

# The longtable package\*

David Carlisle†

1996/05/24

## Abstract

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

## List of Tables

1	An optional table caption (used in the list of tables) . . . . .	1
2	A floating table . . . . .	4
3	A difficult \multicolumn combination: pass 1 . . . . .	6
4	A difficult \multicolumn combination: pass 2 . . . . .	6
5	A difficult \multicolumn combination: pass 3 . . . . .	6
6	A difficult \multicolumn combination: pass 4 . . . . .	6
7	A summary of longtable commands . . . . .	9

## 1 Introduction

`longtable` 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  $\text{\TeX}$ 's standard page-breaking algorithm. It also shares some features with the `table` environment. In particular it uses 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 8.

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

Table 1: A long table

*	This part appears at the top of the table		*
*	FIRST	SECOND	*
*	longtable columns are specified	in the	*
*	This goes at the	bottom.	*

\*This file has version number v4.04, last revised 1996/05/24.

†The new algorithm for aligning 'chunks' of a table used in version 4 of this package was devised coded and documented by David Kastrup, `dak@pool.informatik.rwth-aachen.de`.



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 T <sub>E</sub> X 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 <sup>1</sup> of lines	like this. *
* Lots of lines	like this <sup>2</sup> *
* 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 T<sub>E</sub>X multi-page tables, it is necessary to break up the table into smaller chunks, so that T<sub>E</sub>X 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}`.<sup>3</sup> These chunks do not affect page breaking, thus if you are using a T<sub>E</sub>X with a lot of memory, you can set `LTchunksize` to be several pages of the table. T<sub>E</sub>X will run faster with a large `LTchunksize`. 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

<sup>1</sup>This is a footnote.

<sup>2</sup>`longtable` takes special precautions, so that footnotes may also be used in ‘p’ columns.

<sup>3</sup>You can also use the plain T<sub>E</sub>X syntax `\LTchunksize=10`.

A	tabular	environment
within	a floating	table

Table 2: A floating table

foot (see below), the `LTchunksizes` must be at least as large as the number of rows in each of the head or foot sections.

This document specifies `\setcounter{LTchunksizes}{10}`. 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, using a new algorithm<sup>4</sup> and so `\setlongtables` is no longer needed. It is defined (but does nothing) for the benefit of old documents that use it.

### 3 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 `LTchunksizes` 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 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.

`\endhead`  
`\endfirsthead`

`\endfoot`  
`\endlastfoot`

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}}`.

`\caption`

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

---

<sup>4</sup>Due to David Kastrup.

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.

## 4 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 `LTchunksizes` 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 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 too 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

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

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.

## 5 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{...}}...}
```

## 6 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  $\text{\LaTeX}$ .
- 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.)



## 7 Summary

Table 7: A summary of longtable commands

<b>Parameters</b>		
<code>\LTleft</code>	Glue to the left of the table.	<code>(\fill)</code>
<code>\LTright</code>	Glue to the right of the table.	<code>(\fill)</code>
<code>\LTpre</code>	Glue before the the table.	<code>(\bigskipamount)</code>
<code>\LTpost</code>	Glue after the the table.	<code>(\bigskipamount)</code>
<code>\LTcapwidth</code>	The width of a parbox containing the caption.	<code>(4in)</code>
<code>LTchunksize</code>	The number of rows per chunk.	<code>(20)</code>
<b>Setlongtables</b>		
<code>\setlongtables</code>	Use column widths from the previous run.	
<b>Optional arguments to <code>\begin{longtable}</code></b>		
<code>none</code>	Position as specified by <code>\LTleft</code> and <code>\LTright</code> .	
<code>[c]</code>	Centre the table.	
<code>[l]</code>	Place the table flush left.	
<code>[r]</code>	Place the table flush right.	
<b>Commands to end table rows</b>		
<code>\\</code>	Specifies the end of a row.	
<code>\tabularnewline</code>	Alternative to <code>\\</code> for use in the scope of <code>\raggedright</code> and similar commands that redefine <code>\\</code> .	
<code>\kill</code>	Row is 'killed', but is used in calculating widths.	
<code>\endhead</code>	Specifies rows to appear at the top of every page.	
<code>\endfirsthead</code>	Specifies rows to appear at the top the first page.	
<code>\endfoot</code>	Specifies rows to appear at the bottom of every page.	
<code>\endlastfoot</code>	Specifies rows to appear at the bottom of the last page.	
<b>longtable caption commands</b>		
<code>\caption{foo}</code>	Caption 'Table ?: foo', and a 'foo' entry in the list of tables.	
<code>\caption[bar]{foo}</code>	Caption 'Table ?: foo', and a 'bar' entry in the list of tables.	
<code>\caption[] {foo}</code>	Caption 'Table ?: foo', but no entry in the list of tables.	
<code>\caption*{foo}</code>	Caption 'foo', but no entry in the list of tables.	
<b>Other commands available inside longtable</b>		
<code>\newpage</code>	Force a page break.	
<code>\footnote</code>	Footnotes, but may not be used in the table head & foot.	
<code>\footnotemark</code>	Footnotemark, May be used in the table head & foot.	
<code>\footnotetext</code>	Footnote text, Use in the table body.	

## 8 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}

```

## 9 The Macros

1 (\*package)

### 9.1 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 widths have changed. Rerun LaTeX.\@gobbletwo}}%`  
7 `\global\let\LT@final@warn\relax`

### 9.2 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` The next options are just alternative syntax for the `\setlongtables` declaration.  
`final`  
13 `\DeclareOption{set}{}%`  
14 `\DeclareOption{final}{}%`  
15 `\ProcessOptions`

### 9.3 User Setable Parameters

`\LTleft` Glue to the left and right of the table, default `\fill` (ie centred).  
`\LTright` 16 `\newskip\LTleft` `\LTleft=\fill`  
17 `\newskip\LTright` `\LTright=\fill`

`\LTpre` Glue before and after the longtable. `\bigskip` by default.  
`\LTpost` 18 `\newskip\LTpre` `\LTpre=\bigskipamount`  
19 `\newskip\LTpost` `\LTpost=\bigskipamount`

`\LTchunksize` Chunk size (The number of rows taken per `\halign`). Default 20.  
20 `\newcount\LTchunksize` `\LTchunksize=20`

`\c@LTchunksiz` Added in V3.07 to allow the  $\LaTeX$  syntax `\setcounter{LTchunksiz}{10}`.  
`21 \let\c@LTchunksiz\LTchunksiz`

`\LTcapwidth` Width of the `\parbox` containing the caption. Default 4in.  
`22 \newdimen\LTcapwidth \LTcapwidth=4in`

### 9.4 Internal Parameters

`\LT@head` Boxes for the table head and foot.  
`\LT@firsthead` `23 \newbox\LT@head`  
`\LT@foot` `24 \newbox\LT@firsthead`  
`\LT@lastfoot` `25 \newbox\LT@foot`  
`26 \newbox\LT@lastfoot`

`\LT@cols` Counter for number of columns.  
`27 \newcount\LT@cols`

`\LT@rows` Counter for rows up to chunksize.  
`28 \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.  
`29 \newcounter{LT@tables}`

`\c@LT@chunks` We need to count through the chunks of our tables from Version 4 on.  
`30 \newcounter{LT@chunks}[LT@tables]`

`\c@table` If the table counter is not defined (eg in letter style), define it. (Added in V3.06.)  
`\fnum@table` `31 \ifx\c@table\undefined`  
`\tablename` `32 \newcounter{table}`  
`33 \def\fnum@table{\tablename~\thetable}`  
`34 \fi`  
`35 \ifx\tablename\undefined`  
`36 \def\tablename{Table}`  
`37 \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.

```
38 \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.

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

## 9.5 The longtable environment

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

```
40 \def\longtable{%
41   \par
42   \ifx\multicols\@undefined
43   \else
44     \ifnum\col@number>\@ne
45       \@twocolumntrue
46     \fi
47   \fi
48   \if@twocolumn
49     \LT@err{longtable not in 1-column mode}\@ehc
50   \fi
51   \begingroup
```

Check for an optional argument.

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

`\LT@array` 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.

```
53 \def\LT@array[#1]#2{%
54   \refstepcounter{table}\stepcounter{LT@tables}%
```

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

```
55   \if l#1%
56     \LTleft\z@ \LTright\fill
57   \else\if r#1%
58     \LTleft\fill \LTright\z@
59   \else\if c#1%
60     \LTleft\fill \LTright\fill
61   \fi\fi\fi
```

Set up these internal commands for `longtable`.

```
\global\let\LT@mcw@rn\relax
62 \let\LT@acol\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`.

```
63 \let\LT@@tabarray\@tabarray
64 \let\LT@@hl\hline
65 \def\@tabarray{%
66   \let\hline\LT@@hl
```

.....longtable.sty .....

```
\let\multicolumn\LT@mccl
67 \LT@@tabarray}%
68 \let\\\LT@tabularcr\let\tabularnewline\\%
69 \def\newpage{\noalign{\break}}%
70 \let\hline\LT@hline \let\kill\LT@kill\let\caption\LT@caption
71 \@tempdima\ht\strutbox
```

Set up internal commands according to Lamport or Mittelbach.

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

Initialise these commands as in `tabular` from the `LATEX` kernel.

```
73 \let\@acol\@tabacol
74 \let\@classz\@tabclassz \let\@classiv\@tabclassiv
75 \def\@startpbox{\vtop\LT@startpbox}\let\LT@LL@FM@cr\@tabularcr
76 \else
```

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

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

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

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

Interpret the preamble argument.

```
86 \beginngroup
87 \@mkpream{#2}%
```

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!

```
88 \xdef\LT@bchunk{%
89 \global\advance\c@LT@chunks\@ne
90 \global\LT@rows\z@\setbox\z@\vbox\bgroup
91 \tabskip\LT@left\halign to\hsize\bgroup
92 \tabskip\z@ \@arstrut \@preamble \tabskip\LT@right \cr}%
93 \endgroup
```

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

```
94 \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.

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

A few more internal commands for `longtable`.

```
96 \let\@endpbox\LT@endpbox
97 \m@th\let\par\@empty
98 \everycr{\lineskip\z@\baselineskip\z@
```

Start the first chunk.

```
99 \LT@bchunk}
```

`\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.

```
100 \def\LT@start{%
101 \let\LT@start\endgraf
102 \endgraf\penalty\z@\vskip\LTpre
```

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

```
103 \dimen@\pagetotal
104 \advance\dimen@ \ht\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
105 \advance\dimen@ \dp\ifvoid\LT@firsthead\LT@head\else\LT@firsthead\fi
106 \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 `\vsplitting` it out of the first chunk.

```
107 \dimen@ii\vfuzz
108 \vfuzz\maxdimen
109 \setbox\tw@\copy\z@
110 \setbox\tw@\vsplit\tw@ to \ht\@arstrutbox
111 \setbox\tw@\vbox{\unvbox\tw@}%
112 \vfuzz\dimen@ii
113 \advance\dimen@ \ht
114 \ifdim\ht\@arstrutbox>\ht\tw@\@arstrutbox\else\tw@\fi
115 \advance\dimen@\dp
116 \ifdim\dp\@arstrutbox>\dp\tw@\@arstrutbox\else\tw@\fi
117 \advance\dimen@ -\pagegoal
118 \ifdim \dimen@>\z@\vfil\break\fi
```

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

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

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

```
120 \ifvoid\LT@foot\else
121 \advance\vsize-\ht\LT@foot
122 \global\advance\@colroom-\ht\LT@foot
123 \dimen@\pagegoal\advance\dimen@-\ht\LT@foot\pagegoal\dimen@
124 \maxdepth\z@
125 \fi
```

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

```
126 \ifvoid\LT@firsthead\copy\LT@head\else\box\LT@firsthead\fi
127 \output{\LT@output}}
```

`\endlongtable` Called by `\end{longtable}`.

```
128 \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.

```

129 \crrc
130 \noalign{%
131   \let\LT@entry\LT@entry@chop
132   \xdef\LT@save@row{\LT@save@row}}%
133 \LT@echunk
134 \LT@start
135 \unvbox\z@
136 \LT@get@widths

```

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

```

137 \if@filesw
138   {\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.

```

139   \gdef\expandafter\noexpand
140   \csname LT@\romannumeral\c@LT@tables\endcsname
141   {\LT@save@row}}}%
142 \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.

```

143 \ifx\LT@save@row\LT@@save@row
144 \else
145   \LT@warn{Column widths have changed\MessageBreak
146           in table \thetable}%
147   \LT@final@warn
148 \fi

```

Force one more go with the `longtable` output routine.

```
149 \endgraf\penalty -\LT@end@pen
```

Now close the group to return to the standard routine.

```
150 \endgroup
```

Reset `\@mparbottom` to allow marginpars close to the end of the table.<sup>5</sup>

```

151 \global\@mparbottom\z@
152 \pagegoal\vsizel
153 \endgraf\penalty\z@\addvspace\LTpost

```

Footnotes. As done in the `multicol` package.

```
154 \ifvoid\footins\else\insert\footins{}\fi}
```

<sup>5</sup>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?



## 9.6 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 L<sup>A</sup>T<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X definition) defines `@tabclassz` so that `@sharp` is inside a group.)

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

```
155 \def\LT@nofcols#1&{%
156   \futurelet\@let@token\LT@nofcols}
```

`\LT@nofcols` 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.

```
157 \def\LT@nofcols{%
158   \advance\LT@cols\@ne
159   \ifx\@let@token\LT@nofcols
160     \expandafter\@gobble
161   \else
162     \expandafter\LT@nofcols
163   \fi}
```

## 9.7 The `\` and `\kill` Commands

`\LT@tabularcr` The internal definition of `\`.

```
164 \def\LT@tabularcr{%
Increment the counter, and do tabular's \ or finish the chunk.
The \expandafter trick was added in Version 3.
```

```
165   \global\advance\LT@rows\@ne
166   \ifnum\LT@rows=\LT@chunksiz
167     \expandafter\LT@tabularcr
168   \else
169     \expandafter\LT@LL@FM@cr
170   \fi}
```

`\LT@t@bularcr` 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.

```
171 \def\LT@t@bularcr{%
172   \relax\iffalse{\fi\ifnum0='}\fi
173   \@ifstar\LT@xtabularcr\LT@xtabularcr}
```

`\LT@xtabularcr` This justs looks for an optional argument.

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

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

```
176 \def\LT@ntabularcr{%
177   \ifnum0='{\fi
178   \LT@echunk
179   \LT@start
180   \unvbox\z@
181   \LT@get@widths
182   \LT@bchunk}
```

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

```
183 \def\LT@argtabularcr[#1]{%
184   \ifnum0='{\fi
185   \ifdim #1>\z@
186     \unskip\@xargarraycr{#1}%
187   \else
188     \@yargarraycr{#1}%
189   \fi
```

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

```
190   \LT@echunk
191   \LT@start
192   \unvbox\z@
193   \LT@get@widths
194   \LT@bchunk}
```

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

```
195 \def\LT@echunk{%
196   \crr\LT@save@row\cr\egroup
197   \global\setbox\@ne\lastbox
198   \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 `&`.

```
199 \def\LT@entry#1#2{%
200   \ifhmode\@firstofone{&}\fi\omit
201   \ifnum#1=\c@LT@chunks
202   \else
203     \kern#2\relax
204   \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.

```
205 \def\LT@entry@chop#1#2{%
206   \noexpand\LT@entry
207     {\ifnum#1>\c@LT@chunks
208      1}{0pt%
209     \else
```

```
210     #1}{#2%
211     \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...).

```
212 \def\LT@entry@write{%
213   \@percentchar^^J\@spaces
214   \noexpand\LT@entry}
```

`\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.

```
215 \def\LT@kill{%
216   \LT@echunk
217   \LT@get@widths
218   \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.

```
219 \def\LT@rebox#1\bgroup{%
220   #1\bgroup
221   \unvbox\z@
222   \unskip
223   \setbox\z@\lastbox}
```

## 9.8 The Dummy Row

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

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

```
\LT@build@blank 224 \def\LT@blank@row{%
225   \xdef\LT@save@row{\expandafter\LT@build@blank
226     \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  $\TeX$ book. The `\romannumeral` produces `\LT@cols` instances of `m` followed by `i`. The below macro then replaces the `ms` by appropriate entries.

```
227 \def\LT@build@blank#1{%
228   \if#1m%
229     \noexpand\LT@entry{1}{0pt}%
230     \expandafter\LT@build@blank
231   \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.

```
232 \def\LT@make@row{%
233   \global\expandafter\let\expandafter\LT@save@row
234   \csname LT@\romannumeral\c@LT@tables\endcsname
235   \ifx\LT@save@row\relax
236     \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.

```

237 \else
238   {\let\LT@entry\or
239    \if!%
240     \ifcase\expandafter\expandafter\expandafter\LT@cols
241     \expandafter@gobble\LT@save@row
242     \or
243     \else
244     \relax
245     \fi
246     !%
247   \else
248   \aftergroup\LT@blank@row
249   \fi}%
250 \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 L<sup>A</sup>T<sub>E</sub>X 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.

```

251 \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 T<sub>E</sub>Xbook.

```

252 \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.

```

253 \setbox\tw@\hbox{%
254   \unhbox@ne
255   \let\LT@old@row\LT@save@row
256   \global\let\LT@save@row\@empty
257   \count@\LT@cols
258   \loop
259     \unskip
260     \setbox\tw@\lastbox
261   \ifhbox\tw@
262     \LT@def@row
263     \advance\count@\m@ne
264   \repeat}%

```

Remember the widths if we are in the first chunk.

```
265 \ifx\LT@@save@row\undefined
266 \let\LT@@save@row\LT@save@row
267 \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.

```
268 \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.

```
269 \let\LT@entry\or
270 \edef\@tempa{%
271 \ifcase\expandafter\count@\LT@old@row
272 \else
273 {1}{0pt}%
274 \fi}%
```

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

```
275 \let\LT@entry\relax
276 \xdef\LT@save@row{%
277 \LT@entry
278 \expandafter\LT@max@sel\@tempa
279 \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.

```
280 \def\LT@max@sel#1#2{%
281 {\ifdim#2=\wd\tw@
282 #1%
283 \else
284 \number\c@LT@chunks
285 \fi}%
286 {\the\wd\tw@}}
```

## 9.9 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.<sup>6</sup> Also this \hline is more like a \cline{1-\LT@cols}. tabular’s \hline would draw lines the full width of the page.

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

<sup>6</sup>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.

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

```

291 \def\LT@hline{%
292   \ifx\@let@token\hline
293     \global\let\@gtempa\@gobble
294     \gdef\LT@sep{\penalty-\@medpenalty\vskip\doublerulesep}%
295   \else
296     \global\let\@gtempa\@empty
297     \gdef\LT@sep{\penalty-\@lowpenalty\vskip-\arrayrulewidth}%
298   \fi
299   \ifnum0='{\fi}%
300   \multispan\LT@cols
301     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
302   \noalign{\LT@sep}%
303   \multispan\LT@cols
304     \unskip\leaders\hrule\@height\arrayrulewidth\hfill\cr
305   \noalign{\penalty\@M}%
306   \@gtempa}

```

### 9.10 Captions

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

```

307 \def\LT@caption{%
308   \noalign\bgroup
309     \@ifnextchar[{\egroup\LT@caption\@firstofone}\LT@caption}

```

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

```

310 \def\LT@caption#1[#2]#3{%
311   \LT@makecaption#1\fnum@table{#3}%
312   \def\@tempa{#2}%
313   \ifx\@tempa\@empty\else
314     {\let\\\space
315     \addcontentsline{lot}{table}{\protect\numberline{\thetable}{#2}}}%
316   \fi}

```

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

```

317 \def\LT@caption{%
318   \@ifstar
319     {\egroup\LT@caption\@gobble[]}%
320     {\egroup\@xdblarg{\LT@caption\@firstofone}}

```

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

```

321 \def\LT@makecaption#1#2#3{%
322   \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.

```

323   \sbox\@tempboxa{#1{#2: }#3}%
324   \ifdim\wd\@tempboxa>\hsiz
325     #1{#2: }#3%
326   \else
327     \hbox to\hsiz{\hfil\box\@tempboxa\hfil}%

```

```

328   \fi
329   \endgraf\vskip\baselineskip}%
330   \hss}}}
```

## 9.11 The Output Routine

The method used here for interfacing a special purpose output routine to the standard L<sup>A</sup>T<sub>E</sub>X routine is lifted straight out of F. Mittelbach's multicol package.

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

```

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

If this was a float or a marginpar we complain.

```

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

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

```

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

The last foot might not fit, so:<sup>7</sup>

```

338     \dimen@\pagegoal
339     \advance\dimen@-\ht\LT@lastfoot
340     \ifdim\dimen@<\ht\z@
341       \setbox\@cclv\vbox{\unvbox\z@\copy\LT@foot\vss}%
342       \@makecol
343       \@outputpage
344       \setbox\z@\vbox{\box\LT@head}%
```

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

```

345     \fi
```

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

```

346     \fi
```

Reset \@colroom.

```

347     \global\@colroom\@colht
348     \global\size\@colht
```

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

```

349     \vbox
350     {\unvbox\z@\box\ifvoid\LT@lastfoot\LT@foot\else\LT@lastfoot\fi}%
```

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

```

351   \fi
```

Else \outputpenalty > -\@Mi.

```

352   \else
```

If we have not reached the end of the table,

```

353     \setbox\@cclv\vbox{\unvbox\@cclv\copy\LT@foot\vss}%
354     \@makecol
355     \@outputpage
```

---

<sup>7</sup>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.

Reset \vsize.

356 \global\vsize\@colroom

Put the head at the top of the next page.

357 \copy\LT@head

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

358 \fi}

### 9.12 Commands for the 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.

359 \def\LT@end@hd@ft#1{%

360 \LT@echunk

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

361 \ifx\LT@start\endgraf

362 \LT@err

363 {Longtable head or foot not at start of table}%

364 {Increase LTchunksiz}%

365 \fi

366 \setbox#1\box\z@

367 \LT@get@widths

368 \LT@bchunk}

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

\endhead 369 \def\endfirsthead{\LT@end@hd@ft\LT@firsthead}

\endfoot 370 \def\endhead{\LT@end@hd@ft\LT@head}

\endlastfoot 371 \def\endfoot{\LT@end@hd@ft\LT@foot}

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

### 9.13 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

### 9.14 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.

373 \def\LT@startpbox#1{%

374 \bgroup

375 \let\@footnotetext\LT@p@ftntext

376 \hsize#1%

377 \@arrayparboxrestore

378 \vrule \@height \ht\@arstrutbox \@width \z@}



`\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.

```
379 \def\LT@endpbox{%
380   \@finalstrut\@arstrutbox
381   \egroup
382   \the\LT@p@ftn
383   \global\LT@p@ftn{ }%
384   \hfil}
```

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

```
385 \def\LT@p@ftntext#1{%
386   \edef\@tempa{\the\LT@p@ftn\noexpand\footnotetext[\the\c@footnote]}%
387   \global\LT@p@ftn\expandafter{\@tempa{#1}}}%
388 \end{package}
```