Abstract

This package defines the \texttt{longtable} environment, a multi-page version of \texttt{tabular}.

List of Tables

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An optional table caption (used in the list of tables)</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>A floating table</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>A difficult \texttt{\multicolumn} combination: pass 1</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>A difficult \texttt{\multicolumn} combination: pass 2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>A difficult \texttt{\multicolumn} combination: pass 3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>A difficult \texttt{\multicolumn} combination: pass 4</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>A summary of \texttt{longtable} commands</td>
<td>9</td>
</tr>
</tbody>
</table>

1 Introduction

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

The following example uses most of the features of the \texttt{longtable} environment. An edited listing of the input for this example appears in Section 8.

\textbf{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.

\footnote{This file has version number v4.19, last revised 2023-11-01.}

\footnote{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

<table>
<thead>
<tr>
<th><strong>FIRST</strong></th>
<th><strong>SECOND</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>longtable columns are specified in the same way as in the <code>tabular</code> environment.</td>
<td></td>
</tr>
<tr>
<td>`@{}r</td>
<td></td>
</tr>
<tr>
<td>The <code>\</code> command has an argument, just as in ``tabular` environment.</td>
<td></td>
</tr>
<tr>
<td>See the effect of <code>\[10pt]</code> Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Also <code>\hline</code> may be used, as in <code>tabular</code>.</td>
<td></td>
</tr>
<tr>
<td>That was a <code>\hline</code>.</td>
<td></td>
</tr>
<tr>
<td>This is a `\multicolumn{2}{</td>
<td></td>
</tr>
<tr>
<td>The <code>[t] [b] [c]</code> argument of <code>tabular</code> can not be used.</td>
<td></td>
</tr>
<tr>
<td>The optional argument may be one of <code>[l] [r] [c]</code> to specify whether the table should be adjusted or centrally.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>Lots of lines like this.</td>
<td></td>
</tr>
<tr>
<td>This goes at the bottom.</td>
<td></td>
</tr>
</tbody>
</table>
Table 1: (continued)

<table>
<thead>
<tr>
<th>First</th>
<th>Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>Lots of lines</td>
<td>like this.</td>
</tr>
<tr>
<td>These lines will appear in place of the usual foot of the table</td>
<td></td>
</tr>
<tr>
<td>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 the \hline commands.</td>
<td></td>
</tr>
</tbody>
</table>

2 Chunk Size

\texttt{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 \texttt{longtable} uses 20 rows per chunk, but this can be set by the user, with e.g., \texttt{\setcounter{LTchunksize}{10}}. These chunks do not affect page breaking, thus if you are using a \TeX with a lot of memory, you can set \texttt{LTchunksize} to be several pages of the table. \TeX will run faster with a large \texttt{LTchunksize}.

\footnote{This is a footnote.}
\footnote{\texttt{longtable} takes special precautions, so that footnotes may also be used in ‘p’ columns.}
\footnote{You can also use the plain \TeX syntax \texttt{\LTchunksize=10}.}
However, if necessary, `longtable` can work with \texttt{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 \texttt{LTchunksize} must be at least as large as the number of rows in each of the head or foot sections.

This document specifies \texttt{\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 .\texttt{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\footnote{Due to David Kastrup.}, 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 \texttt{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 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.

\texttt{\caption{...}} The `\caption{...}` command is essentially equivalent to `\multicolumn{n}{c}{\parbox{\LTcapwidth}{...}}` where \texttt{n} is the number of columns of the table. You may set the width of the caption with a command such as \texttt{\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 re-defined the standard \maketitle command to produce a different format for the captions, you may need to make similar changes to the longtable version, \LTmaketitle. 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 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 information that there is a wide 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.

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 \kill-ed 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
Table 3: A difficult \multicolumn combination: pass 1

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide multicolumn spanning 1–3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>multicolumn 1–2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>wide 1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4: A difficult \multicolumn combination: pass 2

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide multicolumn spanning 1–3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>multicolumn 1–2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>wide 1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5: A difficult \multicolumn combination: pass 3

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide multicolumn spanning 1–3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>multicolumn 1–2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>wide 1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6: A difficult \multicolumn combination: pass 4

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wide multicolumn spanning 1–3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>multicolumn 1–2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>wide 1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
used. If any \texttt{\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 \texttt{\multicolumn} entry, was not known when the final width for column 3 was fixed, due to the fact that both \texttt{\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 \texttt{longtable} will notice the error and reduce the column width accordingly. If this has to propagate to chunks before the \texttt{\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 \texttt{\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 \texttt{\setlongtables}, and the better choice of final column widths in the case of multiple \texttt{\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 \texttt{\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 \texttt{longtable} a \texttt{\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 \texttt{longtable} controls the horizontal alignment of the table. The possible options are \texttt{[c]}, \texttt{[r]} and \texttt{[l]}, for centring, right and left adjustment, respectively. Normally centring is the default, but this document specifies

\begin{verbatim}
\setlength\LTleft{parindent}
\setlength\LTright{\fill}
\end{verbatim}

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 \texttt{\extracolsep} command. For instance

\begin{verbatim}
\begin{tabular*}{\textwidth}{@{\extracolsep{...}}...}
\end{verbatim}

produces a full width table, to get a similar effect with \texttt{longtable} specify

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

.................longtable.sty..........................
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 \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.
7 Summary

Table 7: A summary of \longtable commands

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LTleft \LTright \LTPre \LTpost \LTcapwidth \LTchunksize</td>
<td>Glue to the left/right of the table/after/before the table/width/cap width of caption/number of rows per chunk.</td>
</tr>
</tbody>
</table>

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

- \\[⟨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[⟨lot⟩]{⟨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.
8 Verbatim highlights from Table 1

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

9.1 Initial code

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

\LT@err \LT@warn \LT@final@warn

9.2 Options

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

erro show pausing

set final

9.3 User Settable Parameters

\LTleft \LTright \LTpre \LTpost \LTchunksize

\LTleft \LTright \LTpre \LTpost \LTchunksize

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

\LTcapwidth Width of the $\texttt{parbox}$ 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
\LT@foot
\LT@lastfoot

\LT@gbox

\LT@cols Counter for number of columns.
28 \newcount\LT@cols

\LT@rows Counter for rows up to chunksize.
29 \newcount\LT@rows

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

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

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

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

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

Not needed for new letter class.
\ifx\startlabels\undefined

..............................longtable.sty..............................
\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.
\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

9.5 The \texttt{longtable} environment
\longtable Called by \texttt{\begin{longtable}}. This implementation does not work in multiple column formats. \texttt{\par} added at V3.04.
\def\longtable{%
  \par
  \if@noskipsec\mbox{\par}\fi
  \nobreakfalse
  \if\multicols@undefined
    \else
      \ifnum\col@number>1
        \twocolumntrue
      \fi
  \fi
  \if@twocolumn
    \LT@err{longtable not in 1-column mode}\@ehc
  \fi
  \begingroup

Check for an optional argument.
  \@ifnextchar\[
    \LT@array{\LT@array[x]}
\LT@array Start setting the alignment. Based on \texttt{\array} from the \LaTeX kernel and the \texttt{array} package.
  Since Version 3.02, \texttt{longtable} has used the internal counter \texttt{\c@LT@tables}. The \LaTeX counter \texttt{table} is still incremented so that \texttt{\caption} works correctly.
\def\LT@array[#1]{%\par
  \refstepcounter{table}\stepcounter{LT@tables}%
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 \texttt{longtable}.
\global\let\LT@mcw@rn\relax
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@mcol\multicolumn
\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}}%
\def\pagebreak{\noalign{\ifnum'}=0\fi\@testopt{\LT@no@pgbk-}4}\pagebreak
\def\nopagebreak{\noalign{\ifnum'}=0\fi\@testopt\LT@no@pgbk4}\nopagebreak
\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\else
\let\@acol\@tabacol \let\@classz\@tabclassz \let\@classiv\@tabclassiv
\def\@startpbox{\vtop\LT@startpbox}\@@startpbox\@@endpbox
\let\LT@LL@FM@cr\@tabularcr
\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}
```

----------------------------------------longtable.sty----------------------------------------
We need to rename `\@preamble` here as F.M.’s scheme uses `\global`, and we may need to nest `\@mkpream`, e.g., for `\multicolumn` or an array. We do not need to worry about nested `longtable`s though!

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

```latex
\LT@setprevdepth
```

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

```latex
\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.

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

A few more internal commands for `longtable`.

```latex
\m@th\let\par\@empty
\everycr{}\lineskip\z@\baselineskip\z@
```

Start the first chunk.

```latex
\LT@bchunk}
```

\LT@no@pgbk

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

```latex
\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.

```latex
\def\LT@start{%
\let\LT@start\endgraf
\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 \texttt{\pagetotal} will be for the page where the table will actually end up. To achieve this, we need to give \LaTeX{} an awfully bad, but legal, breakpoint. The simplest way of doing this seems to be to insert a \texttt{\kern} that counters the \texttt{\pageshrink} for the page, followed by a \texttt{\penalty} and a \texttt{\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 \texttt{9999} (10000 is infinite and thus not a legal breakpoint) and cancel out the \texttt{\kern} with a new \texttt{\kern}.

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

\begin{verbatim}
\ifdim \pagetotal<\pagegoal \else
  \dimen@=\pageshrink
  \advance \dimen@ 1sp %
  \kern\dimen@\penalty 9999\endgraf \kern-\dimen@
\fi
\end{verbatim}

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

\begin{verbatim}
\dimen@=\pagetotal
\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
\edef\LT@reset@vfuzz{\vfuzz\the\vfuzz\vbadness\the\vbadness\relax}%
\vfuzz\maxdimen
\vbadness\@M
\setbox\z@\copy\z@
\setbox\z@\vsplit\z@ to \ht\@arstrutbox
\setbox\z@\vbox{\unvbox\z@}
\LT@reset@vfuzz
\advance\dimen@ \ht
\ifdim\ht\@arstrutbox>\ht\z@\@arstrutbox\else\ht\z@\fi
\advance\dimen@ \dp
\ifdim\dp\@arstrutbox>\dp\z@\@arstrutbox\else\dp\z@\fi
\ifdim\pagegoal<\z@\pagegoal\z@\fi
\vfil\break
\else
\fi
\end{verbatim}

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.

\begin{verbatim}
\ifdim\pageshrink>\z@\pageshrink\z@\fi
\fi
\end{verbatim}

Store height of page minus table foot in \texttt{@colroom}.

\begin{verbatim}
\global{@colroom}=@colht
\end{verbatim}
If the foot is non empty, reduce the $\vsize$ and $\@colroom$ accordingly.
\begin{longtable}
% 
\ifvoid\LT@foot\else
\advance\vsize-\ht\LT@foot
\global\advance\vsize-\ht\LT@foot
\global\advance\@colroom-\ht\LT@foot
\global\advance\maxdepth-\ht\LT@foot
\fi

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
\output{\LT@output}
\endlongtable
\end{longtable}
\def\endlongtable{\def\end{longtable}}

\if@filesw
\immediate\write\@auxout{
\gdef\expandafter
\csname LT@\romannumeral\c@LT@tables\endcsname{\LT@save@row}}
\fi

\LT@mcw@rn
\ifx\LT@save@row\LT@@save@row
\else
\LT@warn{Column $\@width$ s have changed\MessageBreak in table \thetable}
\fi

Write the dummy row to the .aux file. Since V3.06, use .lta for letter.sty.
\if@filesw
\immediate\write\@auxout{
\let\LT@entry\LT@entry@write\immediate\write\@auxout{
\let\LT@entry\LT@entry@chop
\let\LT@entry\LT@entry@chop
\def\LT@entry@chop\lt@save@row}}
\fi

At this point used to issue a warning if a $\multicolumn$ has been set in draft mode.
\LT@mcw@rn
\ifx\LT@save@row\LT@@save@row
\else
\LT@warn{Column $\@width$ s have changed\MessageBreak in table \thetable}
\fi

Since Version 3.02, longtable has used the internal counter $\c@LT@tables$ rather than the \LTX 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.
\if@filesw
\immediate\write\@auxout{
\gdef\expandafter\noexpand\csname LT@\romannumeral\c@LT@tables\endcsname{\LT@save@row}}
\fi

If the last chunk has different widths than the first, warn the user. Also trigger a warning to rerun \LTX at the end of the document.
\ifx\LT@save@row\LT@@save@row
\else
\LT@warn{Column $\@width$ s have changed\MessageBreak in table \thetable}
\fi
Force one more go with the `longtable` output routine.

```
172 \LT@final@warn
173 \fi
```

Now close the group to return to the standard routine.

```
178 \ifvoid\LT@foot\else
179 \global\advance\vsize\ht\LT@foot
180 \global\advance\@colroom\ht\LT@foot
181 \dimen@\pagegoal\advance\dimen@\ht\LT@foot\pagegoal\dimen@\fi
183 \endgraf\penalty\z@\addvspace\LTpost
```

Footnotes. As done in the `multicol` package.

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

### 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 TeX'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 TeX definition) defines `\@tabclassz` so that `\@sharp` is inside a group.)

```
\LT@nofcols Find the next `&`, then look ahead to see what is next.
185 \def\LT@nofcols#1\\% 186 \futurelet\@let@token\LT@n@fcols
```

```
\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.
187 \def\LT@nofcols\% 188 \advance\LT@cols\@ne 189 \ifx\@let@token\LT@nofcols 190 \expandafter\@gobble 191 \else 192 \expandafter\LT@nofcols 193 \fi
```

---

\footnote{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?}
9.7 The `\` and `\kill` Commands

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{
def\crcr{\LT@crcr \noalign{\nobreak}}\let\cr\crcr
\LT@t@bularcr}%
\LT@t@bularcr}
\LT@cr
\let\LT@cr\crcr
\LT@setprevdepth This will be redefined to set the \prevdepth at the start of a chunk.
\let\LT@setprevdepth\relax
\LT@t@bularcr
\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
\gdef\LT@setprevdepth{%
\prevdepth\z@
\global\let\LT@setprevdepth\relax}%
\expandafter\LT@xtabularcr
\else
\ifnum0='{}\fi
\expandafter\LT@LL@FM@cr
\fi}
\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
```
The version with an optional argument. \ifnum0='{\fi} added in version 3.01. Changed in 3.14.
\begin{verbatim}
\def\LT@argtabularcr[#1]{% 
\ifnum0='{}\fi
\ifdim #1>\z@ \unskip\@xargarraycr{#1}%%
\else \@yargarraycr{#1}%%
\fi
Add the dummy row, and finish the \halign.
\end{verbatim}

This ends the current chunk, and removes the dummy row.
\begin{verbatim}
\def\LT@echunk{% 
\crcr\LT@save@row\cr\egroup
\global\setbox\LT@gbox\lastbox
\end{verbatim}

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.

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 &.
\begin{verbatim}
\def\LT@entry#1#2{% 
\ifhmode\@firstofone{&}\fi\omit
\ifnum#1=\c@LT@chunks\else\kern#2\relax\fi
\end{verbatim}
This definition for the argument of \LTsaverow 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.

\LTentrychop
\LTentrywrite
\LTkill
\LTrebox

9.8 The Dummy Row

The dummy row is kept inside of the macro \LTsaverow.

\LTblankrow
\LTbuildblank

Whoops! What's that supposed to be? A drop-in replacement for the first task of Appendix D in the LaTeXbook. The \romannumeral produces \LTcols instances of \Roman followed by i. The below macro then replaces the ms by appropriate entries.
\LT@make@row \par
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
\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%
\fi
\setlongtables \par
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 \LT@table counter. 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.

\LT@get@widths \par
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 \TeX{}book.

\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 \def\LT@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:
\def\LT@max@sel#1#2{%\ifdim#2=\wd\tw@#1\else\number\c@LT@chunks\fi}%\the\wd\tw@}
\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.
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.\footnote{\textit{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.} Also this \hline is more like a \cline\{1-\LT@cols\}.\tabular's \hline would draw lines the full width of the page.

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

9.10 Captions

\LT@caption The caption is \multicolumn{\LT@cols}{c}{\{a parbox with the table's caption\}}\protect\fnum@table added at V3.05.

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

\LT@c@ption Caption command (no [optional argument])
\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.

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

\LT@output

Actually this is not so bad, with FM leading the way.

\LT@output

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.

\endgraf

\vskip\baselineskip

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

\fi

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

\fi

\endgraf

\vskip\baselineskip

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

\fi

\endgraf

\vskip\baselineskip

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
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}%%
\@makecol
\@outputpage
Reset \vsize.
\global\vsize\@colroom
Put the head at the top of the next page.
\copy\LT@head\nobreak
End of \ifnum\outputpenalty < -\@Mi.
\fi}

9.12 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.
\% \def\LT@end@hd@ft#1{%
\LT@echunk
\ifx\LT@start\endgraf
\LT@err
{Longtable head or foot not at start of table}%%
{Increase LTchunksize}%%
\fi
\setbox#1\box\z@
\LT@get@widths
\LT@bchunk

\endfirsthead Call \LT@end@hd@ft with the appropriate box.
\% \def\endfirsthead{\LT@end@hd@ft\LT@firsthead}
\endfoot \def\endhead{\LT@end@hd@ft\LT@head}
\endlastfoot \def\endfoot{\LT@end@hd@ft\LT@foot}
\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.
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.
\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]{#1}%  
\global\LT@p@ftn\expandafter\{\@tempa\}  
%(package)