The longtable package*

David Carlisle†

2024-10-27

Abstract

This package defines the longtable environment, a multi-page version of tabular.

List of Tables

1 Introduction

`longtable` (*env.*) The longtable package defines a new environment, longtable, which has most of the features of the tabular environment, but produces tables which may be broken by TeX's standard page-breaking algorithm. It also shares some features with the table environment. In particular it uses by default the same counter, `table`, and has a similar `\caption` command. Also, the standard `\listoftables` command lists tables produced by either the table or longtable environments.

The following example uses most of the features of the longtable environment. An edited listing of the input for this example appears in Section 9.

Note: Various parts of the following table will **not** line up correctly until this document has been run through LaTeX several times. This is a characteristic feature of this package, as described below.

*This file has version number v4.22, last revised 2024-10-27.

†The new algorithm for aligning 'chunks' of a table used in version 4 of this package was devised, coded and documented by David Kastrup.

Table 1: A long table

*	This part appears at the top of the table		*
*	First	Second	*
*	longtable columns are specified	in the	*
*	same way as in the tabular	environment.	*
*	`@{*}r||p{1in}@{*}`	in this case.	*
*	Each row ends with a	\\ command.	*
*	The \\ command has an	optional	*
*	argument, just as in	the	*
*	tabular	environment.	*
*	See the effect of `\\[10pt]`	?	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Also `\hline` may be used,	as in tabular.	*
*	That was a `\hline`	.	*
*	That was `\hline\hline`	.	*
‖	This is a `\multicolumn{2}{||c||}`		‖
*	If a page break occurs at a `\hline` then	a line is drawn	*
*	at the bottom of one page and at the	top of the next.	*
*	The `[t]` `[b]` `[c]` argument of tabular	can not be used.	*
*	The optional argument may be one of	`[l]` `[r]` `[c]`	*
*	to specify whether the table should be	adjusted	*
*	to the left, right	or centrally.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	Lots of lines	like this.	*
*	This goes at the	bottom.	*

Table 1: (continued)

* This part appears at the top of every other page *	
* **First**	**Second** *
*Some lines may take up a lot of space, like this:	This last* column is a “p” column so this “row” of the table can take up several lines. Note however that TEX will never break a page within such a row. Page breaks only occur between rows of the table or at `\hline` commands.
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots of lines	like this. *
* Lots[1] of lines	like this. *
* Lots of lines	like this[2] *
* Lots of lines	like this. *
* Lots of lines	like this. *
* These lines will	appear *
* in place of the	usual foot *
* at the end	of the table *

2 Chunk Size

`LTchunksize` In order to TEX multi-page tables, it is necessary to break up the table into smaller chunks, so that TEX does not have to keep everything in memory at one time. By default longtable uses 20 rows per chunk, but this can be set by the user, with e.g., `\setcounter{LTchunksize}{10}`.[3] These chunks do not affect page breaking, thus if you are using a TEX with a lot of memory, you can set `LTchunksize` to be several pages of the table. TEX will run faster with a large `LTchunksize`.

[1] This is a footnote.

[2] longtable takes special precautions, so that footnotes may also be used in ‘p’ columns.

[3] You can also use the plain TEX syntax `\LTchunksize=10`.

A	tabular	environment
within	a floating	table

Table 2: A floating table

However, if necessary, longtable can work with `LTchunksize` set to 1, in which case the memory taken up is negligible. Note that if you use the commands for setting the table head or foot (see below), the `LTchunksize` must be at least as large as the number of rows in each of the head or foot sections.

This document specifies `\setcounter{LTchunksize}{200}`. If you look at the previous table, after the *first* run of LaTeX you will see that various parts of the table do not line up. LaTeX will also have printed a warning that the column widths had changed. longtable writes information onto the `.aux` file, so that it can line up the different chunks. Prior to version 4 of this package, this information was not used unless a `\setlongtables` command was issued, however, now the information is always used, via a new algorithm,[4] and so `\setlongtables` is no longer needed. It is defined (but does nothing) for the benefit of old documents that use it.

3 Counter and Caption Types

As mentioned in the introduction longtable uses and updates by default the `table` counter, the `\caption` command creates a table caption which is added to the list of tables. Packages like ltcaption added more flexibility here by adding the command `\LTcaptype` which allowed to change the type, e.g. to a listing. Starting with version 4.21 longtable supports this command directly. By redefining this command it is possible to change the counter and caption type. After `\renewcommand\LTcaptype{`⟨*counter*⟩`}` longtable will update the counter ⟨*counter*⟩, use `\fnum@`⟨*counter*⟩ in the caption (which typically will make use of `\`⟨*counter*⟩`name` and `\the`⟨*counter*⟩), and write content line entries into the file with the extension stored in the command `\ext@`⟨*counter*⟩. When hyperref is loaded the name of the anchor will use ⟨*counter*⟩ too. Packages or documents that change `\LTcaptype` to some nonstandard value must ensure that the counter ⟨*counter*⟩ and the commands `\fnum@`⟨*counter*⟩ and `\ext@`⟨*counter*⟩ exist and do not error. If `\LTcaptype` is empty no counter is advanced and `\`⟨*counter*⟩`name` in the caption is suppressed.

`\LTcaptype`

4 Captions and Headings

At the start of the table one may specify lines which are to appear at the top of every page (under the headline, but before the other lines of the table). The lines are entered as normal, but the last `\\` command is replaced by a `\endhead` command. If the first page should have a different heading, then this should be entered in the same way, and terminated with the `\endfirsthead` command. The `LTchunksize` should be at least as large as the number of rows in the heading. There are also `\endfoot` and `\endlastfoot` commands which are used in the same

`\endhead`
`\endfirsthead`
`\endfoot`
`\endlastfoot`

[4] Due to David Kastrup.

way (at the *start* of the table) to specify rows (or an `\hline`) to appear at the bottom of each page. In certain situations, you may want to place lines which logically belong in the table body at the end of the firsthead, or the beginning of the lastfoot. This helps to control which lines appear on the first and last page of the table.

`\caption` The `\caption{...}` command is essentially equivalent to
`\multicolumn{n}{c}{\parbox{\LTcapwidth}{...}}`
where `n` is the number of columns of the table. You may set the width of the caption with a command such as `\setlength{\LTcapwidth}{2in}` in the preamble of your document. The default is 4in. `\caption` also writes the information to produce an entry in the list of tables. As with the `\caption` command in the figure and table environments, an optional argument specifies the text to appear in the list of tables if this is different from the text to appear in the caption. Thus the caption for table 1 was specified as `\caption[An optional table caption (used in the list of tables)]{A long table\label{long}}`.

You may wish the caption on later pages to be different to that on the first page. In this case put the `\caption` command in the first heading, and put a subsidiary caption in a `\caption[]` command in the main heading. If the optional argument to `\caption` is empty, no entry is made in the list of tables. Alternatively, if you do not want the table number to be printed each time, use the `\caption*` command.

The captions are set based on the code for the article class. If you have redefined the standard `\@makecaption` command to produce a different format for the captions, you may need to make similar changes to the longtable version, `\LT@makecaption`. See the code section for more details.

A more convenient method of customising captions is given by the caption(2) package, which provides commands for customising captions, and arranges that the captions in standard environments, and many environments provided by packages (including longtable) are modified in a compatible manner.

You may use the `\label` command so that you can cross reference longtables with `\ref`. Note, however, that the `\label` command should not be used in a heading that may appear more than once. Place it either in the firsthead, or in the body of the table. It should not be the *first* command in any entry.

5 Multicolumn entries

The `\multicolumn` command may be used in longtable in exactly the same way as for tabular. So you may want to skip this section, which is rather technical, however coping with `\multicolumn` is one of the main problems for an environment such as longtable. The main effect that a user will see is that certain combinations of `\multicolumn` entries will result in a document needing more runs of LaTeX before the various 'chunks' of a table align.

The examples in this section are set with `LTchunksize` set to the minimum value of one, to demonstrate the effects when `\multicolumn` entries occur in different chunks.

Consider Table 3. In the second chunk, longtable sees the wide multicolumn entry. At this point it thinks that the first two columns are very narrow. All the width of the multicolumn entry is assumed to be in the third column. (This is a 'feature' of TeX's primitive `\halign` command.) longtable then passes the

Table 3: A difficult `\multicolumn` combination: pass 1

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

Table 4: A difficult `\multicolumn` combination: pass 2

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

Table 5: A difficult `\multicolumn` combination: pass 3

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

Table 6: A difficult `\multicolumn` combination: pass 4

1	2	3
wide multicolumn spanning 1–3		
multicolumn 1–2		3
wide 1	2	3

information that there is a wide third column to the later chunks, with the result that the first pass over the table is too wide.

If the 'saved row' from this first pass was re-inserted into the table on the next pass, the table would line up in two passes, but would be much two wide.

\kill The solution to this problem used in Versions 1 and 2, was to use a \kill line. If a line is \killed, by using \kill rather than \\ at the end of the line, it is used in calculating column widths, but removed from the final table. Thus entering \killed copies of the last two rows before the wide multicolumn entry would mean that \halign 'saw' the wide entries in the first two columns, and so would not widen the third column by so much to make room for the multicolumn entry.

In Version 3, a new solution was introduced. If the saved row in the .aux file was not being used, longtable used a special 'draft' form of \multicolumn, this modified the definition, so the spanning entry was never considered to be wider than the columns it spanned. So after the first pass, the .aux file stored the widest normal entry for each column, no column was widened due to \spanned columns. By default longtable ignored the .aux file, and so each run of LaTeX was considered a first pass. Once the \setlongtables declaration was given, the saved row in the .aux file, and the proper definition of \multicolumn were used. If any \multicolumn entry caused one of the columns to be widened, this information could not be passed back to earlier chunks, and so the table would not correctly line up until the third pass. This algorithm always converged in three passes as described above, but in examples such as the ones in Tables 3–6, the final widths were not optimal as the width of column 2, which is determined by a \multicolumn entry, was not known when the final width for column 3 was fixed, due to the fact that *both* \multicolumn commands were switched from 'draft' mode to 'normal' mode at the same time.

Version 4 alleviates the problem considerably. The first pass of the table will indeed have the third column much too wide. However, on the next pass longtable will notice the error and reduce the column width accordingly. If this has to propagate to chunks before the \multicolumn one, an additional pass will, of course, be needed. It is possible to construct tables where this rippling up of the correct widths takes several passes to 'converge' and produce a table with all chunks aligned. However in order to need many passes one needs to construct a table with many overlapping \multicolumn entries, all being wider than the natural widths of the columns they span, and all occurring in different chunks. In the typical case the algorithm will converge after three or four passes, and the benefits of not needing to edit the document before the final run to add \setlongtables, and the better choice of final column widths in the case of multiple \multicolumn entries will hopefully more than pay for the extra passes that may possibly be needed.

So Table 3 converges after 4 passes, as seen in Table 6.

You can still speed the convergence by introducing judicious \kill lines, if you happen to have constellations like the above.

If you object even to LaTeX-ing a file twice, you should make the first line of every longtable a \kill line that contains the widest entry to be used in each column. All chunks will then line up on the first pass.

6 Adjustment

The optional argument of longtable controls the horizontal alignment of the table. The possible options are `[c]`, `[r]` and `[l]`, for centring, right and left adjustment, respectively. Normally centring is the default, but this document specifies

\LTleft
\LTright

```
\setlength\LTleft\parindent
\setlength\LTright\fill
```

in the preamble, which means that the tables are set flush left, but indented by the usual paragraph indentation. Any lengths can be specified for these two parameters, but at least one of them should be a rubber length so that it fills up the width of the page, unless rubber lengths are added between the columns using the `\extracolsep` command. For instance

```
\begin{tabular*}{\textwidth}{@{\extracolsep{...}}...}
```

produces a full width table, to get a similar effect with longtable specify

```
\setlength\LTleft{0pt}
\setlength\LTright{0pt}
\begin{longtable}{@{\extracolsep{...}}...}
```

7 Changes

This section highlights the major changes since version 2. A more detailed change log may be produced at the end of the code listing if the `ltxdoc.cfg` file specifies

```
\AtBeginDocument{\RecordChanges}
\AtEndDocument{\PrintChanges}
```

Changes made between versions 2 and 3.

- The mechanism for adding the head and foot of the table has been completely rewritten. With this new mechanism, longtable does not need to issue a `\clearpage` at the start of the table, and so the table may start half way down a page. Also the `\endlastfoot` command, which could not safely be implemented under the old scheme, has been added.
- longtable now issues an error if started in the scope of `\twocolumn`, or the multicols environment.
- The separate documentation file `longtable.tex` has been merged with the package file, `longtable.dtx` using Mittelbach's doc package.
- Support for footnotes has been added. Note however that `\footnote` will not work in the 'head' or 'foot' sections of the table. In order to put a footnote in those sections (e.g., inside a caption), use `\footnotemark` at that point, and `\footnotetext` anywhere in the table *body* that will fall on the same page.
- The treatment of `\multicolumn` has changed, making `\kill` lines unnecessary, at the price of sometimes requiring a third pass through LaTeX.
- The `\newpage` command now works inside a longtable.

Changes made between versions 3 and 4.

- A new algorithm is used for aligning chunks. As well as the widest width in each column, longtable remembers which chunk produced this maximum. This allows it to check that the maximum is still achieved in later runs. As longtable can now deal with columns shrinking as the file is edited, the `\setlongtables` system is no longer needed and is disabled.
- An extra benefit of the new algorithm's ability to deal with 'shrinking' columns is that it can give better (narrower) column widths in the case of overlapping `\multicolumn` entries in different chunks than the previous algorithm produced.
- The 'draft' multicolumn system has been removed, along with related commands such as `\LTmulticolumn`.
- The disadvantage of the new algorithm is that it can take more passes. The theoretical maximum is approximately twice the length of a 'chain' of columns with overlapping `\multicolumn` entries, although in practice it usually converges as fast as the old version. (Which always converged in three passes once `\setlongtables` was activated.)
- `\\*` and `\nopagebreak` commands may be used to control page breaking.

8 Summary

Table 7: A summary of longtable commands

Parameters		
`\LTleft`	Glue to the left of the table.	(`\fill`)
`\LTright`	Glue to the right of the table.	(`\fill`)
`\LTpre`	Glue before the table.	(`\bigskipamount`)
`\LTpost`	Glue after the table.	(`\bigskipamount`)
`\LTcapwidth`	The width of a parbox containing the caption.	(4in)
`LTchunksize`	The number of rows per chunk.	(20)
Optional arguments to `\begin{longtable}`		
none	Position as specified by `\LTleft` and `\LTright`.	
`[c]`	Centre the table.	
`[l]`	Place the table flush left.	
`[r]`	Place the table flush right.	
Commands to end table rows		
`\\`	Specifies the end of a row	
`\\[`⟨*dim*⟩`]`	Ends row, then adds vertical space (as in the tabular environment).	
`\\*`	The same as `\\` but disallows a page break after the row.	
`\tabularnewline`	Alternative to `\\` for use in the scope of `\raggedright` and similar commands that redefine `\\`.	
`\kill`	Row is 'killed', but is used in calculating widths.	
`\endhead`	Specifies rows to appear at the top of every page.	
`\endfirsthead`	Specifies rows to appear at the top of the first page.	

`\endfoot`	Specifies rows to appear at the bottom of every page.
`\endlastfoot`	Specifies rows to appear at the bottom of the last page.

longtable caption commands

`\caption{`⟨*caption*⟩`}`	Caption 'Table ?: ⟨*caption*⟩', and a '⟨*caption*⟩' entry in the list of tables.
`\caption[`⟨*lot*⟩`]{`⟨*caption*⟩`}`	Caption 'Table ?: ⟨*caption*⟩', and a '⟨*lot*⟩' entry in the list of tables.
`\caption[]{`⟨*caption*⟩`}`	Caption 'Table ?: ⟨*caption*⟩', but no entry in the list of tables.
`\caption*{`⟨*caption*⟩`}`	Caption '⟨*caption*⟩', but no entry in the list of tables.

Commands available at the start of a row

`\pagebreak`	Force a page break.
`\pagebreak[`⟨*val*⟩`]`	A 'hint' between 0 and 4 of the desirability of a break.
`\nopagebreak`	Prohibit a page break.
`\nopagebreak[`⟨*val*⟩`]`	A 'hint' between 0 and 4 of the undesirability of a break.
`\newpage`	Force a page break.

Footnote commands available inside longtable

`\footnote`	Footnotes, but may not be used in the table head & foot.
`\footnotemark`	Footnotemark, may be used in the table head & foot.
`\footnotetext`	Footnote text, use in the table body.

Setlongtables

`\setlongtables`	Obsolete command. Does nothing now.

9 Verbatim highlights from Table 1

```
\begin{longtable}{@{*}r||p{1in}@{*}}
KILLED & LINE!!!! \kill
\caption[An optional table caption ...]{A long table\label{long}}\\
\hline\hline
\multicolumn{2}{@{*}c@{*}}%
     {This part appears at the top of the table}\\
\textsc{First}&\textsc{Second}\\
\hline\hline
\endfirsthead
\caption[]{(continued)}\\
\hline\hline
\multicolumn{2}{@{*}c@{*}}%
      {This part appears at the top of every other page}\\
\textbf{First}&\textbf{Second}\\
\hline\hline
\endhead
\hline
This goes at the&bottom.\\
\hline
\endfoot
\hline
These lines will&appear\\
in place of the & usual foot\\
at the end& of the table\\
\hline
\endlastfoot
\env{longtable}  columns  are specified& in the \\
same way as  in the \env{tabular}& environment.\\
...
\multicolumn{2}{||c||}{This is a ...}\\
...
Some lines may take...&
    \raggedleft This last column is a ``p'' column...
    \tabularnewline
...
Lots of lines& like this.\\
...
\hline
Lots\footnote{...} of lines& like this.\\
Lots   of   lines& like this\footnote{...}\\
\hline
Lots of lines& like this.\\
...
\end{longtable}
```

10 The Macros

1 ⟨*package⟩

10.1 Initial code

Before declaring the package options, we must define some defaults here.

`\LT@err` The error generating command

```
2 \def\LT@err{\PackageError{longtable}}
```

`\LT@warn` The warning generating command

```
3 \def\LT@warn{\PackageWarning{longtable}}
```

`\LT@final@warn` If any longtables have not aligned, generate a warning at the end of the run at `\AtEndDocument`.

```
4 \def\LT@final@warn{%
5   \AtEndDocument{%
6     \LT@warn{Table \@width s have changed. Rerun LaTeX.\@gobbletwo}}%
7   \global\let\LT@final@warn\relax}
```

10.2 Options

The first two options deal with error handling. They are compatible with the options used by the `tracefnt` package.

`errorshow` *Only* show errors on the terminal. 'warnings' are just sent to the log file.

```
8 \DeclareOption{errorshow}{%
9   \def\LT@warn{\PackageInfo{longtable}}}
```

`pausing` Make every warning message into an error so TEX stops. May be useful for debugging.

```
10 \DeclareOption{pausing}{%
11   \def\LT@warn#1{%
12     \LT@err{#1}{This is not really an error}}}
```

`set` `final` The next options are just alternative syntax for the `\setlongtables` declaration.

```
13 \DeclareOption{set}{}
14 \DeclareOption{final}{}

15 \ProcessOptions
```

10.3 User Settable Parameters

`\LTleft` `\LTright` Glue to the left and right of the table, default `\fill` (ie centred).

```
16 \newskip\LTleft       \LTleft=\fill
17 \newskip\LTright      \LTright=\fill
```

`\LTpre` `\LTpost` Glue before and after the longtable. `\bigskip` by default.

```
18 \newskip\LTpre        \LTpre=\bigskipamount
19 \newskip\LTpost       \LTpost=\bigskipamount
```

`\LTchunksize` Chunk size (the number of rows taken per `\halign`). Default 200.

```
20 \newcount\LTchunksize \LTchunksize=200
```

\c@LTchunksize Added in V3.07 to allow the LaTeX syntax \setcounter{LTchunksize}{10}.

```
\let\c@LTchunksize\LTchunksize
```

\LTcapwidth Width of the \parbox containing the caption. Default 4in.

```
\newdimen\LTcapwidth  \LTcapwidth=4in
```

\LTcaptype The name used as counter, in caption, \addcontentsline and in targets. \providecommand is used for compability with ltcaption

```
\providecommand\LTcaptype{table}
```

10.4 Internal Parameters

\LT@head \LT@firsthead \LT@foot \LT@lastfoot Boxes for the table head and foot.

```
\newbox\LT@head
\newbox\LT@firsthead
\newbox\LT@foot
\newbox\LT@lastfoot
```

\LT@gbox

```
\newbox\LT@gbox
```

\LT@cols Counter for number of columns.

```
\newcount\LT@cols
```

\LT@rows Counter for rows up to chunksize.

```
\newcount\LT@rows
```

\c@LT@tables Counter for the tables, added in V3.02. Previous versions just used the LaTeX counter table, but this fails if table is reset during a document, eg report class resets it every chapter.

This was changed from \newcount\LT@tables in V3.04. LaTeX counters are preserved correctly when \includeonly is used. In the rest of the file \LT@tables has been replaced by \c@LT@tables without further comment.

```
\newcounter{LT@tables}
\providecommand\theHLT@tables{\theLT@tables}
```

\c@LT@chunks We need to count through the chunks of our tables from Version 4 on.

```
\newcounter{LT@chunks}[LT@tables]
```

\c@table \fnum@table \tablename \ext@table If the table counter is not defined (eg in letter style), define it. (Added in V3.06.)

```
\ifx\c@table\undefined
  \newcounter{table}
  \def\fnum@table{\tablename~\thetable}
\fi
\ifx\tablename\undefined
  \def\tablename{Table}
\fi
\ifx\ext@table\undefined
  \def\ext@table{lot}
\fi
```

\LT@out In a normal style, longtable uses the .aux file to record the column widths. With letter.sty, use a separate .lta file. (Added in V3.06.)

Not needed for new letter class.

```
\ifx\startlabels\undefined
  \let\@auxout\@auxout
\else
  {\@input{\jobname.lta}}%
  \newwrite\@auxout
  \immediate\openout\@auxout=\jobname.lta
\fi
```

\LT@p@ftn Temporary storage for footnote text in a 'p' column.

```
\newtoks\LT@p@ftn
```

\LT@end@pen Special penalty for the end of the table. Done this way to save using up a count register.

```
\mathchardef\LT@end@pen=30000
```

10.5 The longtable environment

\longtable Called by \begin{longtable}. This implementation does not work in multiple column formats. \par added at V3.04.

```
\def\longtable{%
  \par
  \if@noskipsec\mbox{}\par\fi
  \@nobreakfalse
  \ifx\multicols\@undefined
  \else
     \ifnum\col@number>\@ne
       \@twocolumntrue
     \fi
  \fi
  \if@twocolumn
    \LT@err{longtable not in 1-column mode}\@ehc
  \fi
  \UseTaggingSocket{tbl/vmode/begin}%
  \begingroup
```

Check for an optional argument.

```
  \@ifnextchar[\LT@array{\LT@array[x]}}
```

\LT@array

```
⟨@@=tbl⟩
\ExplSyntaxOn
```

Start setting the alignment. Based on \@array from the LaTeX kernel and the array package.

Since Version 3.02, longtable has used the internal counter \c@LT@tables. The LaTeX counter table is still incremented so that \caption works correctly.

```
\def\LT@array[#1]#2{%
```

With respect to tagging we have a complicated situation with longtable. When at the begin the `\endhead`, `\endfirsthead`, `\endfoot` and `\endlastfoot` are used to setup head and foot they create each a structure subtree with one or more rows. From these structures we want to keep at most two (head and foot) and move the foot to the end of the table. When the head and foot boxes are (re)inserted on following pages we want to mark them up as artifact with the exception of the head at the begin and the foot box at the end.

TODO: When a line is killed the structure subtree is there already too and must be removed. If hard to do, then maybe at first warn if the construction is used.

`\LT@array` is executed in a group, so we can disable para-tagging here.

```
\UseTaggingSocket{tbl/init}
\tl_if_empty:eTF { \LTcaptype }
  {
    \tl_gset:Ne \@currentHref {LT@tables.\theHLT@tables}
  }
  {
    \@kernel@refstepcounter{\LTcaptype}\stepcounter{LT@tables}
```

The target is created rather late and a `\label` can come earlier, so we have to define `\@currentHref` explicitly. We can't currently assume that `\theHtable` is defined always.

```
    \tl_gset:Ne \@currentHref {\LTcaptype.\cs_if_exist_use:c {theH\LTcaptype}}
  }
\tbl_gzero_row_count:
\UseTaggingSocket{tbl/longtable/init}
```

Set up the glue around the table if an optional argument given.

```
\if l#1%
  \LTleft\z@ \LTright\fill
\else\if r#1%
  \LTleft\fill \LTright\z@
\else\if c#1%
  \LTleft\fill \LTright\fill
\fi\fi\fi
```

Set up these internal commands for longtable.

```
\global\let\LT@mcw@rn\relax
\let\LT@mcol\multicolumn
```

Now redefine `\@tabarray` to restore `\hline` and `\multicolumn` so that arrays and tabulars nested in longtable (or in page headings on longtable pages) work out OK. Saving the original definitions done here so that you can load the array package before or after longtable.

```
\let\LT@@@@tabarray\@tabarray
\let\LT@@@@hl\hline
\def\@tabarray{%
  \let\hline\LT@@@@hl
  \let\multicolumn\LT@mcol
```

```
    \LT@@@@tabarray}%
  \let\\\LT@tabularcr
  \let\tabularnewline\\%
  \def\newpage{\noalign{\break}}%
```

More or less standard definitions, but first start a `\noalign`.

```
  \def\pagebreak{\noalign{\ifnum`}=0\fi\@testopt{\LT@no@pgbk-}4}%
  \def\nopagebreak{\noalign{\ifnum`}=0\fi\@testopt\LT@no@pgbk4}%
  \let\hline\LT@hline \let\kill\LT@kill\let\caption\LT@caption
  \@tempdima\ht\strutbox
  \let\@endpbox\LT@endpbox
```

Set up internal commands according to Lamport or Mittelbach.

```
  \ifx\extrarowheight\@undefined
```

Initialise these commands as in tabular from the LaTeX kernel.

```
    \let\@acol\@tabacol
    \let\@classz\@tabclassz \let\@classiv\@tabclassiv
    \def\@startpbox{\vtop\LT@startpbox}%
    \let\@@@@startpbox\@startpbox
    \let\@@@@endpbox\@endpbox
    \let\LT@LL@FM@cr\@tabularcr
  \else
```

Initialise these commands as in array. `\d@llar` replaced by `\d@llarbegin` `\d@llarend` in V3.03 to match array V2.0h. We do not need to set `\d@llarbegin` and `\d@llarend` as the array package gives them the correct values at the top level.

```
    \advance\@tempdima\extrarowheight
    \col@sep\tabcolsep
    \let\@startpbox\LT@startpbox\let\LT@LL@FM@cr\@arraycr
  \fi
```

The rest of this macro is mainly based on array package, but should work for the standard tabular too.

```
  \setbox\@arstrutbox\hbox{\vrule
    \@height \arraystretch \@tempdima
    \@depth \arraystretch \dp \strutbox
    \@width \z@}%
  \let\@sharp##\let\protect\relax
```

Interpret the preamble argument.

```
   \begingroup
    \@mkpream{#2}%
    \tbl_count_table_cols:
```

We need to rename `\@preamble` here as F.M.'s scheme uses `\global`, and we may need to nest `\@mkpream`, eg for `\multicolumn` or an array. We do not need to worry about nested longtables though!

```
    \xdef\LT@bchunk{%
```

We aren't inside any row when a chunk starts.

```
      \tbl_inbetween_rows:
       \global\advance\c@LT@chunks\@ne
       \global\LT@rows\z@\setbox\z@\vbox\bgroup
```

The following line was added in v4.05. In order to get the `\penalties` to work at chunk boundaries, we need to take more care about where and when `\lineskip` glue is added. The following does nothing at top of table, and in header chunks, but in normal body chunks it sets `\prevdepth` (to 0pt, but any value would do) so that `\lineskip` glue will be added. The important thing to note is that the glue will be added *after* any vertical material coming from `\noalign`.

```
        \LT@setprevdepth
        \tabskip\LTleft \noexpand\halign to\hsize\bgroup
%        \tabskip\LTleft\halign to\hsize\bgroup
        \tabskip\z@ \@arstrut
```

Insert the tagging socket to start the row and initialize the cell data for the row.

```
        \UseTaggingSocket{tbl/row/begin}%
        \tbl_init_cell_data_for_row:
        \@preamble \tabskip\LTright \cr}%
  \endgroup
```

Find out how many columns we have (store in `\LT@cols`).

```
  \expandafter\LT@nofcols\LT@bchunk&\LT@nofcols
```

Get the saved row from `\LT@i`... `\LT@ix` (from the `.aux` file), or make a new blank row.

```
  \LT@make@row
```

A few more internal commands for longtable.

```
  \m@th\let\par\@empty
```

Tagging socket and conditional

```
  \everycr{%
    \noalign{%
```

In longtable we have a bunch of extra `\cr`s that are executed whenever a chunk ends. In that case they should not increment the main row counter, sigh.

TODO: At the moment this tracing still exposes the internal row counter!

```
      \@@_trace:n {--longtable-->~chunk~row:~ \the\LT@rows \space
                   row:~ \the\g_@@_row_int   \space
                   column:~ \the\g_@@_col_int
      }
      \tbl_if_row_was_started:T
          {
            \UseTaggingSocket{tbl/row/end}
```

The next setting prevents any of the additional `\cr`s at the end of the chunk to add another /TR. Then once we really start a new chunk it gets incremented so...

```
            \tbl_inbetween_rows:
          }
```

And for the same reason such `\cr`s should not increment the main row counter (but it has to be incremented after the preamble of a chunk), so here we test against `\LT@rows` which is `\LTchunksize` at the end of a chunk.

```
      \int_compare:nNnT \LT@rows < \LTchunksize
         { \tbl_gincr_row_count: }   % next is row about to start
    }%
  }%
```

```
147   \lineskip\z@\baselineskip\z@
```

Start the first chunk.

```
148   \LT@bchunk}
149 \ExplSyntaxOff
150 ⟨@@=⟩
```

`\LT@no@pgbk` Can simplify the standard `\@no@pgbk` as this is vmode only but then need to close the `\noalign`.

```
151 \def\LT@no@pgbk#1[#2]{\penalty #1\@getpen{#2}\ifnum‘{=0\fi}}
```

`\LT@start` This macro starts the process of putting the table on the current page. It is not called until either a `\\` or `\endlongtable` command ends a chunk, as we do not know until that point which of the four possible head or foot sections have been specified.

It begins by redefining itself, so that the table is only started once! Until V3.04, was redefined to `\relax`, now use `\endgraf` to force the page-breaker to wake up. The second `\endgraf` is there so that `\pagetotal` is updated and so takes `\LTpre` into account.

```
152 ⟨@@=tbl⟩
153 \ExplSyntaxOn
```

```
154 \def\LT@start{%
155   \let\LT@start\endgraf
156   \endgraf\penalty\z@\vskip\LTpre\endgraf
```

This next block was suggested by Lars Hellström in pr tools/3396. He documents it as:

The original problem occurs because TeX has not yet found an awfully bad `(b=*)` breakpoint and is therefore still collecting material to see if there is a really good break somewhere just ahead. As we know there aren't, we want to make it stop looking and break the page, so that `\pagetotal` will be for the page where the table will actually end up. To achieve this, we need to give TEX an awfully bad, but legal, breakpoint. The simplest way of doing this seems to be to insert a `\kern` that counters the `\pageshrink` for the page, followed by a `\penalty` and a `\par` (to exercise the page builder). We also have to make sure that this breakpoint doesn't affect how the next page is broken, so we make the penalty 9999 (10000 is infinite and thus not a legal breakpoint) and cancel out the `\kern` with a new `\kern`.

I don't think this is the *right* solution to the problem (that would be that the standard output routine has a feature for syncronizing with typesetting, as part of the preparations for switching output routine), but it's OK. Perhaps XOR will make it better.

```
157   \ifdim \pagetotal<\pagegoal \else
158     \dimen@=\pageshrink
159     \advance \dimen@ 1sp %
160     \kern\dimen@\penalty 9999\endgraf \kern-\dimen@
161   \fi
```

Start a new page if there is not enough room for the table head, foot, and one extra line.

```
162   \dimen@\pagetotal
163   \advance\dimen@ \ht\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
```

```
  \advance\dimen@ \dp\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
  \advance\dimen@ \ht\LT@foot
```

At this point I used to add `\ht\@arstrutbox` and `\dp\@arstrutbox` as a measure of a row size. However this can fail spectacularly for `p` columns which might be much larger. Previous versions could end up with the table starting with a foot, then a page break then a head *then* a 'first head'! So now measure the first line of the table accurately by `\vsplit`ting it out of the first chunk.

```
  \edef\LT@reset@vfuzz{\vfuzz\the\vfuzz\vbadness\the\vbadness\relax}%
  \vfuzz\maxdimen
  \vbadness\@M
  \setbox\tw@\copy\z@
  \setbox\tw@\vsplit\tw@ to \ht\@arstrutbox
  \setbox\tw@\vbox{\unvbox\tw@}%
  \LT@reset@vfuzz
  \advance\dimen@ \ht
        \ifdim\ht\@arstrutbox>\ht\tw@\@arstrutbox\else\tw@\fi
  \advance\dimen@\dp
        \ifdim\dp\@arstrutbox>\dp\tw@\@arstrutbox\else\tw@\fi
  \advance\dimen@ -\pagegoal
  \ifdim \dimen@>\z@
    \vfil\break
  \else
```

The LT output routine does not handle shrink on the page, which can cause the first page to be over-long, so forget it is there.

```
    \ifdim\pageshrink>\z@\pageshrink\z@\fi
  \fi
```

Store height of page minus table foot in `\@colroom`.

```
  \global\@colroom\@colht
```

If the foot is non empty, reduce the `\vsize` and `\@colroom` accordingly.

```
  \ifvoid\LT@foot\else
    \global\advance\vsize-\ht\LT@foot
    \global\advance\@colroom-\ht\LT@foot
    \dimen@\pagegoal\advance\dimen@-\ht\LT@foot\pagegoal\dimen@
    \maxdepth\z@
  \fi
  \tl_if_empty:eTF{\LTcaptype}
    {
      \MakeLinkTarget{LT@tables}
    }
    {
      \MakeLinkTarget{\LTcaptype}
    }
```

Put the table head on the page, and then switch to the new output routine.

```
  \ifvoid\LT@firsthead\copy\LT@head\else\box\LT@firsthead\fi\nobreak
  \UseTaggingSocket{tbl/longtable/head}
  \output{\LT@output}}
\ExplSyntaxOff
```

⟨@@=⟩

\endlongtable

```
202 ⟨@@=tbl⟩
203 \ExplSyntaxOn
```

Called by \end{longtable}.

```
204 \def\endlongtable{%
```

Essentially add a final \\. But as we now know the number of actual chunks, we first strip away all entries referring to a maximum entry beyond the table (this can only happen if a table has been shortened, or the table numbering has gone awry). In that case we at least start collecting valid new information with the last chunk of this table, by removing the width constraint.

```
205   \tbl_crcr:n {endlongtable}
206   \noalign{%
207     \UseTaggingSocket{tbl/longtable/finalize}
208     \let\LT@entry\LT@entry@chop
209     \xdef\LT@save@row{\LT@save@row}}%
210   \LT@echunk
211   \LT@start
212   \unvbox\z@
213   \LT@get@widths
```

Write the dummy row to the .aux file. Since V3.06, use .lta for letter.sty.

```
214   \if@filesw
215     {\let\LT@entry\LT@entry@write\immediate\write\@auxout{%
```

Since Version 3.02, longtable has used the internal counter \c@LT@tables rather than the LaTeX counter table. This information looks entirely different from version 3 information. Still, we don't need to rename the macro name because later code will consider the information to have no columns, and thus will throw the old data away.

```
216       \gdef\expandafter\noexpand
217         \csname LT@\romannumeral\c@LT@tables\endcsname
218           {\LT@save@row}}}%
219   \fi
```

At this point used to issue a warning if a \multicolumn has been set in draft mode.

\LT@mcw@rn

If the last chunk has different widths than the first, warn the user. Also trigger a warning to rerun LaTeX at the end of the document.

```
220   \ifx\LT@save@row\LT@@@@save@row
221   \else
222     \LT@warn{Column~ widths~ have~ changed\MessageBreak
223              in~
224                \tl_if_empty:eTF{\LTcaptype}
225                  {longtable~ \theLT@tables}
226                  {\LTcaptype\c_space_tl\use:c{the\LTcaptype}}
227              }
228     \LT@final@warn
229   \fi
```

Force one more go with the longtable output routine. Guard the special start of page value of \pagegoal.

```
  \endgraf\penalty -\LT@end@pen
  \ifvoid\LT@foot\else
    \global\advance\vsize\ht\LT@foot
    \global\advance\@colroom\ht\LT@foot
    \ifdim\pagegoal<\maxdimen
      \dimen@\pagegoal\advance\dimen@\ht\LT@foot\pagegoal\dimen@
    \fi
  \fi
```

Now close the group to return to the standard routine.

```
  \endgroup
```

Reset \@mparbottom to allow marginpars close to the end of the table.[5]

```
  \global\@mparbottom\z@
%  \pagegoal\vsize
  \endgraf\penalty\z@\addvspace\LTpost
```

Footnotes. As done in the multicol package.

```
  \ifvoid\footins\else\insert\footins{}\fi
  \UseTaggingSocket{tbl/vmode/end}%
}
\ExplSyntaxOff
```

⟨@@=⟩

10.6 Counting Columns

Columns are counted by examining \@preamble, rather than simply getting \@mkpream to increment the counter as it builds the preamble so that this package works with many of the packages which add extra column specifiers to LaTeX's standard ones.

Version 1 counted \@sharp's to calculate the number of columns, this was changed for Version 2 as it does not work with the NFSS. Now count &'s. (lfonts.new (and now the Standard LaTeX definition) defines \@tabclassz so that \@sharp is inside a group.)

\LT@nofcols Find the next &, then look ahead to see what is next.

```
\def\LT@nofcols#1&{%
  \futurelet\@let@token\LT@n@fcols}
```

\LT@n@fcols Add one, then stop at an \LT@nofcols or look for the next &. The \expandafter trick was added in Version 3, also the name changed from \@LT@nofcols to preserve the \LT@ naming convention.

```
\def\LT@n@fcols{%
  \advance\LT@cols\@ne
  \ifx\@let@token\LT@nofcols
    \expandafter\@gobble
  \else
    \expandafter\LT@nofcols
  \fi}
```

[5] This can not be the correct. However if it is omitted, there is a problem with marginpars, for example on page 3 of this document. Any Output Routine Gurus out there?

10.7 The \\ and \kill Commands

\LT@tabularcr The internal definition of \\. In the * form, insert a \nobreak after the next \cr (or \crcr).

This star form processing was finally added in v4.05. For the previous six or seven years the comment at this point said

> This definition also accepts *, which acts in the same way as \\. tabular does this, but longtable probably ought to make * prevent page breaking.

{\ifnum0=`}\fi added in version 3.01, required if the first entry is empty. The above in fact is not good enough, as with array package it can introduce a {} group in math mode, which changes the spacing. So use the following variant. Added in v3.14.

```
\protected\def\LT@tabularcr{%
  \relax\iffalse{\fi\ifnum0=`}\fi
  \@ifstar
```

TODO: as we replace crcr later in one case, we probably have to implement some further logic there!

```
    {\def\crcr{\LT@crcr\noalign{\nobreak}}\let\cr\crcr
     \LT@t@bularcr}%
    {\LT@t@bularcr}}
```

\LT@crcr

```
\let\LT@crcr\crcr
```

\LT@setprevdepth This will be redefined to set the \prevdepth at the start of a chunk.

```
\let\LT@setprevdepth\relax
```

\LT@t@bularcr

```
⟨@@=tbl⟩
\ExplSyntaxOn
\def\LT@t@bularcr{%
```

Increment the counter, and do tabular's \\ or finish the chunk.
The \expandafter trick was added in Version 3. Set the \prevdepth at the start of a new chunk. (Done here so not set in header chunks.)

```
  \global\advance\LT@rows\@ne
  \ifnum\LT@rows=\LTchunksize
```

At the end of the chunk \\ is doing something special and so we lose \tbl_count_missing_cells:n. Below is about the right place to add it do this code branch.

```
    \tbl_count_missing_cells:n {echunk}
    \gdef\LT@setprevdepth{%
      \prevdepth\z@
      \global\let\LT@setprevdepth\relax}%
    \expandafter\LT@xtabularcr
  \else
    \ifnum0=`{}\fi
    \expandafter\LT@LL@FM@cr
  \fi}
\ExplSyntaxOff
⟨@@=⟩
```

\LT@xtabularcr This just looks for an optional argument.

```
\def\LT@xtabularcr{%
  \@ifnextchar[\LT@argtabularcr\LT@ntabularcr}
```

\LT@ntabularcr The version with no optional argument. `\ifnum0=‘{\fi}` added in version 3.01. Changed in 3.14.

```
\def\LT@ntabularcr{%
  \ifnum0=‘{}\fi
  \LT@echunk
  \LT@start
  \unvbox\z@
  \LT@get@widths
  \LT@bchunk}
```

\LT@argtabularcr The version with an optional argument. `\ifnum0=‘{\fi}` added in version 3.01. Changed in 3.14.

```
\def\LT@argtabularcr[#1]{%
  \ifnum0=‘{}\fi
  \ifdim #1>\z@
    \unskip\@xargarraycr{#1}%
  \else
    \@yargarraycr{#1}%
  \fi
```

Add the dummy row, and finish the `\halign`.

```
  \LT@echunk
  \LT@start
  \unvbox\z@
  \LT@get@widths
  \LT@bchunk}
```

\LT@echunk This ends the current chunk, and removes the dummy row.

```
\def\LT@echunk{%
  \crcr\LT@save@row\cr\egroup
  \global\setbox\LT@gbox\lastbox
```

The following line was added in v4.05. longtable relies on `\lineskip` glue (which is 0pt) to provide break points between each row so the table may be split into pages.

Previous releases left the `\lineskip` glue at the end of each chunk that had been added when the dummy row was added. There was no glue at the start of the next chunk as TEX normally does not put `\lineskip` glue at the top of a box. This meant that normally the chunks fitted together perfectly, however `\noalign` material at a chunk boundary came before the first row of the next chunk but after the lineskip glue at the end of this chunk. This is the wrong place, e.g., it means even a `\penalty10000` does not stop a break as the `\lineskip` glue in the previous item on the list provides a legal breakpoint. So now remove the `\lineskip` glue that was before the dummy row and introduce `\LT@setprevdepth` to set the `\prevdepth` at the start of the next chunk, to make sure `\lineskip` glue is added later.

```
    \unskip
  \egroup}
```

\LT@entry We here give the 'basic' definition of \LT@entry, namely that used in alignment templates. It has a \kern only if the maximum is imposed from a different chunk. The \ifhmode test reveals the first entry, when we don't want to add an &.

```
\def\LT@entry#1#2{%
  \ifhmode\@firstofone{&}\fi\omit
  \ifnum#1=\c@LT@chunks
  \else
    \kern#2\relax
  \fi}
```

\LT@entry@chop This definition for the argument of \LT@save@row is used to scrap all those maxima which could not be verified because they occur after the end of the table. This can happen only if a table has been shortened (or the sequencing got mixed up) since the previous run. Note that this is premature: the last chunk still is going to be set, and with the chopped limits.

```
\def\LT@entry@chop#1#2{%
  \noexpand\LT@entry
    {\ifnum#1>\c@LT@chunks
       1}{0pt%
     \else
       #1}{#2%
     \fi}}
```

\LT@entry@write To write an entry for the aux file, we use a slightly surprising definition which has the sole purpose of avoiding overfull lines (which might break TEX's limits when reading the aux file, probably you'd need to have a few hundred columns before this happened but...).

```
\def\LT@entry@write{%
  \noexpand\LT@entry^^J%
  \@spaces}
```

\LT@kill This ends the current chunk as above, but strips off two rows, the 'dummy row' and the 'killed row' before starting the next chunk. Since V3.04, the old chunk is reboxed at the start of the box containing the next chunk. This allows \kill to be used in headers, which must be processed in a single box.

```
\def\LT@kill{%
  \LT@echunk
  \LT@get@widths
  \expandafter\LT@rebox\LT@bchunk}
```

\LT@rebox Drop the old chunk (box0) back at the top of the new chunk, removing the killed row. This macro added at V3.04.

```
\def\LT@rebox#1\bgroup{%
  #1\bgroup
  \unvbox\z@
  \unskip
  \setbox\z@\lastbox}
```

10.8 The Dummy Row

The dummy row is kept inside of the macro \LT@save@row.

\LT@blank@row
\LT@build@blank
Create a blank row if we are not using the info in the .aux file.

```
\def\LT@blank@row{%
  \xdef\LT@save@row{\expandafter\LT@build@blank
    \romannumeral\number\LT@cols 001 }}
```

Whoops! What's that supposed to be? A drop-in replacement for the first task of Appendix D in the TEXbook. The \romannumeral produces \LT@cols instances of m followed by i. The below macro then replaces the ms by appropriate entries.

```
\def\LT@build@blank#1{%
  \if#1m%
    \noexpand\LT@entry{1}{0pt}%
    \expandafter\LT@build@blank
  \fi}
```

\LT@make@row
Prior to version 4, by default did not use information in the .aux file but now we can define \LT@make@row to use the .aux file, even on the 'draft' passes.

```
\def\LT@make@row{%
  \global\expandafter\let\expandafter\LT@save@row
    \csname LT@\romannumeral\c@LT@tables\endcsname
  \ifx\LT@save@row\relax
    \LT@blank@row
```

Now a slightly difficult part comes. Before we decide making the template from the .aux file info we check that the number of fields has remained the same. If it hasn't, either the table format has changed, or we have the wrong table altogether. In both cases, we decide to better drop all gathered information and start over.

The expansion between !... ! below will be empty if the number of \LT@entry macros including arguments in \LT@save@row is equal to \LT@cols. If it is not empty, we throw the row away and start from scratch.

```
  \else
    {\let\LT@entry\or
     \if!%
         \ifcase\expandafter\expandafter\expandafter\LT@cols
         \expandafter\@gobble\LT@save@row
         \or
         \else
           \relax
         \fi
        !%
     \else
       \aftergroup\LT@blank@row
     \fi}%
  \fi}
```

\setlongtables
Redefine \LT@make@row to use information in the .aux file, if there is a saved row for this table with the right number of columns.

Since Version 3.02, longtable has used the internal counter \c@LT@tables rather than the LATEX counter table. The warning message was added at V3.04, as was the \global, to stop save-stack overflow.

Since Version 4.01 \setlongtables does nothing as it is not needed, but is defined as \relax for the benefit of old documents.

```
\let\setlongtables\relax
```

\LT@get@widths This is the heart of longtable. If it were not for the table head and foot, this macro together with the modified \\ command would form the basis of quite a simple little package file for long tables. It is closely modelled on the \endvrulealign macro of appendix D of the TEXbook.

```
\def\LT@get@widths{%
```

\global added at V3.04, to stop save-stack overflow.

Loop through the last row, discarding glue, and saving box widths. At V3.04 changed the scratch box to 2, as the new \kill requires that \box0 be preserved.

```
  \setbox\tw@\hbox{%
    \unhbox\LT@gbox
    \let\LT@old@row\LT@save@row
    \global\let\LT@save@row\@empty
    \count@\LT@cols
    \loop
      \unskip
      \setbox\tw@\lastbox
    \ifhbox\tw@
      \LT@def@row
      \advance\count@\m@ne
    \repeat}%
```

Remember the widths if we are in the first chunk.

```
  \ifx\LT@@save@row\@undefined
    \let\LT@@save@row\LT@save@row
  \fi}
```

\LT@def@row Add a column to the dummy row. Name changed from \defLT@save@row in Version 3, to preserve the \LT@ naming convention.

```
\def\LT@def@row{%
```

We start by picking the respective entry from our old row. These redefinitions of \LT@entry are local to the group started in \LT@get@widths.

```
  \let\LT@entry\or
  \edef\@tempa{%
    \ifcase\expandafter\count@\LT@old@row
    \else
      {1}{0pt}%
    \fi}%
```

Now we tack the right combination in front of \LT@save@row:

```
  \let\LT@entry\relax
  \xdef\LT@save@row{%
    \LT@entry
    \expandafter\LT@max@sel\@tempa
    \LT@save@row}}
```

\LT@max@sel And this is how to select the right combination. Note that we take the old maximum information only if the size does not change in *either* direction. If the size has grown, we of course have a new maximum. If the size has shrunk, the old maximum (which was explicitly not enforced because of being in the current chunk) is invalid, and we start with this chunk as the new size. Note that even in the case of equality we *must* use the \the\wd\tw@ construct instead of #2 because #2

might be read in from the file, and so could have \catcode 11 versions of p and t in it which we want to be replaced by their 'proper' \catcode 12 versions.

```
\def\LT@max@sel#1#2{%
  {\ifdim#2=\wd\tw@
     #1%
   \else
     \number\c@LT@chunks
   \fi}%
  {\the\wd\tw@}}
```

10.9 The \hline Command

\LT@hline \hline and \hline\hline both produce *two* lines. The only difference being the glue and penalties between them. This is so that a page break at a \hline produces a line on both pages.[6] Also this \hline is more like a \cline{1-\LT@cols}. tabular's \hline would draw lines the full width of the page.

```
\def\LT@hline{%
  \noalign{\ifnum0=`}\fi
    \penalty\@M
    \futurelet\@let@token\LT@@hline}
```

\LT@@hline This code is based on \cline. Two copies of the line are produced, as described above.

```
⟨@@=tbl⟩
\ExplSyntaxOn
\def\LT@@@@hline{%
  \ifx\@let@token\hline
    \global\let\@gtempa\@gobble
    \gdef\LT@sep{\penalty-\@medpenalty\vskip\doublerulesep}%
  \else
    \global\let\@gtempa\@empty
    \gdef\LT@sep{\penalty-\@lowpenalty\vskip-\arrayrulewidth}%
  \fi
  \ifnum0=`{\fi}%
  \multispan\LT@cols
     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
```

Don't update the row counter, or rather undo the update done in \everycr:

```
  \noalign{
    \tbl_gdecr_row_count:
    \LT@sep}
  \multispan\LT@cols
     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
```

Same here.

```
  \noalign{
    \tbl_gdecr_row_count:
    \penalty\@M}
  \@gtempa}
⟨@@=⟩
```

[6] longtable has always done this, but perhaps it would be better if hlines were *omitted* at a page break, as the head and foot usually put a hline here anyway.

10.10 Captions

`\LT@caption` The caption is `\multicolumn{\LT@cols}{c}{`*⟨a parbox with the table's caption⟩*`}`

```
\def\LT@caption{%
  \noalign\bgroup
    \@ifnextchar[{\egroup\LT@c@ption\@firstofone}\LT@capti@n}
```

`\LT@c@ption` Caption command (with [optional argument]). `\protect` added in Version 3. `\fnum@table` added at V3.05.

```
\def\LT@c@ption#1[#2]#3{
  \tl_if_empty:eTF{\LTcaptype}
    {\LT@makecaption\@gobble{}{#3}}
    {\LT@makecaption#1{\csname fnum@\LTcaptype\endcsname}{#3}
    \def\@tempa{#2}
    \ifx\@tempa\@empty\else
      {\let\\\space
      \addcontentsline
        {\@nameuse{ext@\LTcaptype}}
        {\LTcaptype}
        {\protect\numberline{\@nameuse{the\LTcaptype}}{#2}}}
    \fi}}
\ExplSyntaxOff
```

`\LT@capti@n` Caption command (no [optional argument])

```
\def\LT@capti@n{%
  \@ifstar
    {\egroup\LT@c@ption\@gobble[]}%
    {\egroup\@xdblarg{\LT@c@ption\@firstofone}}}
```

`\LT@makecaption` Put the caption in a box of width 0pt, so that it never affects the column widths. Inside that is a `\parbox` of width `\LTcapwidth`.

```
\def\LT@makecaption#1#2#3{%
  \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]\LTcapwidth{%
```

Based on article class `\@makecaption`, `#1` is `\@gobble` in star form, and `\@firstofone` otherwise.

```
    \reset@font
    \sbox\@tempboxa{#1{#2: }#3}%
    \ifdim\wd\@tempboxa>\hsize
      #1{#2: }#3%
    \else
      \hbox to\hsize{\hfil\box\@tempboxa\hfil}%
    \fi
    \endgraf\vskip\baselineskip}%
  \hss}}}
```

10.11 The Output Routine

The method used here for interfacing a special purpose output routine to the standard LaTeX routine is lifted straight out of F. Mittelbach's multicol package.

`\LT@output` Actually this is not so bad, with FM leading the way.

```
⟨@@=tbl⟩
```

```
\ExplSyntaxOn
\def\LT@output{%
  \ifnum\outputpenalty <-\@Mi
    \ifnum\outputpenalty > -\LT@end@pen
```

If this was a float or a marginpar we complain.

```
      \LT@err{floats~ and~ marginpars~ not~ allowed~ in~ a~ longtable}\@ehc
    \else
```

We have reached the end of the table, on the scroll at least,

```
      \setbox\z@\vbox{\unvbox\@cclv}%
      \ifdim \ht\LT@lastfoot>\ht\LT@foot
```

The last foot might not fit, so:[7]

```
        \dimen@\pagegoal
        \advance\dimen@\ht\LT@foot
        \advance\dimen@-\ht\LT@lastfoot
        \ifdim\dimen@<\ht\z@
          \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
          \@makecol
          \@outputpage
          \global\vsize\@colroom
          \setbox\z@\vbox{\box\LT@head}%
```

End of `\ifdim\dimen@<\ht\@cclc`.

```
        \fi
```

End of `\ifdim \ht\LT@lastfoot > \ht\LT@foot`.

```
      \fi
```

Reset `\@colroom`.

```
%       \global\@colroom\@colht
%       \global\vsize\@colht
```

Put the last page of the table on to the main vertical list.

```
        \unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi
```

Handle foot box when tagging:

```
        \UseTaggingSocket{tbl/longtable/foot}
```

End of `\ifnum\outputpenalty > -\LT@end@pen`.

```
    \fi
```

Else `\outputpenalty > -\@Mi`.

```
  \else
```

If we have not reached the end of the table,

```
    \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
```

Handle foot box when tagging:

```
    \UseTaggingSocket{tbl/longtable/foot}
    \@makecol
    \@outputpage
```

Reset `\vsize`.

```
      \global\vsize\@colroom
```

[7] An alternative would be to vsplit off a bit of the last chunk, so that the last page did not just have head and foot sections, but it is hard to do this in a consistent manner.

Put the head at the top of the next page.

```
    \copy\LT@head\nobreak
```

End of `\ifnum\outputpenalty <-\@Mi`.

```
  \fi}
\ExplSyntaxOff
⟨@@=⟩
```

10.12 Commands for the table head and foot

`\LT@end@hd@ft` The core of `\endhead` and friends. Store the current chunk in the box specified by `#1`. Issue an error if the table has already started. Then start a new chunk.

```
⟨@@=tbl⟩
\ExplSyntaxOn
\def\LT@end@hd@ft#1{%
```

This command is used to store the head and foot boxes. We need to retrieve and store the row so that we can clean up the structure in the finalize code.

To handle missing columns in the header we need this:

```
  \tbl_if_row_was_started:TF
    {
```

TODO: This is exposing internal counters, so it should be encapsulated in some interface command (but I'm not sure what that should be called, so not done yet.

```
      \tbl_count_missing_cells:n {head/foot}
      \int_step_inline:nn
        { \LT@rows + 1 }
        {
          \seq_gput_left:ce
            {g_@@_\cs_to_str:N #1 _rows_seq }
            { \int_eval:n {\g_@@_row_int + 1 - ##1 } }
        }
```

We also have to set the chunk rows to its max value before calling `\LTechunk` so that we don't get extra increments of the main row counter due to `\everycr`.

```
      \int_gset:Nn \LT@rows { \LTchunksize }
    }
```

If we are still in column zero then we had an empty `\endhead` and so making any assignment, etc., would start a row — something we don't want. To get out of this trap we run `\crcr` (which would normally come inside `\LT@echunk`. That will then trigger `\everycr` and update row counter unnecessarily, but now we have a defined state, so we can use `\noalign` to undo that. We also change `\LT@rows` so that further `\cr`s do not do any harm (as explained above.

The `\crcr` inside `\LT@echunk` will be bypassed in that case as we have just executed a `\crcr` and are still in scanning modus for `\omit` or `\noalign`.

```
    {
      \crcr
      \noalign{
      \int_step_inline:nn
        { \LT@rows }
        {
          \seq_gput_left:ce
```

```
            {g_@@_\cs_to_str:N #1 _rows_seq }
            { \int_eval:n {\g_@@_row_int - ##1 } }
        }
        \tbl_gdecr_row_count:         % undo the increment
        \int_gset:Nn \LT@rows { \LTchunksize }
      }
    }
  \LT@echunk
```

Changed from `\relax` to `\endgraf` at V3.04, see `\LT@start`.

```
  \ifx\LT@start\endgraf
    \LT@err
     {Longtable~ head~ or~ foot~ not~ at~ start~ of~ table}%
     {Increase~ LTchunksize}%
  \fi
  \setbox#1\box\z@
  \@@_trace:n {-->>~ Saving~\noexpand#1}
  \LT@get@widths
  \LT@bchunk}
\ExplSyntaxOff
```

⟨@@=⟩

`\endfirsthead` Call `\LT@end@hd@ft` with the appropriate box.

```
\def\endfirsthead{\LT@end@hd@ft\LT@firsthead}
```

`\endhead`

```
\def\endhead{\LT@end@hd@ft\LT@head}
```

`\endfoot`

```
\def\endfoot{\LT@end@hd@ft\LT@foot}
```

`\endlastfoot`

```
\def\endlastfoot{\LT@end@hd@ft\LT@lastfoot}
```

10.13 The `\multicolumn` command

Earlier versions needed a special 'draft' form of `\multicolumn`. This is not needed in version 4, and so these commands have been removed.

`\LTmulticolumn`

`\LT@mcwarn`

10.14 Footnotes

The standard `\footnote` command works in a `c` column, but we need to modify the definition in a `p` column to overcome the extra level of boxing. These macros are based on the array package, but should be OK for the standard tabular.

`\LT@startpbox` Add extra code to switch the definition of `\@footnotetext`.

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

`\LT@endpbox` After the parbox is closed, expand `\LT@p@ftn` which will execute a series of `\footnotetext[`⟨*num*⟩`]{`⟨*note*⟩`}` commands. After being lifted out of the parbox, they can migrate on their own from here.

```
\def\LT@endpbox{%
  \@finalstrut\@arstrutbox
  \color@endgroup
  \egroup
  \the\LT@p@ftn
  \global\LT@p@ftn{}%
  \hfil}
```

`\LT@p@ftntext` Inside the 'p' column, just save up the footnote text in a token register.

```
\long\def\LT@p@ftntext#1{%
  \edef\@tempa{\the\LT@p@ftn\noexpand\footnotetext[\the\c@footnote]}%
  \global\LT@p@ftn\expandafter{\@tempa{#1}}}%
```

Some variables need for the tagging support.

```
⟨@@=tbl⟩
\ExplSyntaxOn
   \seq_new:N \g_@@_LT@firsthead_rows_seq
   \seq_new:N \g_@@_LT@head_rows_seq
   \seq_new:N \g_@@_LT@lastfoot_rows_seq
   \seq_new:N \g_@@_LT@foot_rows_seq
\ExplSyntaxOff
⟨@@=⟩
⟨/package⟩
```