iftabu@cellright` is set to `false`.
The fact that $|$`\count@` $-$`\@tempcnta` $|>1$ tells us that the previous token `\toks` $<i>$ is necessarily the single one of a “special” `@` or `!` column. We don’t modify this token, as long as *special columns are always inserted as is*: `\rowcolor` has no effect on special columns, nor `\rowfont`.

Thereafter, the original initialisation sequence occurs: `\advance` `\count@` `by\@ne` and initialize the token to prepare (`\toks` `\count@` $=$`\toks` $<i+1>$) to an empty one.

```
\newif \iftabu@cellright
\def\tabu@prepnext@tok{%
    \ifnum \count@<\z@   % <first initialisation>
            \@tempcnta  \@M   % <not initialized by array.sty>
            \tabu@nbcols\z@
            \let\tabu@fornoopORI \@fornoop
            \tabu@cellrightfalse
    \else
        \ifcase \numexpr \count@-\@tempcnta \relax % (case 0): prev. token is left
                \advance \tabu@nbcols \@ne
                \iftabu@cellright % before-previous token is right and is finished
                    \tabu@cellrightfalse % <only once>
                    \tabu@righttok
                \fi
                \tabu@lefttok
        \or                     % (case 1) previous token is right
                \tabu@cellrighttrue \let\@fornoop \tabu@lastnoop
        \else % special column: do not change the token
                \iftabu@cellright    % before-previous token is right
                    \tabu@cellrightfalse
                    \tabu@righttok
                \fi
        \fi % \ifcase
    \fi
    \tabu@prepnext@tokORI
}% \tabu@prepnext@tok
\long\def\tabu@lastnoop#1\@@#2#3{\tabu@lastn@@p #2\@nextchar \in@\in@@}
\def\tabu@lastn@@p #1\@nextchar #2#3\in@@{%
    \ifx \in@#2\else
        \let\@fornoop \tabu@fornoopORI
        \xdef\tabu@mkpreambuffer{\tabu@nbcols\the\tabu@nbcols \tabu@mkpreambuffer}%
        \toks0\expandafter{\expandafter\tabu@everyrowtrue \the\toks0}%
        \expandafter\prepnext@tok
    \fi
}% \tabu@lastnoop
\def\tabu@righttok{%
    \advance \count@ \m@ne
    \toks\count@\expandafter {\the\toks\count@ \tabu@cellright \tabu@cellralign}%
    \advance \count@ \@ne
}% \tabu@righttok
\def\tabu@lefttok{\toks\count@\expandafter{\expandafter\tabu@celllalign
                                    \the\toks\count@ \tabu@cellleft}% after because of $
}% \tabu@lefttok
```

## Neutralisation of glues and alignment modification

**\tabu@cellleft**
**\tabu@celllalign**
**\tabu@cellright**
**\tabu@cellralign**

First initialisation to `\@empty`.

```
\let\tabu@cellleft   \@empty
\let\tabu@cellright  \@empty
\tabu@celllalign@def{\tabu@cellleft}%
\let\tabu@cellralign \@empty
```

**\tabu@cell@align**

```
\def\tabu@cell@align #1#2#3{%
    \let\tabu@maybesiunitx \toks@ \tabu@celllalign
    \global \expandafter \tabu@celllalign@def \expandafter {\the\toks@ #1}%
    \toks@\expandafter{\tabu@cellralign #2}%
    \xdef\tabu@cellralign{\the\toks@}%
    \toks@\expandafter{\tabu@cellleft #3}%
    \xdef\tabu@cellleft{\the\toks@}%
}% \tabu@cell@align
```

**\tabu@cell@l**
**\tabu@cell@c**
**\tabu@cell@r**
**\tabu@cell@j**

Setup macros to modify the alignment. The skips inserted to make the standard alignment specified in the tabular preamble are not the same with standard array tabulars and colortbl tabulars, hence the switch \iftabu@colortbl.

```
\def\tabu@cell@l{% force alignment to left
   \tabu@cell@align
      {\tabu@removehfil \raggedright \tabu@cellleft}% left
      {\tabu@flush1\tabu@ignorehfil}%                 right
      \raggedright
}% \tabu@cell@l
\def\tabu@cell@c{% force alignment to center
   \tabu@cell@align
      {\tabu@removehfil \centering \tabu@flush{.5}\tabu@cellleft}
      {\tabu@flush{.5}\tabu@ignorehfil}
      \centering
}% \tabu@cell@c
\def\tabu@cell@r{% force alignment to right
   \tabu@cell@align
      {\tabu@removehfil \raggedleft \tabu@flush1\tabu@cellleft}
      \tabu@ignorehfil
      \raggedleft
}% \tabu@cell@r
\def\tabu@cell@j{% force justification (for p, m, b columns)
      \tabu@cell@align
         {\tabu@justify\tabu@cellleft}
         {}
         \tabu@justify
}% \tabu@cell@j
\def\tabu@justify{%
   \leftskip\z@skip \@rightskip\leftskip \rightskip\@rightskip
   \parfillskip\@flushglue
}% \tabu@justify
%% ragged2e settings
\def\tabu@cell@L{% force alignment to left (ragged2e)
   \tabu@cell@align
      {\tabu@removehfil \RaggedRight \tabu@cellleft}
      {\tabu@flush 1\tabu@ignorehfil}
      \RaggedRight
}% \tabu@cell@L
\def\tabu@cell@C{% force alignment to center (ragged2e)
```

```
   \tabu@cell@align
      {\tabu@removehfil \Centering \tabu@flush{.5}\tabu@cellleft}
      {\tabu@flush{.5}\tabu@ignorehfil}
      \Centering
}% \tabu@cell@C
\def\tabu@cell@R{% force alignment to right (ragged2e)
   \tabu@cell@align
      {\tabu@removehfil \RaggedLeft \tabu@flush 1\tabu@cellleft}
      \tabu@ignorehfil
      \RaggedLeft
}% \tabu@cell@R
\def\tabu@cell@J{% force justification (ragged2e)
   \tabu@cell@align
      {\justifying \tabu@cellleft}
      {}
      \justifying
}% \tabu@cell@J
\def\tabu@flush#1{%
    \iftabu@colortbl      % colortbl uses \hfill rather than \hfil
        \hskip \ifnum13<\currentgrouptype \stretch{#1}%
        \else  \ifdim#1pt<\p@ \tabu@cellskip
        \else  \stretch{#1}
        \fi\fi \relax
    \else                    % array.sty
        \ifnum 13<\currentgrouptype
                \hfil \hskip1sp \relax  \fi
    \fi
}% \tabu@flush
```

**\tabu@removehfil** \tabu@removehfil removes (eventually) the infinite stretchable glue inserted *before* the cell (in the preamble of \halign) to make the column alignment.

```
\let\tabu@hfil  \hfil
\let\tabu@hfill \hfill
\let\tabu@hskip \hskip
\def\tabu@removehfil{%
    \iftabu@colortbl
        \unkern \tabu@cellskip =\lastskip
        \ifnum\gluestretchorder\tabu@cellskip =\tw@ \hskip-\tabu@cellskip
        \else \tabu@cellskip \z@skip
        \fi
    \else
        \ifdim\lastskip=1sp\unskip\fi
        \ifnum\gluestretchorder\lastskip =\@ne
            \hfilneg % \hfilneg for array.sty but not for colortbl...
        \fi
    \fi
}% \tabu@removehfil
```

**\tabu@ignorehfil** \tabu@ignorehfil removes (eventually) the infinite stretchable glue inserted *after* the cell (in the preamble of \halign) to make the column alignment.

```
\def\tabu@ignorehfil{\aftergroup \tabu@nohfil}
\def\tabu@nohfil{% \hfil -> do nothing + restore original \hfil
   \def\hfil{\let\hfil \tabu@hfil}%   local to (alignment template) group
}% \tabu@nohfil
\def\tabu@colortblalignments {% if colortbl
    \def\tabu@nohfil{%
        \def\hfil  {\let\hfil \tabu@hfil}% local to (alignment template) group
        \def\hfill {\let\hfill \tabu@hfill}% (colortbl uses \hfill) pfff...
        \def\hskip ####1\relax{\let\hskip \tabu@hskip}}% local
}% \tabu@colortblalignments
```

## 11.21 Taking care of footnotes and \arraybackslash

### Footnotes and hyperfootnotes

**\tabu@footenotetext** The macros in case hyperref is not used, or used with the option hyperfootnotes=false:

```
\long\def\tabu@footnotetext #1{%
   \edef\@tempa{\the\tabu@footnotes
      \noexpand\footnotetext [\the\csname c@\@mpfn\endcsname]}%
   \global\tabu@footnotes\expandafter{\@tempa {#1}}}%
\long\def\tabu@xfootnotetext [#1]#2{%
   \global\tabu@footnotes\expandafter{\the\tabu@footnotes
                               \footnotetext [{#1}]{#2}}}
\let\tabu@xfootnote \@xfootnote
```

**\tabu@Hy@ftntext**
**\tabu@Hy@xfootnote** The macros in case hyperref is loaded with the option hyperfootnotes=true:

```
\long\def\tabu@Hy@ftntext{\tabu@Hy@ftntxt {\the \c@footnote }}
\long\def\tabu@Hy@xfootnote [#1]{%
   \begingroup
      \value\@mpfn #1\relax
      \protected@xdef \@thefnmark  {\thempfn}%
   \endgroup
   \@footnotemark \tabu@Hy@ftntxt {#1}%
}% \tabu@Hy@xfootnote
\long\def\tabu@Hy@ftntxt #1#2{%
    \edef\@tempa{%
        \the\tabu@footnotes
        \begingroup
            \value\@mpfn #1\relax
            \noexpand\protected@xdef\noexpand\@thefnmark {\noexpand\thempfn}%
            \expandafter \noexpand \expandafter
                \tabu@Hy@footnotetext \expandafter{\Hy@footnote@currentHref}%
    }%
    \global\tabu@footnotes\expandafter{\@tempa {#2}%
                                         \endgroup}%
}% \tabu@Hy@ftntxt
\long\def\tabu@Hy@footnotetext #1#2{%
    \H@@footnotetext{%
        \ifHy@nesting
            \hyper@@anchor {#1}{#2}%
        \else
            \Hy@raisedlink{%
                \hyper@@anchor {#1}{\relax}%
            }%
            \def\@currentHref {#1}%
            \let\@currentlabelname \@empty
            #2%
        \fi
    }%
}% \tabu@Hy@footnotetext
```

### \centering, \raggedright, \raggedleft and \@normalcr

Inside tabu environment, no need to add \arraybackslash after such commands.

```
\def\tabu@latextwoe {%
\def\tabu@temp##1##2##3{{\toks@\expandafter{##2##3}\xdef##1{\the\toks@}}}
\tabu@temp \tabu@centering   \centering   \arraybackslash
\tabu@temp \tabu@raggedleft  \raggedleft  \arraybackslash
\tabu@temp \tabu@raggedright \raggedright \arraybackslash
}% \tabu@latextwoe
\def\tabu@raggedtwoe {%
\def\tabu@temp ##1##2##3{{\toks@\expandafter{##2##3}\xdef##1{\the\toks@}}}
\tabu@temp \tabu@Centering   \Centering   \arraybackslash
\tabu@temp \tabu@RaggedLeft  \RaggedLeft  \arraybackslash
\tabu@temp \tabu@RaggedRight \RaggedRight \arraybackslash
\tabu@temp \tabu@justifying  \justifying  \arraybackslash
}% \tabu@raggedtwoe
\def\tabu@normalcrbackslash{\let\\\@normalcr}
\def\tabu@trivlist{\expandafter\def\expandafter\@trivlist\expandafter{%
                       \expandafter\tabu@normalcrbackslash \@trivlist}}
```

### Utilities: tabu \fbox

\tabu@fbox works exactly like LaTeX \fbox but allows the syntax: \fbox \bgroup...\egroup suitable for use inside tabular columns. \fbox is \let to \tabu@fbox at the entry inside a tabu environment.

```
\def\tabu@fbox      {\leavevmode\afterassignment\tabu@beginfbox \setbox\@tempboxa\hbox}
\def\tabu@beginfbox {\bgroup \kern\fboxsep
                     \bgroup\aftergroup\tabu@endfbox}
\def\tabu@endfbox   {\kern\fboxsep\egroup\egroup
                     \@frameb@x\relax}
```

\tabu@fcolorbox works exactly like xcolor \fcolorbox but allows the syntax:

\fcolorbox {frame color}{background color}\bgroup...\egroup

suitable for use insed tabular columns. \fcolorbox is \let to \tabu@fcolorbox at the entry inside a tabu environment.

```
\def\tabu@color@b@x #1#2{\leavevmode \bgroup
    \def\tabu@docolor@b@x{#1{#2\color@block{\wd\z@}{\ht\z@}{\dp\z@}\box\z@}}%
    \afterassignment\tabu@begincolor@b@x \setbox\z@ \hbox
}% \tabu@color@b@x
\def\tabu@begincolor@b@x {\kern\fboxsep \bgroup
       \aftergroup\tabu@endcolor@b@x \set@color}
\def\tabu@endcolor@b@x {\kern\fboxsep \egroup
    \dimen@\ht\z@ \advance\dimen@ \fboxsep \ht\z@ \dimen@
    \dimen@\dp\z@ \advance\dimen@ \fboxsep \dp\z@ \dimen@
    \tabu@docolor@b@x \egroup
}% \tabu@endcolor@b@x
```

## 11.22 Corrections

### delarray comptability fix for colortbl and arydshln

Both colortbl and arydshln forgot the control sequence \@arrayright which must be expanded by \endarray. Originally defined for delarray, this control sequence is used by tabu environments when tabu X columns are present in the preamble.

Here is the fix. We test if \endarray contains \@arrayright before modifying the control sequence, in case colortbl and/or arydshln modify their definition.

```
\def\tabu@fix@arrayright {%% \@arrayright is missing from \endarray
```

```
    \iftabu@colortbl
        \ifdefined\adl@array  % <colortbl + arydshln>
        \def\tabu@endarray{%
            \adl@endarray \egroup \adl@arrayrestore \CT@end \egroup %<original>
            \@arrayright      % <FC>
            \gdef\@preamble{}}% <FC>
        \else                 % <colortbl / no arydshln>
        \def\tabu@endarray{%
            \crcr \egroup \egroup    %<original>
            \@arrayright             % <FC>
            \gdef\@preamble{}\CT@end}%
        \fi
    \else
        \ifdefined\adl@array  % <arydshln / no colortbl>
        \def\tabu@endarray{%
            \adl@endarray \egroup \adl@arrayrestore \egroup %<original>
            \@arrayright      % <FC>
            \gdef\@preamble{}}% <FC>
    \else                   % <no arydshln / no colotbl + \@arrayright missing>
        \PackageWarning{tabu}
        {\string\@arrayright\space is missing from the
        \MessageBreak definition of \string\endarray.
        \MessageBreak Comptability with delarray.sty is broken.}%
    \fi\fi
}% \tabu@fix@arrayright
```

## arydshln @ columns

```
\def\tabu@adl@xarraydashrule #1#2#3{%
     \ifnum\@lastchclass=\adl@class@start\else
     \ifnum\@lastchclass=\@ne\else
     \ifnum\@lastchclass=5 \else % <FC> @-arg (class 5) and !-arg (class 1)
             \adl@leftrulefalse \fi\fi           % must be treated the same
     \fi
     \ifadl@zwvrule\else \ifadl@inactive\else
             \@addtopreamble{\vrule\@width\arrayrulewidth
                     \@height\z@ \@depth\z@}\fi \fi
     \ifadl@leftrule
             \@addtopreamble{\adl@vlineL{\CT@arc@}{\adl@dashgapcolor}%
                     {\number#1}#3}%
     \else   \@addtopreamble{\adl@vlineR{\CT@arc@}{\adl@dashgapcolor}%
                     {\number#2}#3}
     \fi
}% \tabu@adl@xarraydashrule
```

## arydshln, colors without colortbl and empty p columns

arydshln redefines `\@endpbox` for p columns. The definition is stored in `\adl@act@endpbox`. Here it is:

```
\unskip \ifhmode \nobreak
    \vrule\@width\z@\@height\z@\@depth\dp\@arstrutbox
    \fi
\egroup \adl@colhtdp \box\adl@box \hfil
```

The `\vrule` inserted is exactly what package array calls: `\@finalstrut \@arstrutbox`.

However, just like in `array.sty`, this array-`strut` should be inserted inconditionnally, and `\ifhmode` applies only to `\nobreak` (misplaced `\fi` in arydshln definition).

Finally, arydshln is not compatible with colors in columns, such that: `>{\color {red}}p3in`, Unless colortbl is also loaded, the color group is missing.

Fixed inside **tabu** environment.

```
\def\tabu@adl@act@endpbox {%
    \unskip \ifhmode \nobreak \fi    \@finalstrut \@arstrutbox
    \egroup \egroup
    \adl@colhtdp \box\adl@box \hfil
}% \tabu@adl@act@endpbox
\def\tabu@adl@fix {%
    \let\adl@xarraydashrule \tabu@adl@xarraydashrule % <fix> arydshln
    \let\adl@act@endpbox    \tabu@adl@act@endpbox    % <fix> arydshln
    \let\adl@act@@endpbox   \tabu@adl@act@endpbox    % <fix> arydshln
    \let\@preamerror        \@preamerr               % <fix> arydshln
}% \tabu@adl@fix
```

### longtable \@startpbox: \everypar needed

**\tabu@LT@startpbox** The leading strut should be inserted at **\everypar** in order for \tabulinesep to work (otherwise, TEX is in horizontal mode and **\nointerlineskip** breaks).

```
\def\tabu@LT@startpbox #1{%
    \bgroup
        \let\@footnotetext\LT@p@ftntext
        \setlength\hsize{#1}%
        \@arrayparboxrestore
        \everypar{%
            \vrule \@height \ht\@arstrutbox \@width \z@
            \everypar{}}%
}% \tabu@LT@startpbox
```

## 11.23 Package options and Initialisation

### \tracingtabu and the package options

**delarray (package option)** The **delarray** package option is only there for convenience: it simply loads the delarray package.

```
\DeclareOption{delarray}{\AtEndOfPackage{\RequirePackage{delarray}}}
```

**linegoal (package option)** The **linegoal** package option only sets **\tabudefaulttarget** to be equal to **\linegoal**. The required package linegoal is loaded.

```
\DeclareOption{linegoal}{%
   \AtEndOfPackage{%
      \RequirePackage{linegoal}[2010/12/07]%
      \let\tabudefaulttarget \linegoal% \linegoal is \linewidth if not pdfTeX
}}
```

**\scantokens (package option)** The **scantokens** package option makes **tabu** equal to **tabu∗** .

```
\DeclareOption{scantokens}{\tabuscantokenstrue}
```

**\tracingtabu**
**debugshow (package option)** \tracingtabu is the same as the package option debugshow.

```
\DeclareOption{debugshow}{\AtEndOfPackage{\tracingtabu=\tw@}}
\def\tracingtabu {\begingroup\@ifnextchar=%
    {\afterassignment\tabu@tracing\count@}
    {\afterassignment\tabu@tracing\count@1\relax}}
\def\tabu@tracing{\expandafter\endgroup
    \expandafter\tabu@tr@cing \the\count@ \relax
}% \tabu@tracing
\def\tabu@tr@cing #1\relax {%
    \ifnum#1>\thr@@ \let\tabu@tracinglines\message
    \else           \let\tabu@tracinglines\@gobble
    \fi
    \ifnum#1>\tw@   \let\tabu@DBG        \tabu@@DBG
```

```
                    \def\tabu@mkarstrut {\tabu@DBG@arstrut}%
                    \tabustrutrule      1.5\p@
    \else           \let\tabu@DBG        \@gobble
                    \def\tabu@mkarstrut {\tabu@arstrut}%
                    \tabustrutrule      \z@
    \fi
    \ifnum#1>\@ne   \let\tabu@debug      \message
    \else           \let\tabu@debug      \@gobble
    \fi
    \ifnum#1>\z@
        \let\tabu@message             \message
        \let\tabu@tracing@save        \tabu@message@save
        \let\tabu@starttimer          \tabu@pdftimer
    \else
        \let\tabu@message             \@gobble
        \let\tabu@tracing@save        \@gobble
        \let\tabu@starttimer          \relax
    \fi
}% \tabu@tr@cing
```

## Initialisation and setup \AtBeginDocument

At the end of the **tabu** package:

- \tracingtabu is set to 0: this initialises the message commands. Eventually,, t he value will be overwritten by the **debugshow** package option later.
- \everyrow is set to empty: this initialises the process at **\everycr** to the default process,
- a new *empty* line style is defined, to be equivalent to **\hline**: this creates the *default leaders*, which will be used if a line style specification cannot be parsed successfully.
  Then this default line style is set to be the current one.

At Begin Document,a fix for arydshln and colortbl comptability with delarray shortcuts available inside **tabu**: requirement for this fix is checked by **\tabu@fix@arrayright**.

Then the switch **\iftabu@colortbl** is set.

Finally, the **longtabu** environment is defined only if the longtable package is detected.

```
\AtBeginDocument{\tabu@AtBeginDocument}
\def\tabu@AtBeginDocument{\let\tabu@AtBeginDocument \@undefined
    \ifdefined\arrayrulecolor   \tabu@colortbltrue       % <colortbl>
                                \tabu@colortblalignments % different glues are used
    \else                       \tabu@colortblfalse \fi
    \ifdefined\CT@arc@ \else \let\CT@arc@  \relax \fi
    \ifdefined\CT@drsc@\else \let\CT@drsc@ \relax \fi
    \let\tabu@arc@L \CT@arc@ \let\tabu@drsc@L \CT@drsc@
    \ifodd 1\ifcsname siunitx_table_collect_begin:Nn\endcsname    % <siunitx: ok>
            \expandafter\ifx
                \csname siunitx_table_collect_begin:Nn\endcsname\relax 0\fi\fi\relax
            \tabu@siunitxtrue
    \else   \let\tabu@maybesiunitx   \@firstofone                 % <not siunitx: setup>
            \let\tabu@siunitx        \tabu@nosiunitx
            \tabu@siunitxfalse
    \fi
    \ifdefined\adl@array        % <arydshln>
    \else     \let\tabu@adl@fix \relax
              \let\tabu@adl@endtrial \@empty \fi
    \ifdefined\longtable        % <longtable>
    \else     \let\longtabu \tabu@nolongtabu \fi
    \ifdefined\cellspacetoplimit \tabu@warn@cellspace\fi
    \csname\ifcsname ifHy@hyperfootnotes\endcsname % <hyperfootnotes>
            ifHy@hyperfootnotes\else iffalse\fi\endcsname
        \let\tabu@footnotetext \tabu@Hy@ftntext
        \let\tabu@xfootnote    \tabu@Hy@xfootnote \fi
    \ifdefined\FV@DefineCheckEnd% <fancyvrb>
            \tabu@fancyvrb  \fi
    \ifdefined\color             % <color / xcolor>
        \let\tabu@color \color
        \def\tabu@leavevmodecolor ##1{%
            \def\tabu@leavevmodecolor {\leavevmode ##1}%
        }\expandafter\tabu@leavevmodecolor\expandafter{\color}%
    \else
        \let\tabu@color            \tabu@nocolor
        \let\tabu@leavevmodecolor \@firstofone \fi
    \tabu@latextwoe
    \ifdefined\@raggedtwoe@everyselectfont     % <ragged2e>
        \tabu@raggedtwoe
    \else
        \let\tabu@cell@L \tabu@cell@l
        \let\tabu@cell@R \tabu@cell@r
        \let\tabu@cell@C \tabu@cell@c
        \let\tabu@cell@J \tabu@cell@j   \fi
    \expandafter\in@ \expandafter\@arrayright \expandafter{\endarray}%
    \ifin@ \let\tabu@endarray \endarray
    \else  \tabu@fix@arrayright \fi% <fix for colortbl & arydshln (delarray)>
    \everyrow{}%
}% \tabu@AtBeginDocument
\def\tabu@warn@cellspace{%
    \PackageWarning{tabu}{%
                  Package cellspace has some limitations
    \MessageBreak And redefines some macros of array.sty.
    \MessageBreak Please use \string\tabulinesep\space to control
    \MessageBreak vertical spacing of lines inside tabu environnement}%
}% \tabu@warn@cellspace
```

\ProcessOption ∗ is much quicker than without the star...

```
\tabuscantokensfalse
\let\tabu@arc@G         \relax
\let\tabu@drsc@G        \relax
\let\tabu@evr@G         \@empty
\let\tabu@rc@G          \@empty
\def\tabu@ls@G          {\tabu@linestyle@}%
\let\tabu@@rowfontreset \@empty % <init>
\let\tabu@@celllalign   \@empty
\let\tabu@@cellralign   \@empty
\let\tabu@@cellleft     \@empty
\let\tabu@@cellright    \@empty
\def\tabu@naturalXmin   {\z@}
\def\tabu@naturalXmax   {\z@}
\let\tabu@rowfontreset  \@empty
\def\tabulineon {4pt}\let\tabulineoff \tabulineon
\tabu@everyrowtrue
\ifdefined\pdfelapsedtime                    % <pdfTeX>
        \def\tabu@pdftimer {\xdef\tabu@starttime{\the\pdfelapsedtime}}%
\else   \let\tabu@pdftimer \relax \let\tabu@message@etime \relax
\fi
\tracingtabu=\z@
\newtabulinestyle {=\maxdimen}% creates the 'factory' settings \tabu@linestyle@
```

```
\tabulinestyle{}
\taburowcolors{}
\let\tabudefaulttarget  \linewidth
\ProcessOptions*                 % \ProcessOptions* is quicker !

⟨/package⟩
```

# 12 References

[1] *A new implementation of LaTeX's* `tabular` *and* `array` *environments* by Frank Mittelbach
2008/09/09 v2.4c – Tabular extension package (FMi)
CTAN:help/Catalogue/entries/array.html

[2] *The* varwidth *package* by Donald Arseneau
2009/03/30 ver 0.92 – Variable-width minipages
CTAN:help/Catalogue/entries/varwidth.html

[3] *The* enumitem-zref *package* by FC
2011/02/18 ver 1.8 – Extended references for enumitem pkg
CTAN:help/Catalogue/entries/enumitem-zref.html

# 13 History

**[2011/02/26 v2.8]**

- Bug in the starred version (with `\scantokens`) of the `longtabu*` environment.

**[2011/02/25 v2.7]**

- Automatic `\par` after the end of the `tabu` environment used with its default target is removed in case of `tabu spread`: this was a bug.
- Some `\ignorespaces` were missing (in `\everyrow`, `\taburulecolor`, `\taburowcolors` and `\tabulinestyle`).

**[2011/02/24 v2.6]**

- `\savetabu` now also saves `\tabulinesep` (*ie.*`\abovetabulinesep` and `\belowtabulinesep`)
- Bug fixed for custom-environments when nested.
- `\taburulecolor` works even if colortbl is not loaded for the `tabu` environment.
  This is now the same for the `longtabu` environment.

**[2011/02/19 v2.5]**

- Bug fixed for `\pdfelapsedtime` when compilation without pdfTeX.
- Modification of `\@finalstrut` ("null-rule" added) to avoid problems with `\columncolor`.

**[2011/02/17 v2.4]**

- Documentation revisited

**[2011/02/13 v2.3]**

- Fixed two bugs for nested `tabu` environment: when using `\rowfont` and when `tabu` is nested inside `longtabu`

**[2011/02/12 v2.2 – New implementation - Absolutely no modification of `array.sty`]**

- $\tau_{\aleph}b\subset$ has been totally reimplemented, including the algorithms.
  In particular, outside of the `tabu` environment, absolutely none of the macros of `array.sty`, (and obviously none of LaTeX) is modified.

  The process has been completely reinvented: `tabu` follows a path along different modes (or strategies) measuring natural width of cells, fixing `X` column widths, measuring vertical length of rows and then printing the final tabular. The process is optimized, especially in the case of nested `tabu` environments: a tabular is not built twice for measuring purpose... As a result, many new features are now possible... vertical leaders (dashed lines), dynamic vertical spacing adjustment, and hopefully still more in a next release.

  `tabu` now systematically collects the environment body. But with `\scantokens`, it is possible to insert verbatim material inside the columns: use `tabu*` instead of `tabu`, for the outer most tabular.

- New: `\firsthline` and `\lasthline` can draw multiple lines, and there is an option to set `\extratabsurround` instantly, and locally.
- New: `\taburulecolor` with a good behaviour with groupings (like `\everyrow`)
- Modification: `\tabulinestyle` sets the line style for the tabu, `\newtabulinestyle` defines a new line style.

| | | | | | |
|---|---|---|---|---|---|
| **This** | **is** | **the** | **new** | τℵb⊂ | **package** |

### [2011/01/19 v2.1]

- Vertical spacing had a bug with `longtabu` and paragraph columns.
  Fixed.
- New: `\everyrow`.
- Fix a bug of `\rowfont` when using siunitx `S` columns.
- Some code optimisation.
- To do (if possible): a syntax `X[6mc]S[...]` to "embed" siunitx `S` column inside `tabu` and `longtabu` `X` columns...

### [2011/01/18 v2.0]

- Vertical spacing of lines implemented ! See `\tabulinesep` and `\extrarowsep`.
- `\tabulinestyle` : user defined line style can now be used inside the optional argument of the `|[...]` preamble token.
- `|[...]` is now allowed in `\multicolumn` preamble inside `tabu` environment.
- Bug fixed inside `\tabu@prepnexttok` (again !!! - a difficult case !)
- Incompatibility of package cellspace with `tabu spread` and `tabu` with negativ coefficients for `X` columns with has been lifted.
  However, as said in the documentation of package cellspace, `S` column modifier does not work in the case of nested tabulars.
  The `S` column modifier becomes `C` when the package siunitx is loaded (see siunitx documentation).
  Moreover, cellspace does not work with color or xcolor and paragraph column types !!
  Finally, cellspace redefines **globally** `\@startpbox` and `\@endpbox` and is therefore not fully compatible with array.sty and therefore with τℵb⊂.
  For all those reasons, τℵb⊂ displays a warning to discourage the use of cellspace with the `tabu` environment.

### [2011/01/15 v1.9]

- Bug in `\savetabu` when used inside `longtabu`...
- Bug when `tabu` with `X` column is nested inside `lontabu`.
- Documentation (`\rowfont` was missing in the summary).

### [2010/12/28 v1.8]

- `\tracingtabu` / debugshow package option:
  reporting of the time elapsed during trials (if `\pdfelapsedtime` and thus pdfTEX is available)
  Slight modifications for better reporting on the .log file.
- Fix a bug when `\savetabu` is used after `\multicolumn` (`\multicolumn` globally redefines `\@preamble`).
- Fix a bug with `\tabucline` and `\CT@arc@` (colortbl).
- Better privacy of columns types specifically defined for `tabu`.
- Improvement in the rewritting process (but only very few people should notice...)
- Documentation.

## [2010/12/18 v1.7]

- Code optimisation
- Modification in the columns rewritting process (bug with some new column types defined by the user).

## [2010/12/07 v1.5]

- Implementation of negativ width coefficients for `X` columns (cf. `tabu X` columns – Mastering horizontal space point 2).
- Columns natural widths computation (for `tabu spread` with `X` columns and negativ coefficients) is based on the code of the `varwidth` package by Donald Arseneau.
- longtabu is now provided, based on the longtable package by David Carlisle.
  longtabu can be used just like `tabu`.
- Vertical lines can be used whatever the catcode of | is.
- `\savetabu` reports saved informations in the .log (`debugshow` option).
- `\savetabu`... `\usetabu` now restores the `\halign` preamble rather than the `tabu` preamble!
  `\preamble` can be use in the `tabu` preamble to restore a `tabu` preamble.
- `\tabucline` is more robust with "special" preambles containing `>` or `<` tokens.
  `\tabucline` now takes care of `\arrayrulecolor` (package colortbl).
- enumitem-zref package has been added to the documentation (see the link point 1)
- Optimisation of some parts of the code.

## [2010/11/22 v1.4]

- Compatibility improvement with linegoal for the syntax:
  `\begin {tabu} to\linegoal {...}`
- Hyper footnotes now work correctly.
- Fix a bug when using colored vertical lines in `tabu` in math mode.
- Fix a bug with vertical lines and colortbl `\arrayrulecolor` specification.
- Fix a compatibility bug with arydshln: when nesting a tabular that use vertical dashed lines (arydshln) inside `tabu spread` with `X` columns.

## [2010/11/18 v1.3]

- Fix a bug that may appear in `\tabucline` depending on the preamble due to arbitrary `\countdef`.
- Improvement in the use of `\everycr`: no `\global` stuff. Thus bug fixed when nesting `tabu` inside AMS-`align` environment for example. Same issue with `\rowfont` which now works without global modification of `\everycr`.
- No phantom line is added to `tabu` but a command `\tabuphantomline` is provided for this purpose (required with `\multicolumn` in some cases).
- Improvement on vertical alignment.
- To do: an example file to test a wide range of possibilities...
- Documentation.

**[2010/11/15 v1.2]**

- Improvement in parameters parsing for optional parameters (| and \tabucline).
- Modification / optimization in \tabu@prepnext@tok.
- Modification of \tabucline to get better results with m columns (X[m]) and also when \minrowclearance $> 0$ (package colortbl).

**[2010/10/28 v1.1]**

- First version.

# 14 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; plain numbers refer to the code lines where the entry is used.