longtablex — A mix of tabularx and longtable[*]

Wybo Dekker[†]

Released ?

Abstract

The `longtablex` package combines the properties of the `longtable` end `tabularx` packages, making it possible to use the X column specifier in tables that can be split up by TeX's page-breaking mechanism. This package requires the `array` and `longtable` packages.

1 Introduction

The `longtablex` package is intended to replace the `ltablex` package. The latter, by Anil K. Goel, makes modifications to the `tabularx` package (by David Carlisle) with (almost) the same properties as the package presented here. However, this makes it impossible to use the unmodified tabularx in floats.

`longtablex` `\begin{longtablex}{`⟨*width*⟩`}{`⟨*preamble*⟩`}`

The arguments of `longtablex` are essentially the same as those of the standard `tabular*` environment. However rather than adding space between the columns to achieve the desired width, it adjusts the widths of some of the columns. The columns which are affected by the `longtablex` environment should be denoted with the letter `X` in the preamble argument. The `X` column specification will be converted to `p{`⟨*some value*⟩`}` once the correct column width has been calculated.

2 Example

The following table is set with `\begin{longtablex}{250pt}{|c|X|c|X|} ....` It is essentially the same as the first example in the `tabularx` documentation; however, the second row has been repeated four times, so that the table would normally not have enough space on the current page.

Table 1: Example

Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.

[*]This file describes version ?, last revised ?.
[†]E-mail: wybo@dekkerdocumenten.nl

Table 1: Example (continued)

one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.
one	The width of this column depends on the width of the table.	three	Column four will act in the same way as column two, with the same width.

3 Implementation

```
⟨*package⟩
\DeclareOption{infoshow}{\AtEndOfPackage\tracinglongtablex}
\DeclareOption{debugshow}{\AtEndOfPackage\tracinglongtablex}
\ProcessOptions
```

This requires `array.sty`.

```
\RequirePackage{array}[1994/02/03]
\RequirePackage{longtable}[1994/12/08]
```

First some registers etc. that we need.

```
\newdimen\LTX@col@width
\newdimen\LTX@old@table
\newdimen\LTX@old@col
\newdimen\LTX@target
\newdimen\LTX@delta
\newcount\LTX@cols
\newif\ifLTX@convertX@
\LTX@convertX@true

\newcommand\keepXColumns{
  \LTX@convertX@false
}

\newcommand\convertXColumns{
  \LTX@convertX@true
}

\newif\ifLTX@
```

Now a trick to get the body of an environment into a token register, without doing any expansion. This does not do any real checking of nested environments, so if you should need to nest one `longtablex` inside another, the inner one must be surrounded by `{ }`.

`\longtablex` Prior to v1.06, this macro took two arguments, which were saved in separate registers before the table body was saved by `\LTX@get@body`. Unfortunatly this disables the `[t]` optional argument. Now just save the width specification separately, then clear the token register `\toks@`. Finally call `\LTX@get@body` to begin saving the body of the table. The `{\ifnum0=‘}\fi` was added at v1.03, to allow `longtablex` to appear inside a `\halign`.[1]

[1]This adds an extra level of grouping, which is not really needed. Instead, I could use here, and `\ifnum0=‘{}\fi` below, however the code here would then have to be moved after the first line, because of the footnote to page 386 of the TEXBook, and I do not think I should be writing code that is so obscure as to be documented in a footnote in an appendix called “Dirty Tricks”!

This mechanism of grabbing an environment body does have the disadvantage (shared with the AMS alignment environments) that you can not make extension environments by code such as

```
\newenvironment{foo}{\begin{longtablex}{XX}}{\end{longtablex}}
```

as the code is looking for a literal string `\end{longtablex}` to stop scanning. Since version 2.02, one may avoid this problem by using `\longtablex` and `\endlongtablex` directly in the definition:

```
\newenvironment{foo}{\longtablex{XX}}{\endlongtablex}
```

The scanner now looks for the end of the current environment (`foo` in this example.) There are some restrictions on this usage, the principal one being that `\endlongtablex` is the *first* token of the 'end code' of the environment.

```
\def\longtablex#1{%
```

Allow `\longtablex` `\endlongtablex` (but not `\begin{longtablex}` `\end{longtablex}`) to be used in `\newenvironment` definitions.

```
\edef\LTX@{\@currenvir}%
  {\ifnum0=`}\fi
```

`\relax` added at v1.05 so that non-expandable length tokens, like `\textwidth` do not generate an extra space, and an overfull box. `\relax` removed again at v4.09 in favour of `\setlength` so if you use the calc package you can use a width of `(\textwidth-12pt)/2`.

```
  \setlength\LTX@target{#1}%
  \LTX@typeout{Target width: #1 = \the\LTX@target.}%
  \toks@{}\LTX@get@body}
```

`\endlongtablex` This does not do very much…

```
\let\endlongtablex\relax
```

`\LTX@get@body` Place all tokens as far as the first `\end` into a token register. Then call `\LTX@find@end` to see if we are at `\end{longtablex}`.

```
\long\def\LTX@get@body#1\end
  {\toks@\expandafter{\the\toks@#1}\LTX@find@end}
```

`\LTX@find@end` If we are at `\end{longtablex}`, call `\LTX@endlongtablex`, otherwise add `\end{...}` to the register, and call `\LTX@get@body` again.

```
\def\LTX@find@end#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\LTX@\expandafter\LTX@endlongtablex
  \else\toks@\expandafter
    {\the\toks@\end{#1}}\expandafter\LTX@get@body\fi}
```

`\LTX@` The string `longtablex` as a macro for testing with `\ifx`.

```
\def\LTX@{longtablex}
```

Now that all the parts of the table specification are stored in registers, we can begin the work of setting the table.

The algorithm for finding the correct column widths is as follows. Firstly set the table with each `X` column the width of the final table. Assuming that there is at least one `X` column, this will produce a table that is too wide. Divide the excess width by the number of `X` columns, and reduce the column width by this amount. Reset the table. If the table is not now the correct width, a `\multicolumn` entry must be 'hiding' one of the `X` columns, and so there is one less `X` column affecting the width of the table. So we reduce by 1 the number of X columns and repeat the process.

`\LTX@endlongtablex` Although I have tried to make `longtablex` look like an environment, it is in fact a command, all the work is done by this macro.

```
\newcommand\LTX@endlongtablex{%
```

Define the X column, with an internal version of the \newcolumntype command. The \expandafter commands enable \NC@newcol to get the *expansion* of \longtablexcolumn{\LTX@col@width} as its argument. This will be the definition of an X column, as discussed in section 4.

```
\expandafter\LTX@newcol\expandafter{\longtablexcolumn{\LTX@col@width}}%
```

Initialise the column width, and the number of X columns. The number of X columns is set to one, which means that the initial count will be one too high, but this value is decremented before it is used in the main loop.

Since v1.02, switch the definition of \verb.

```
\let\verb\LTX@verb
```

Since v1.05, save the values of all LaTeX counters, the list \cl@@ckpt contains the names of all the LaTeX counters that have been defined so far. We expand \setcounter at this point, as it results in fewer tokens being stored in \LTX@ckpt, but the actual resetting of the counters occurs when \LTX@ckpt is expanded after each trial run. Actually since v1.07, use something equivalent to the expansion of the original definition of \setcounter, so that longtablex works in conjunction with calc.sty.

```
\def\@elt##1{\global\value{##1}\the\value{##1}\relax}%
\edef\LTX@ckpt{\cl@@ckpt}%
\let\@elt\relax
\LTX@old@table=\maxdimen
\LTX@col@width=\LTX@target
\global\LTX@cols=\@ne
```

Typeout some headings (unless this is disabled).

```
\LTX@typeout@
  {\@spaces Table Width\@spaces Column Width\@spaces X Columns}%
```

First attempt. Modify the X definition to count X columns. From here to to the end of the macro has been replaced with the code from ltablex.sty:

```
\let\savecaption\caption
\renewcommand{\caption}[2][]{\tabularnewline}
\let\saveendhead\endhead
\def\endhead{\\}
\let\saveendfirsthead\endfirsthead
\def\endfirsthead{\\}
\let\saveendfoot\endfoot
\def\endfoot{\\}
\let\saveendlastfoot\endlastfoot
\def\endlastfoot{\\}
\ifLTX@convertX@
  \LTX@trial{\def\NC@rewrite@X{\NC@find l}}
  \ifdim\wd\@tempboxa<\LTX@target
    \LTX@newcol{l}
  \else
    \LTX@convertX@false
  \fi
\fi

\ifLTX@convertX@
  \relax
\else
  \LTX@trial{\def\NC@rewrite@X{%
      \global\advance\LTX@cols\@ne\NC@find p{\LTX@col@width}}}%
  \loop
    \LTX@arith
    \ifLTX@
    \LTX@trial{}%
  \repeat
\fi
{\let\@footnotetext\LTX@ftntext\let\@xfootnotenext\LTX@xftntext
  \LTchunksize\maxdimen
  \let\caption\savecaption
```

```
    \let\endhead\saveendhead
    \let\endfirsthead\saveendfirsthead
    \let\endfoot\saveendfoot
    \let\endlastfoot\saveendlastfoot
    %
    \expandafter\longtable
      \the\toks@
    \endlongtable
    }%
  \global\LTX@ftn\expandafter{\expandafter}\the\LTX@ftn
  \ifnum0=`{\fi}%
   \end{longtablex}
}
```

\LTX@arith Calculate the column width for the next try, setting the flag \ifLTX@ to false if the loop should be aborted.

```
\def\LTX@arith{%
  \LTX@false
  \ifdim\LTX@old@table=\wd\@tempboxa
```

If we have reduced the column width, but the table width has not changed, we stop the loop, and output the table (which will cause an over-full alignment) with the previous value of \LTX@col@width.

```
    \LTX@col@width\LTX@old@col
    \LTX@typeout@{Reached minimum width, backing up.}%
  \else
```

Otherwise calculate the amount by which the current table is too wide.

```
    \dimen@\wd\@tempboxa
    \advance\dimen@ -\LTX@target
    \ifdim\dimen@<\LTX@delta
```

If this amount is less than \LTX@delta, stop. (\LTX@delta should be non-zero otherwise we may miss the target due to rounding error.)

```
      \LTX@typeout@{Reached target.}%
    \else
```

Reduce the number of effective X columns by one. (Checking that we do not get 0, as this would produce an error later.) Then divide excess width by the number of effective columns, and calculate the new column width. Temporarily store this value (times -1) in \dimen@.

```
      \ifnum\LTX@cols>\@ne
        \advance\LTX@cols\m@ne
      \fi
      \divide\dimen@\LTX@cols
      \advance\dimen@ -\LTX@col@width
      \ifdim \dimen@ >\z@
```

If the new width would be too narrow, abort the loop. At the moment too narrow, means less than 0 pt!

Prior to v2.03, if the loop was aborted here, the X columns were left with the width of the previous run, but this may make the table far too wide as initial guesses are always too big. Now force to \LTX@error@width which defaults to be 1em. If you want to get the old behaviour stick
\renewcommand\LTX@error@width{\LTX@col@width}
in a package file loaded after longtablex.

```
        \PackageWarning{longtablex}%
          {X Columns too narrow (table too wide)\MessageBreak}%
      \LTX@col@width\LTX@error@width\relax
      \else
```

Otherwise save the old settings, and set the new column width. Set the flag to true so that the table will be set, and the loop will be executed again.

```
        \LTX@old@col\LTX@col@width
        \LTX@old@table\wd\@tempboxa
        \LTX@col@width-\dimen@
        \LTX@true
      \fi
    \fi
  \fi}
```

`\LTX@error@width` If the calculated width is negative, use this instead.

```
\def\LTX@error@width{1em}
```

`\LTX@delta` Accept a table that is within `\hfuzz` of the correct width.

```
\LTX@delta\hfuzz
```

Initialise the `X` column. The definition can be empty here, as it is set for each `longtablex` environment.

```
\newcolumntype{X}{}
```

`\longtablexcolumn` The default definition of `X` is `p{#1}`.

```
\def\longtablexcolumn#1{p{#1}}
```

`\LTX@newcol` A little macro just used to cut down the number of `\expandafter` commands needed.

```
\def\LTX@newcol{\newcol@{X}[0]}
```

`\LTX@trial` Make a test run.

```
\def\LTX@trial#1{%
  \setbox\@tempboxa\hbox{%
```

Any extra commands. This is used on the first run to count the number of `X` columns.

```
    #1\relax
```

Since v1.04, make `\footnotetext` gobble its arguments. Also locally clear `\LTX@vwarn` so that the warning is generated by the `final` run, and does not appear in the middle of the table if `\tracinglongtablex`.

```
  \let\@footnotetext\LTX@trial@ftn
  \let\LTX@vwarn\@empty
```

Do not nest `longtablex` environments during trial runs. This would waste time, and the global setting of `\LTX@cols` would break the algorithm.

```
   \expandafter\let\expandafter\longtablex\csname tabular*\endcsname
   \expandafter\let\expandafter\endlongtablex\csname endtabular*\endcsname
```

Added at v1.05: dissable `\write`s during a trial run. This trick is from the TEXBook.[2]

```
   \def\write{\begingroup
     \def\let{\afterassignment\endgroup\toks@}%
        \afterassignment\let\count@}%
```

Turn off warnings (see appendix D). Also prevent them being turned back on by setting the parameter names to be registers.

```
    \hbadness\@M
    \hfuzz\maxdimen
    \let\hbadness\@tempcnta
    \let\hfuzz\@tempdima
```

Make the table, and finish the hbox. Since v1.06, `\toks@` contains the preamble specification, and possible optional argument, as well as the table body.

```
    \expandafter\tabular\the\toks@
    \endtabular}%
```

Since v1.05 reset all LaTeX counters, by executing `\LTX@ckpt`.

```
  \LTX@ckpt
```

[2]Actually the TEXBook trick does not work correctly, so changed for v2.05.

Print some statistics. Added `\LTX@align` in v1.05, to line up the columns.

```
  \LTX@typeout@{\@spaces
     \expandafter\LTX@align
        \the\wd\@tempboxa\space\space\space\space\space\@@
     \expandafter\LTX@align
        \the\LTX@col@width\space\space\space\space\space\@@
     \@spaces\the\LTX@cols}}
```

`\LTX@align` Macro added at v1.05, to improve the printing of the tracing info.

```
\def\LTX@align#1.#2#3#4#5#6#7#8#9\@@{%
  \ifnum#1<10 \space\fi
  \ifnum#1<100 \space\fi
  \ifnum#1<\@m\space\fi
  \ifnum#1<\@M\space\fi
  #1.#2#3#4#5#6#7#8\space\space}
```

`\arraybackslash` `\\` hack.

```
\def\arraybackslash{\let\\\@arraycr}
```

`\tracinglongtablex` Print statistics on column and table widths.

```
\def\tracinglongtablex{%
  \def\LTX@typeout{\PackageWarningNoLine{longtablex}}%
  \def\LTX@typeout@##1{\typeout{(longtablex) ##1}}}
```

`\LTX@typeout` The default is to be to be quiet

```
\let\LTX@typeout\@gobble
\let\LTX@typeout@\@gobble
```

`\LTX@ftn` A token register for saving footnote texts.

```
\newtoks\LTX@ftn
```

`\LTX@ftntext` `\LTX@xftntext` Inside the alignment just save up the footnote text in a token register.

```
\long\def\LTX@ftntext#1{%
  \edef\@tempa{\the\LTX@ftn\noexpand\footnotetext
                    [\the\csname c@\@mpfn\endcsname]}%
  \global\LTX@ftn\expandafter{\@tempa{#1}}}%
\long\def\LTX@xftntext[#1]#2{%
  \global\LTX@ftn\expandafter{\the\LTX@ftn\footnotetext[#1]{#2}}}
```

`\LTX@trial@ftn` On trial runs, gobble footnote texts.

```
\long\def\LTX@trial@ftn#1{}
```

This last section was added at Version 1.02. Previous versions documentented the fact that `\verb` did not work inside `longtablex`, but that did not stop people using it! This usually put LaTeX into an irrecoverable error position, with error messages that did not mention the cause of the error. The 'poor man's `\verb`' (and `\verb*`) defined here is based on page 382 of the TeXBook. As explained there, doing verbatim this way means that spaces are not treated correctly, and so `\verb*` may well be useless, however I consider this section of code to be error-recovery, rather than a real implementation of verbatim.

The mechanism is quite general, and any macro which wants to allow a form of `\verb` to be used within its argument may `\let\verb=\LTX@verb`. (Making sure to restore the real definition later!)

`\verb` and `\verb*` are subject to the following restictions:

1. Spaces in the argument are not read verbatim, but may be skipped according to TeX's usual rules.

2. Spaces will be added to the output after control words, even if they were not present in the input.

3. Unless the argument is a single space, any trailing space, whether in the original argument, or added as in (2), will be omitted.
4. The argument must not end with \, so \verb|\| is not allowed, however, because of (3), \verb|\ | produces \.
5. The argument must be balanced with respect to { and }. So \verb|{| is not allowed.
6. A comment character like % will not appear verbatim. It will act as usual, commenting out the rest of the input line!
7. The combinations ?` and !` will appear as ¿ and ¡ if the cmtt font is being used.

\LTX@verb The internal definition of \verb. Spaces will be replaced by ~, so for the star-form, \let ~ be ␣, which we obtain as \uppercase{*}. Use {\ifnum0=`}\fi rather than \bgroup to allow & to appear in the argument.

```
{\uccode`\*=`\ %
\uppercase{\gdef\LTX@verb{%
  \leavevmode\null\LTX@vwarn
  {\ifnum0=`}\fi\ttfamily\let\\\ignorespaces
  \@ifstar{\let~*\LTX@vb}{\LTX@vb}}}}
```

\LTX@vb Get the 'almost verbatim' text using \meaning. The '!' is added to the front of the user supplied text, to ensure that the whole argument does not consist of a single { } group. TeX would strip the outer braces from such a group. The '!' will be removed later.

Originally I followed Knuth, and had \def\@tempa{##1}, however this did not allow # to appear in the argument. So in v1.04, I changed this to to use a token register, and \edef. This allows # appear, but makes each one appear twice!, so later we loop through, replacing ## by #.

```
\def\LTX@vb#1{\def\@tempa##1#1{\toks@{##1}\edef\@tempa{\the\toks@}%
    \expandafter\LTX@v\meaning\@tempa\\ \\\ifnum0=`{\fi}}\@tempa!}
```

\LTX@v Strip the initial segment of the \meaning, including the '!' added earlier.

```
\def\LTX@v#1!{\afterassignment\LTX@vfirst\let\@tempa= }
```

As explained above we are going to replace ## pairs by #. To do this we need non-special # tokens. Make * into a parameter token so that we can define macros with arguments. The normal meanings will be restored by the \endgroup later.

```
\begingroup
\catcode`\*=\catcode`\#
\catcode`\#=12
```

\LTX@vfirst As a special case, prevent the first character from being dropped. This makes \verb*| | produce ␣. Then call \LTX@v@. This is slightly tricky since v1.04, as I have to ensure that an actual # rather than a command \let to # is passed on if the first character is #.

```
\gdef\LTX@vfirst{%
  \if\@tempa#%
    \def\@tempb{\LTX@v@#}%
  \else
    \let\@tempb\LTX@v@
    \if\@tempa\space~\else\@tempa\fi
  \fi
  \@tempb}
```

\LTX@v@ Loop through the \meaning, replacing all spaces by ~. If the last charcter is a space it is dropped, so that \verb*|\LaTeX| produces \LaTeX not \LaTeX␣. The rewritten tokens are then further processed to replace ## pairs.

```
\gdef\LTX@v@*1 *2{%
  \LTX@v@hash*1##\relax\if*2\\\else~\expandafter\LTX@v@\fi*2}
```

`\LTX@v@hash` The inner loop, replacing `##` by `#`.

```
\gdef\LTX@v@hash*1##*2{*1\ifx*2\relax\else#\expandafter\LTX@v@hash\fi*2}
```

As promised, we now restore the normal meanings of `#` and `*`.

```
\endgroup
```

`\LTX@vwarn` Warn the user the first time this `\verb` is used.

```
\def\LTX@vwarn{%
  \@warning{\noexpand\verb may be unreliable inside longtablex}%
  \global\let\LTX@vwarn\@empty}
```

```
⟨/package⟩
```

4 Change History

v1.00
- General: First public release 1

v2.00
- General: New Option Handling 2
  - Use LaTeX2e's `\RequirePackage` to load array 2

v2.02
- `\endlongtablex`: Macro added 3
- `\longtablex`: New local setting of `\LTX@` 3

v2.03
- `\LTX@arith`: Improve warning message and force to 1em. 5
- `\LTX@error@width`: macro added. 6

v2.04
- `\LTX@error@width`: spurious brace removed. 6

v2.05
- `\LTX@trial`: New `\write` trick. tools/2607 6

v2.07
- `\LTX@trial`: Newer `\write` trick. tools/2792 6

v4.09
- `\longtablex`: Use `\setlength`, so that calc extensions apply. tools/2793 . . 3

5 Index

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