The xtab package[*]

Peter Wilson
herries dot press (at) earthlink dot net

2008/07/26

Abstract

The xtab package enables long tables to be automatically broken at page boundaries. It is an extension of the supertabular package and also reduces or eliminates some of its weaknesses.

Contents

List of Tables

1 Introduction

Although the xtab package was originally developed as part of a suite for typesetting ISO international standards [Wil96], it is also applicable for use with the LaTeX standard classes. The package is an extension of the supertabular package developed by Johannes Braams and Theo Jurriens.[1] It reduces some of the weaknesses noted in the supertabular documentation and provides additional functionality.

Section 2 provides the user manual for the package which enables long tables to be automatically broken across multiple pages. Section 3 describes the implementation.

[*]This file (`xtab.dtx`) has version number v2.3c, last revised 2008/07/26.
[1]supertabular.sty, version 4.1c, 7 November 1997.

This manual is typeset according to the conventions of the LaTeX DOCSTRIP utility which enables the automatic extraction of the LaTeX macro source files [GMS94].

2 The xtab package

The supertabular package provides for the automatic breaking of a long table across page boundaries. The extension provided here enables the heading on the table on the last page to differ from those on earlier pages of the table. The downside of the extension is that LaTeX has to be run twice if the document contains a `supertabular`. However, LaTeX is usually run at least twice for any but the simplest document in order to get cross-references and Table of Contents, etc., resolved correctly.

The current version of the extension also either cures or reduces following weaknesses in the supertabular package.[2]

1. Sometimes the top caption of a `supertabular` is printed on one page and the body is printed on the following page(s). That is, there is a lonely caption.
2. Sometimes the last page of a `supertabular` consists of an empty table. That is, just the head and foot of the table are printed.
3. If the number of lines in the first header for the table differs from the number of lines in subsequent headers, then the continuation pages of the table may be too short or, more troubling, too long.

The weaknesses are caused by trying to guess where TeX will put a page break. The package has to guesstimate how long the next entry will be in the table and, if it is too long for the available space, it puts in its own page break. If its guess is off too much in one direction, TeX will break the page unexpectadly; if it's off in the other direction supertabular will put in an unnecessary page break.

The xtab package has reduced, but perhaps not entirely eliminated, these weaknesses. Some hand tuning may still be required.

The principal commands available are given in Table 1.

Table 1: The principal xtab package commands

Command	**Effect**
`\begin{xtabular}{...}`	This is equivalent to the normal `\begin{tabular}{...}` environment. You supply the specification of the columns just as for the normal tabular environment.
	Continued on next page

[2]I have corresponded with the authors of supertabular about these.

Table 1 – continued from previous page

Command	**Effect**
	All commands that can be used within a `tabular` environment can also be used within the `xtabular` environment. Unlike the `tabular` environment which prevents page breaking within the tabular, the `xtabular` allows page breaking, so that tabulars can extend automatically across several pages. `xtabular` starts off with a `tabular` environment and checks the amount of space left on the page as it adds each row to the tabulation. If the space left on the page is too short for another row, then it ends the current `tabular`, performs a page break and starts another `tabular` on the following page. This process is repeated until all the rows have been output. There are special commands for captioning an `xtabular` as a table, and also elements can be automatically inserted after each (internal) `\begin{tabular}` and immediately before each `\end{tabular}`. Do not put a `xtabular` in a `table` environment, as the `table` environment keeps its contents on a single page (presumably you are using `xtabular` because its contents are longer than one page).
`\end{xtabular}`	End the `xtabular` environment.
`\begin{mpxtabular}`	Like the `xtabular` environment except that each 'page' is put into a `minipage` first. Thus it is possible to have footnotes inside an `mpxtabular`. The footnote text is printed at the end of each page.
	Continued on next page

Table 1 – continued from previous page

Command	**Effect**
`\end{mpxtabular}`	End the `mpxtabular` environment.
	Note: If any of the following commands are used, then they should be placed before the particular `xtabular` environment that they apply to.
`\topcaption{...}`	A command to provide a caption for the table. The caption is placed at the top of the table.
`\bottomcaption{...}`	A command to provide a caption for the table. The caption is placed at the bottom of the table.
`\tablecaption{...}`	A command to provide a caption for the table. The caption is placed at the default position, which is at the top of the table. **Notes:** You cannot use the `\caption` command but you can put a label after any of these captioning commands. If you want captioning, the command must be specified before the start of the supertabular environment. The `\...caption{}` command(s) remain in effect until changed by another `\...caption` command.
`\tablefirsthead{...}`	Defines the contents of the first occurence of the tabular head. The tabular head is some special treatment of the first row in the table. This command is optional. If used, the header must be closed by the end of line command for tabulars (e.g., `\\`).
`\tablehead{...}`	Defines the contents of the table head on subsequent pages. For example, you might want to note that this is a continuation of the table on the previous page, as well as repeating any column headings that were given at the start of the `xtabular` by `\tablefirsthead`.

Continued on next page

Table 1 – concluded from previous page

Command	**Effect**
	The header must be closed like the `\tablefirsthead` command.
`\tablelasthead{...}`	Defines the contents of the table head on the last page of the table. For example, you might want to note that the table is concluded on this page. The header must be closed like the `\tablefirsthead` command.
`\notablelasthead`	Switches off the last `\tablelasthead`. A `\tablelasthead` stays in effect until overwritten by a new `\tablelasthead` or cancelled by this command.
`\tabletail{...}`	The contents of this command are inserted before the (internal) `\end{tabular}` on each page except for the last page of the table. For example, you might want to note that the table is continued on the next page.
`\tablelasttail{...}`	The contents of this command are inserted before the final (internal) `\end{tabular}` of the table. For example, you might want to note that this is where the table ends.

As well as the `xtabular` and `mpxtabular` environments there are the corresponding starred versions (i.e., `xtabular*` and `mpxtabular*` for use in two column mode where the table is meant to span both columns.

Table 1 was produced by code similar to the following:

```
\topcaption{The principal xtab package commands} \label{tab:xtab}
\tablefirsthead{\hline \multicolumn{1}{|c|}{\textbf{Command}} &
                      \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tablehead{\multicolumn{2}{c}%
          {{\captionsize\bfseries \tablename\ \thetable{} --
            continued from previous page}} \\
  \hline   \multicolumn{1}{|c|}{\textbf{Command}} &
          \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tablelasthead{\multicolumn{2}{c}%
          {{\captionsize\bfseries \tablename\ \thetable{} --
            concluded from previous page}} \\
  \hline   \multicolumn{1}{|c|}{\textbf{Command}} &
          \multicolumn{1}{c|}{\textbf{Effect}} \\ \hline }
\tabletail{\hline \multicolumn{2}{|r|}{{Continued on next page}} \\ \hline}
\tablelasttail{\hline \hline}

\begin{center}
```

```
\begin{xtabular}{|l|p{0.5\textwidth}|}
\verb|\begin{xtabular}{...}| & This is equivalent to the normal
                         \verb|\begin{tabular}{...}| environment.
                         You supply the specification of the columns
                         just as for the normal \Lenv{tabular} environment.
 \\
 &
                         All commands that can be used within a \Lenv{tabular}
                         environment can also be used within
                         the \Lenv{xtabular} environment.
 \\
 &
    Unlike the \Lenv{tabular} environment which prevents page breaking
within the tabular, the \Lenv{xtabular} allows page breaking, so that
tabulars can extend automatically across several pages.
... ... ...
\verb|\tablelasttail{...}| & The contents of this command are inserted before
                         the final (internal) \verb|\end{tabular}| of the table.
 \\
 &
    For example, you might want to note that  this is where
the table ends.
 \\
\end{xtabular}
\end{center}
```

The table is only broken between rows — a row will not be split across pages. This can lead to some bad page breaks, especially if there are rows with a large vertical height (like some in Table 1). It is best to keep rows not too tall.

Unkike the `table` environment which floats, an `xtabular` environment is typeset at the point in the document where the environment is specified. It is best not to start an `xtabular` too close to the bottom of a page otherwise there might be an ugly page break.

`\shrinkheight` The command `\shrinkheight{`⟨*length*⟩`}` may be used after the first `\\` in the table to modify the allowed height of the table on that page. A positive ⟨*length*⟩ decreases the allowed space on the page and a negative ⟨*length*⟩ increases the allowed space.

For example:

`\shrinkheight{2\baselineskip}` decreases the space per page by two lines.

`\shrinkheight{-\baselineskip}` increases the space per page by one line.

Note that I have never tried using this command so I cannot comment on its efficacy. Instead, I use the `\xentrystretch` command when necessary.

`\xentrystretch` The command `\xentrystretch{`⟨*decimal-fraction*⟩`}` can be used before a table to modify the amount of vertical space apparently consumed by each entry in the subsequent table(s). The default is `\xentrystretch{0.1}` which specifies a 10% overestimate in the vertical space. Similarly, `\xentrystretch{0.25}` will overestimate the space by 25%. A different value may be used for each table in order to eliminate, or at least reduce, bad page breaks. Increasing the value causes fewer entries to be put on a page, thus reducing the chance of TEX putting in a page break before the xtab package is prepared for one.

You may specify the font used for the `\tablehead` and `\tablelasthead` yourself.

Note: Within ISO documents, captions shall be in bold font. The `iso` class also provides a command for setting the size of the font used in captions, namely `\captionsize`. The default value for this is set by the `iso` class. For the curious, the default definition is:

```
\newcommand{\captionsize}{\normalsize}
```

2.1 Options

The xtab package has three options which control the amount of information that is written to the `.log` file. The options are:

1. The option errorshow (the default) does not write any extra information;
2. The option pageshow writes information about when and why xtab decides to produce a new page;
3. The option debugshow, which also includes pageshow, additionally writes information about each line that is added to the table.

Under normal circumstances xtab is used without invoking any option. The pageshow option may be useful when attempting to cure a bad page break. The debugshow option, as its name implies, is principally of use to the xtab developer.

Independently of the options, the command `\sstraceon` may be used at any point in the document to turn on printing of debugshow data. This can be turned off later by the `\sstraceoff` command, which will stop all ...show printing.

3 The implementation

The xtab package provides an extension to the supertabular package written by Johannes Braams and Theo Jurriens.[3] The major portion of the following documentation is taken from `supertabular.dtx`. The package is designed to be used with the `iso` class in addition to the usual `article`, etc., classes.

The extension provided here enables the heading on the table on the last page to differ from those on earlier pages of the table. The implementation of the extension is based on ideas in David Carlisle's longtable package. The downside of the extension is that LATEX has to be run twice if the document contains a `supertabular`. However, LATEX is usually run at least twice for any but the simplest document in order to get cross-references and Table of Contents, etc., resolved correctly.

[3]`supertabular.sty`, version 4.1c, 7 November 1997.

The current version of the extension also either cures or reduces following weaknesses in the supertabular package.[4]

1. Sometimes the top caption of a `supertabular` is printed on one page and the body is printed on the following page(s). That is, there is a lonely caption.
2. Sometimes the last page of a `supertabular` consists of an empty table. That is, just the head and foot of the table are printed.
3. If the number of lines in the first header for the table differs from the number of lines in subsequent headers, then the continuation pages of the table may be too short or, more troubling, too long.

The first version of `xtab` imported much of the code from the supertabular package (version 3.7) but I found that this did not work well because there were incompatible coded versions of supertabular available on CTAN. Further, I found that there were some problems with the original `supertabular` code in any case.[5] I have to make the assumption that other users may have dissimilar or problematic versions, so include all the code here, and thus any errors can now be laid at my door.

The requirement for compatibilty with the `iso` class is achieved by modifications to the `\ST@caption` command only. Effectively this is orthogonal to the code required to implement the extension.

Now for the code itself. As syntactic sugar, all new macros for the extension have the prefix 'PWST' to distinguish them from the original macros. I have also denoted all extensions to the original supertabular by introducing them as *Extension:*.

Announce the name and version of the package, which requires LaTeX 2_ε.

```
⟨*xtab⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{xtab}[2008/07/26 v2.3c Extended supertabular package]

```

`\c@tracingst` There are three options with the package which control the amount of information written to the log file:

1. errorshow (the default) no extra information
2. pageshow writes information about page breaking
3. debugshow adds information about each line that is added to the tabular

```
\newcount\c@tracingst
\DeclareOption{errorshow}{\c@tracingst\z@}
```

Extension: The next line in the original code did not do what the authors intended; the number should have been 3 rather than 2.

[4]Only the first two of these have been recognised by the authors of supertabular.

[5]I also found a bug in the 4.1b version which the authors kindly fixed in version 4.1c.

```
%%%%\DeclareOption{pageshow}{\c@tracingst\tw@}
\DeclareOption{pageshow}{\c@tracingst\thr@@}
\DeclareOption{debugshow}{\c@tracingst5\relax}
\ProcessOptions

```

`\if@topcaption` `\topcaption` `\bottomcaption`

The user-commands `\topcaption` and `\bottomcaption` set the flag `@topcaption` to determine where to put the tablecaption. The default is to put the caption on the top of the table

```
\newif\if@topcaption \@topcaptiontrue
\def\topcaption{\@topcaptiontrue\tablecaption}
\def\bottomcaption{\@topcaptionfalse\tablecaption}

```

`\PWST@thecaption` `\PWSTcapht`

Extension: `\PWST@thecaption` is used to store the text of the table's caption. The vertical space required by a caption is stored in `\PWSTcapht`.

```
\gdef\PWST@thecaption{}
\newdimen\PWSTcapht
```

`\tablecaption` `\@xtablecaption`

This command has to function exactly like `\caption` does except it has to store its argument (and the optional argument) for later processing *within* the supertabular environment.

```
\long\def\tablecaption{%
  \refstepcounter{table}\@dblarg{\@xtablecaption}}
\long\def\@xtablecaption[#1]#2{%
```

Extension: I store the caption text for later measurement.

```
  \long\gdef\PWST@thecaption{#2}%
```

Finish up with the original code.

```
  \long\gdef\@process@tablecaption{\ST@caption{table}[#1]{#2}}}
\global\let\@process@tablecaption\relax

```

`\ifST@star`

This switch is used in the internal macros to remember which kind of environment was started.

```
\newif\ifST@star
```

`\ifST@mp`

This flag is used in the internal macros to remember if the tabular is to be put in a minipage.

```
\newif\ifST@mp
```

`\ST@wd`

For the `supertabular*` environment it is necessary to store the intended width of the tabular.

```
\newdimen\ST@wd
```

`\ST@rightskip` `\ST@leftskip` `\ST@parfillskip`

For the `mpsupertabular` environments we need special versions of `\leftskip`, `\rightskip` and `\parfillskip`.

```
\newskip\ST@rightskip
\newskip\ST@leftskip
\newskip\ST@parfillskip

```

\@initisotab Required for ISO class, and check if class loaded.

```
\@ifundefined{@initisotab}{%
  \newcommand{\@initisotab}{}%
  \newif\ifinfloat}{\typeout{xtab using iso captions}}

```

\ST@caption This is a redefinition of LaTeX's `\@caption`, `\@makecaption` is called within a group so as not to return to `\normalsize` globally. In the original a fix was made for the 'feature' of the `\@makecaption` of `article.sty` and friends that a caption *always* gets a `\vskip 10pt` at the top and *none* at the bottom; if a user wants to precede his table with a caption this results in a collision. This fix is not implemented here as I think it should be done by the user modifying `\beforecaptionskip` and `\aftercaptionskip`.

Extension: The ISO captioning is also initialised.

```
\long\def\ST@caption#1[#2]#3{\par%
  \@initisotab
  \addcontentsline{\csname ext@#1\endcsname}{#1}%
                  {\protect\numberline{%
                    \csname the#1\endcsname}{\ignorespaces #2}}%
  \begingroup
    \@parboxrestore
    \normalsize
  %%  \if@topcaption \vskip -10\p@ \fi
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
  %%  \if@topcaption \vskip 10\p@ \fi
  \endgroup
```

Extension: The height of the caption is subtracted from the available space on the page.

```
  \global\advance\ST@pageleft -\PWSTcapht
  \ST@trace\tw@{Added caption. Space left for xtabular: \the\ST@pageleft}}

```

\tablehead
\tablefirsthead `\tablehead` activates the new tabular `\cr` commands.

```
\newcommand\tablehead[1]{%
  \gdef\@tablehead{%
  \noalign{%
    \global\let\@savcr=\\
    \global\let\\=\org@tabularcr}%
  #1%
  \noalign{\global\let\\=\@savcr}}}
\tablehead{}
\newcommand\tablefirsthead[1]{\gdef\@table@first@head{#1}}

```

`\c@PWSTtable` `\PWSTlastpage` `\PWSTcurpage` `\PWSTpenultimate` `\PWSTtempc` `\PWSTlines` `\PWSThead` `\PWSTlasthead`

Extension: These are counters for the supertabular extension. *c@PWSTtable* counts the number of supertabulars in case one or more are not captioned. *PW-STlastpage* is a counter holding the number of pages that a supertabular uses and *PWSTpenultimate* is the penultimate page. *PWSTcurpage* counts the current number of supertabular pages processed. *PWSTtempc* is a scratch counter for page processing.

```
\newcounter{PWSTtable}
\newcount\PWSTlastpage
\newcount\PWSTpenultimate
\newcount\PWSTcurpage
\newcount\PWSTtempc
```

Extension: *PWSTlines* is used to count the number of supertabular entry lines on a page. Estimates of the number of lines in the normal table heading is held by *PWSThead*, and similarly *PWSTlasthead* is for the number of lines in the last heading.

```
\newcount\PWSTlines
%%% \newcount\PWSThead
%%% \newcount\PWSTlasthead
```

`\iffirstcall`

Extension: This is used by the extension code to flag if the presumed last page overflows. If overflow occurs, then `firstcall` is set to false.

```
\newif\iffirstcall
```

`\PWST@lastht` `\PWST@generalht` `\PWST@ht`

Extension: The estimated height of a table header and tail (i.e., the height of an empty table) for the last page of a supertabular is stored in `\PWSTlastht`. Similarly, the corresponding height of an empty table on a general page (neither the first nor the last) is stored in `\PWSTgeneralht`. `\PWST@ht` is a scratch variable.

```
\newdimen\PWST@lastht
\newdimen\PWST@generalht
\newdimen\PWST@ht

```

`\tablelasthead` `\@table@last@head` `\notablelasthead`

Extension: `\tablelasthead` is the extension user command to specify the heading for the last page of a supertabular. The command `\notablelasthead` switches off the last heading. This has to be used if a last headed table precedes one that does not have a special last head.

```
\newcommand{\tablelasthead}[1]{\gdef\@table@last@head{#1}}
\newcommand{\notablelasthead}{\let\@table@last@head\relax}
```

Now initialize these commands.

```
\tablelasthead{}
\notablelasthead
```

`\tabletail` `\tablelasttail`

`\tabletail` is the user command to specify the appearance of the bottom of each tabular on a page. Special treatment is given to the end of the supertabular via the `\tablelasttail` command.

If the user uses an extra amount of tabular-data (like `\multicolumn`) in `\tabletail` TeX starts looping because of the definition of `\nextline`. So make

\\ act like just a \cr inside this tail to prevent the loop. Save and restore the value of \\

```
\newcommand\tabletail[1]{%
  \gdef\@tabletail{%
    \noalign{%
      \global\let\@savcr=\\
      \global\let\\=\org@tabularcr}%
    #1%
    \noalign{\global\let\\=\@savcr}}}
\tabletail{}
\newcommand\tablelasttail[1]{\gdef\@table@last@tail{#1}}
\tablelasttail{}

```

\sttraceon \sttraceoff The original supertabular included a tracing mechanism to follow the decisions supertabular made about page breaking. This is now also used as a debugging mechanism for the extension.

```
\newcommand\sttraceon{\c@tracingst5\relax}
\newcommand\sttraceoff{\c@tracingst\z@}
```

\ST@trace A macro that gets the trace message as its argument

```
\newcommand\ST@trace[2]{%
  \ifnum\c@tracingst>#1\relax
    \GenericWarning{(xtab)\@spaces\@spaces}{Package xtab: #2}%
  \fi}

```

\ST@pageleft This register holds the estimate of the amount of space left over on the current page. This is used in the decision when to start a new page.

```
\newdimen\ST@pageleft
```

\shrinkheight \setSTheight `\shrinkheight` is a command to diminish the value of `\ST@pageleft` if necessary.
`\setSTheight` sets the value of `\ST@pageleft` if necessary.

```
\newcommand*\shrinkheight[1]{%
  \noalign{\global\advance\ST@pageleft-#1\relax}}
\newcommand*\setSTheight[1]{%
  \noalign{\global\ST@pageleft=#1\relax}}
```

\xentrystretch \PWST@xentrystretch *Extension:* Provide a user and internal command for fudging the estimated space taken by a table entry. Initialise to 10% increase.

```
\newcommand{\xentrystretch}[1]{\def\PWST@xentrystretch{#1}}
\xentrystretch{0.1}

```

\ST@headht \ST@tailht The register `\ST@headht` holds the height of the first head of a `supertabular`. The register `\ST@tailht` holds the height of the tail.

```
\newdimen\ST@headht
\newdimen\ST@tailht
```

`\ST@pagesofar` Register `\ST@pagesofar` stores the estimate of the amount of the page already filled up.

```
\newdimen\ST@pagesofar
```

`\ST@pboxht` The measured (total) height of a parbox argument.

```
\newdimen\ST@pboxht
```

`\ST@lineht` `\ST@stretchht` `\ST@prevht` The estimated height of a normal line is stored in `\ST@lineht`. The register `\ST@stretchht` is used to store the difference between the normal line height and the line height when `\arraystretch` has a non-standard value. This is used in the case when p-box entries are added to the tabular. `\ST@prevht` stores the height of the previous line to use it as an estimate for the height of the next line. This is needed for a better estimate of when to break the tabular.

```
\newdimen\ST@lineht
\newdimen\ST@stretchht
\newdimen\ST@prevht
```

`\ST@toadd` When a tabular row is ended with `\\[...]` we need to temporarily store the optional argument in `\ST@toadd`.

```
\newdimen\ST@toadd
```

`\ST@dimen` A private scratch dimension register.

```
\newdimen\ST@dimen
```

`\ST@pbox` A box register to store the contents of a parbox.

```
\newbox\ST@pbox

```

`\ST@tabularcr` `\ST@xtabularcr` `\ST@argtabularcr` These are redefinitions of `\@tabularcr` and `\@xtabularcr`. This is needed to include `\ST@cr` in the definition of `\@xtabularcr`.

All redefined macros have names that are similar to the original names, except with a leading 'ST'.

```
\def\ST@tabularcr{%
  {\ifnum0=`}\fi
  \@ifstar{\ST@xtabularcr}{\ST@xtabularcr}}
\def\ST@xtabularcr{%
  \@ifnextchar[%]
    {\ST@argtabularcr}%
    {\ifnum0=`{\fi}\cr\ST@cr}}
\def\ST@argtabularcr[#1]{%
  \ifnum0=`{\fi}%
  \ifdim #1>\z@
    \unskip\ST@xargarraycr{#1}%
  \else
    \ST@yargarraycr{#1}%
  \fi}
```

`\ST@xargarraycr`
`\ST@yargarraycr`

In this case we need to copy the value of the optional argument of `\\` in our private register `\ST@toadd`.

```
\def\ST@xargarraycr#1{%
  \@tempdima #1\advance\@tempdima \dp \@arstrutbox
  \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr
  \noalign{\global\ST@toadd=#1}\ST@cr}
```

Here we need to insert `\ST@cr`

```
\def\ST@yargarraycr#1{%
  \cr\noalign{\vskip #1\global\ST@toadd=#1}\ST@cr}

```

`\ST@startpbox`

The macros that deal with parbox columns need to be redefined, because we need to know the size of the parbox.

```
\def\ST@startpbox#1{%
```

To achieve our goal we need to save the text in box.

```
%%%%  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
  \setbox\ST@pbox\vtop\bgroup\setlength\hsize{#1}\@arrayparboxrestore}
```

`\ST@astartpbox`

supertabular version of `\@astartpbox`.

```
\def\ST@astartpbox#1{%
  \bgroup\hsize#1%
%%%%  \setbox\ST@pbox\vtop\bgroup\hsize#1\@arrayparboxrestore}
  \setbox\ST@pbox\vtop\bgroup\setlength\hsize{#1}\@arrayparboxrestore}
```

`\ST@endpbox`
`\ST@aendpbox`

supertabular versions of `\@endpbox` and `\@aendpbox`.

```
\def\ST@endpbox{%
  \@finalstrut\@arstrutbox\par\egroup
  \ST@dimen=\ht\ST@pbox
  \advance\ST@dimen by \dp\ST@pbox
  \ifnum\ST@pboxht<\ST@dimen
    \global\ST@pboxht=\ST@dimen
  \fi
  \ST@dimen=\z@
  \box\ST@pbox\hfil}
\def\ST@aendpbox{%
  \@finalstrut\@arstrutbox\par\egroup
  \ST@dimen=\ht\ST@pbox
  \advance\ST@dimen by \dp\ST@pbox
  \ifnum\ST@pboxht<\ST@dimen
    \global\ST@pboxht=\ST@dimen
  \fi
  \ST@dimen=\z@
  \unvbox\ST@pbox\egroup\hfil}

```

`\estimate@lineht`

Estimates the height of normal line taking `\arraystretch` into account. Also computes the difference between a ‘normal’ line and a stretched one.

```
\def\estimate@lineht{%
  \ST@lineht=\arraystretch \baselineskip
  \global\advance\ST@lineht by 1\p@
  \ST@stretchht\ST@lineht\advance\ST@stretchht-\baselineskip
  \ifdim\ST@stretchht<\z@\ST@stretchht\z@\fi
  \ST@trace\tw@{Basic line height: \the\ST@lineht\MessageBreak%
                Arrayed line height: \the\ST@stretchht}%
  \global\advance\ST@lineht \PWST@xentrystretch\ST@lineht
  \ST@trace\tw@{Stretched line height: \the\ST@lineht}}

```

`\@calfirstpageht` Estimates the space left on the current page and decides whether the tabular can be started on this page or on a new page. Aspects of the original code are modified for the extension.

```
\def\@calfirstpageht{%
  \ST@trace\tw@{Calculating height of xtabular on first page}%
```

The TEX register `\pagetotal` contains the height of the page sofar, the LATEX register `\@colroom` contains the height of the column.

```
  \global\ST@pagesofar\pagetotal
  \global\ST@pageleft\@colroom
  \ST@trace\tw@{Height of previous text = \the\pagetotal; \MessageBreak
                Height of column = \the\ST@pageleft}%
```

When we are in twocolumn mode TEX may still be collecting material for the first column although there seems to be no space left. In this case we have to check against two times `\ST@pageleft`.

```
  \if@twocolumn
    \ST@trace\tw@{two column mode}%
    \if@firstcolumn
      \ST@trace\tw@{First column}%
      \ifnum\ST@pagesofar > \ST@pageleft
        \global\ST@pageleft=2\ST@pageleft
        \ifnum\ST@pagesofar > \ST@pageleft
          \newpage\@calnextpageht
          \ST@trace\tw@{starting new page}%
        \else
```

In this case we're in the second column, so we have to compensate for the material in the first column.

```
          \ST@trace\tw@{Second column}%
          \global\advance\ST@pageleft -\ST@pagesofar
          \global\advance\ST@pageleft -\@colroom
        \fi
```

When `\ST@pagesofar` is smaller than `\ST@pageleft` TEX is still collecting material for the first column, so we can start a new tabular environment like we do on a single column page.

```
      \else
        \global\advance\ST@pageleft by -\ST@pagesofar
        \global\ST@pagesofar\z@
       \fi
     \else
```

When we end up here, TEX has already decided it had enough material for the first column and is building the second column.

```
     \ST@trace\tw@{Second column}%
     \ifnum\ST@pagesofar > \ST@pageleft
       \ST@trace\tw@{starting new page}%
       \newpage\@calnextpageht
     \else
       \global\advance\ST@pageleft by -\ST@pagesofar
       \global\ST@pagesofar\z@
     \fi
   \fi
 \else
```

In one column mode there is a simple decision.

```
    \ST@trace\tw@{one column mode}%
    \ifnum\ST@pagesofar > \ST@pageleft
      \ST@trace\tw@{starting new page}%
    \newpage\@calnextpageht
```

When we are not starting a new page subtract the size of the material already on it from the available space.

```
   \else
     \global\advance\ST@pageleft by -\ST@pagesofar
     \global\ST@pagesofar\z@
   \fi
 \fi
 \ST@trace\tw@{Available height: \the\ST@pageleft}%
```

Now we need to know the height of the head of the table. In order to measure this we typeset it in a normal `tabular` environment.

```
 \ifx\@@tablehead\@empty
   \ST@headht=\z@
 \else
   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
     \ST@restore
     \expandafter\tabular\expandafter{\ST@tableformat}%
     \@@tablehead\endtabular}%
   \ST@headht=\ht\@tempboxa\advance\ST@headht\dp\@tempboxa
 \fi
 \ST@trace\tw@{Height of head: \the\ST@headht}%
```

To decide when to start a new page, we need to know the vertical size of the tail of the table.

```
 \ifx\@tabletail\@empty
   \ST@tailht=\z@
 \else
   \setbox\@tempboxa=\vbox{\@arrayparboxrestore
```

```
    \ST@restore
    \expandafter\tabular\expandafter{\ST@tableformat}%
      \@tabletail\endtabular}%
  \ST@tailht=\ht\@tempboxa\advance\ST@tailht\dp\@tempboxa
\fi
```

We add the average height of a line to this because when we decide to continue the tabular we need to have enough space left for one line and the tail.

```
\advance\ST@tailht by \ST@lineht
\ST@trace\tw@{Height of tail: \the\ST@tailht}%
```

```
\ST@trace\tw@{Maximum space for xtabular: \the\ST@pageleft}%
```

Now we decide whether we can continue on the current page or whether we need to start a new page. We assume that the minimum height of a tabular is the height of the head and tail and one line of data. If that doesn't fit, start a new page.

```
\@tempdima\ST@headht
\advance\@tempdima\ST@lineht
\advance\@tempdima\ST@tailht
```

Extension: I also add the height of the caption to the required space. The amount to be added depends on whether it is a top or bottom caption. Allowance is also made for skips around the caption.

```
\if@topcaption
  \setbox\@tempboxa=\vbox{\PWST@thecaption}%
  \PWSTcapht=\ht\@tempboxa\advance\PWSTcapht\dp\@tempboxa
  \advance\PWSTcapht by 20\p@
\else
  \PWSTcapht = 10\p@
\fi
\ST@trace\tw@{Caption height: \the\PWSTcapht}%
\advance\@tempdima\PWSTcapht
```

Continue with the original code.

```
\ST@trace\tw@{Minimum height of xtabular: \the\@tempdima}%
\ifnum\@tempdima>\ST@pageleft
  \ST@trace\tw@{starting new page}%
```

Extension: The next line in the original code is `\newpage\@calnextpageht`. I need to start a new page, making allowance for the space required by the caption.

```
  \newpage
  \global\ST@pageleft\@colroom
  \global\advance\ST@pageleft by -\PWSTcapht
  \global\ST@pagesofar=\z@
```

Finish up with the original code.

```
\fi}  % end \@calfirstpageheight

```

`\@calnextpageht` This calculates the maximum height of the tabular on all subsequent pages of the supertabular environment.

```
\def\@calnextpageht{%
  \ST@trace\tw@{Calculating height of xtabular on next page}%
  \global\ST@pageleft\@colroom
  \global\ST@pagesofar=\z@
  \ST@trace\tw@{Maximum space for xtabular: \the\ST@pageleft}}

```

\PWSTcalchtlines *Extension:* A macro to calculate the space required by an empty table and the number of lines in an empty table.

The appropriate heads and tails are typeset in a temporary box so we can measure them.

```
\newcommand{\PWSTcalchtlines}{%
```

Measure the lasttail.

```
  \setbox\@tempboxa=\vbox{\@arrayparboxrestore
    \ST@restore
    \expandafter\tabular\expandafter{\ST@tableformat}%
    \@table@last@tail\endtabular}%
  \PWST@ht=\ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
  \global\PWST@lastht = \PWST@ht
```

And repeat for the lasthead.

```
  \setbox\@tempboxa=\vbox{\@arrayparboxrestore
    \ST@restore
    \expandafter\tabular\expandafter{\ST@tableformat}%
    \@table@last@head\endtabular}%
  \PWST@ht = \ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
  \global\advance\PWST@lastht by \PWST@ht
  \ST@trace\tw@{Height of empty xtabular on last page = \the\PWST@lastht}%
```

Now repeat pretty well all of the above for a general table (i.e., one that is not on the first page nor the designated last page).

First the tail.

```
  \setbox\@tempboxa=\vbox{\@arrayparboxrestore
    \ST@restore
    \expandafter\tabular\expandafter{\ST@tableformat}%
    \@tabletail\endtabular}%
  \PWST@ht=\ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
  \global\PWST@generalht = \PWST@ht
```

And on to the general head.

```
  \setbox\@tempboxa=\vbox{\@arrayparboxrestore
    \ST@restore
    \expandafter\tabular\expandafter{\ST@tableformat}%
    \@tablehead\endtabular}%
  \PWST@ht = \ht\@tempboxa\advance\PWST@ht\dp\@tempboxa
  \global\advance\PWST@generalht by \PWST@ht}

```

\PWSTcalnextpageht *Extension:* From some experiments that I ran it appeared as though the supertabular package ignored the possibility that the space required for the table header and tail on pages after the first one might be different. If the subsequent head/tail combination were longer (i.e., took more vertical space) then the table could overflow the page. This is an attempt to fix this problem by calculating the actual minimum space required after the first page.

The calculations are similar to, but simpler, than those for `\@calfirstpageht`.

```
\newcommand{\PWSTcalnextpageht}{%
  \ifnum\PWSTcurpage = \PWSTpenultimate
    \ST@trace\tw@{Calculating height of xtabular on last page}%
```

We are on the penultimate page, so get the height of the last head/tail.

```
    \PWST@ht=\PWST@lastht
```

Otherwise I need the general page data.

```
  \else
    \ST@trace\tw@{Calculating height of xtabular on next general page}%
    \PWST@ht=\PWST@generalht
  \fi
```

Having dealt with the two cases, I can now calculate the minimum space for a supertabular on the following page.

```
  \global\ST@pageleft\@colroom
  \global\advance\ST@pageleft -\PWST@ht
  \global\ST@pagesofar=\z@
  \ST@trace\tw@{Available space for xtabular: \the\ST@pageleft}}

```

`\x@supertabular` The various supertabular environments share a lot of code. Thus, to avoid needless repetition, the shared code is defined in this macro.

This macro has been modified as part of the supertabular extension.

```
\def\x@supertabular{%
```

First save the original definition of `\tabular` and then make it point to `\inner@tabular`. This is done to enable supertabular cells to contain a `tabular` environment without getting unexpected results when the `supertabular` would be split across this inner `tabular` environment.

```
  \let\org@tabular\tabular
  \let\tabular\inner@tabular
```

The same has to be done for the `tabular*` environment. The coding is more verbose.

```
  \expandafter\let
    \csname org@tabular*\expandafter\endcsname
    \csname tabular*\endcsname
  \expandafter\let\csname tabular*\expandafter\endcsname
    \csname inner@tabular*\endcsname
```

Extension: The original code printed out the top caption at this point. If there is too little space on the first page of the table, the tabular data is printed on the following page. If this is the case (and its not known yet whether it is), then the caption should also be printed on the following page.

```
%%%%  \if@topcaption \@process@tablecaption \fi
```

Back to the original code. Save the original definition of `\\`.

```
\global\let\@oldcr=\\
```

Save the current value of `\baselineskip`, as we need it in the calculation of the average height of a line.

```
\def\baslineskp{\baselineskip}%
```

We have to check whether `array.sty` was loaded, because some of the internal macros have different names.

```
\ifx\undefined\@classix
```

Save old `\@tabularcr` and insert the definition of `\@stabularcr`.

```
  \let\org@tabularcr\@tabularcr
  \let\@tabularcr\ST@tabularcr
```

Activate the new parbox algorithm.

```
  \let\org@startpbox=\@startpbox
  \let\org@endpbox=\@endpbox
  \let\@@startpbox=\ST@startpbox
  \let\@@endpbox=\ST@endpbox
\else
```

When `array.sty` was loaded things are a bit different.

```
  \let\org@tabularcr\@arraycr
  \let\@arraycr\ST@tabularcr
  \let\org@startpbox=\@startpbox
  \let\org@endpbox=\@endpbox
  \let\@startpbox=\ST@astartpbox
  \let\@endpbox=\ST@aendpbox
\fi
```

Check if the head of the table should be different for the first and subsequent pages.

```
\ifx\@table@first@head\undefined
  \let\@@tablehead=\@tablehead
\else
  \let\@@tablehead=\@table@first@head
\fi
```

The first part of a supertabular may be moved to the next page if it doesn't fit on the current page. Subsequent parts can not be moved; therefore we will have to switch the definition of `\ST@skippart` around.

```
\let\ST@skippage\ST@skipfirstpart
```

Now we can estimate the average line height and the height of the first page of the supertabular.

```
\estimate@lineht
\@calfirstpageht
```

Extension: Call the macro to initialize the extension code for this table.

```
\PWSTinit
```

Extension: At this point I know, and have adjusted for, the page on which the first part of the table will be printed. It should now be safe to print the top caption, if any. Unfortunately, in spite of everthing, the TEX page breaking mechanism might still think that there is too little space left.

```
\if@topcaption \@process@tablecaption \fi
\noindent
```

Extension: Finally, subtract the space required by the header and the tail (as these don't update the available space when output).

```
\global\advance\ST@pageleft -\ST@headht%
\ST@trace\tw@{Available space after accounting for header: \the\ST@pageleft}%
\global\advance\ST@pageleft -\ST@tailht%
\ST@trace\tw@{Available space after accounting for tail: \the\ST@pageleft}}

```

`\PWSTinit` *Extension:* This routine initialises the extension data.

```
\newcommand{\PWSTinit}{%
```

At the end of processing each supertabular (see later) the number of pages consumed by the supertabular is written to the `.aux` file. At the start of a supertabular, after incrementing the number of supertabulars processed, the prior number of pages are read from the file. These are stored in *PWSTlastpage*.

```
\global\advance\c@PWSTtable\@ne
\global\expandafter\let\expandafter\PWSTtempc
  \csname PWST@\romannumeral\c@PWSTtable\endcsname
```

I have to take account of the fact that there might be no entry in the `.aux` file, and hence the lastpage number might not be set.

```
\ifx\PWSTtempc\relax
  \ST@trace\tw@{Table \the\c@PWSTtable: Processing for the first time}%
  \PWSTlastpage=\@m% = 1000
\else
  \PWSTlastpage=\PWSTtempc
\fi
\ST@trace\tw@{Table \the\c@PWSTtable: last page set to \the\PWSTlastpage}%
```

Set the current page counter to unity.

```
\PWSTcurpage=\@ne
```

Perform the calculations for the empty table data.

```
\PWSTcalchtlines
```

Initialise the line counter and set `firstcall` to TRUE.

```
\global\PWSTlines=\z@
\global\firstcalltrue
```

If we have the `iso` class, then I have to flag that we are in a 'float'.

```
  \infloattrue}

```

`\xtabular` We start by looking for an optional argument, which will be duly ignored as it seems to make no sense to try to align a multipage table in the middle...

Extension: Use `xtabular` instead of `supertabular`, and similarly for the others, so this will not be mentioned explicitly again.

```
\def\xtabular{%
  \@ifnextchar[{\@supertabular}%]
               {\@supertabular[]}}
```

`\@supertabular` We can now save the preamble of the tabular in a macro.

```
\def\@supertabular[#1]#2{%
  \def\ST@tableformat{#2}%
  \ST@trace\tw@{Starting a new xtabular}%
```

Then remember that this is not a supertabular* environment.

```
  \global\ST@starfalse
```

Don't use minipages.

```
  \global\ST@mpfalse
```

Most of the following code is shared between the `supertabbular` and `supertabular*` environments. So to avoid duplication it is stored in a macro.

```
  \x@supertabular
```

Finally start a normal `tabular` environment.

```
  \expandafter\org@tabular\expandafter{\ST@tableformat}%
  \@@tablehead}

```

`\xtabular*` We start by looking for the optional argument of the tabular environment.

```
\@namedef{xtabular*}#1{%
  \@ifnextchar[{\@nameuse{@supertabular*}{#1}}%
               {\@nameuse{@supertabular*}{#1}[]}%]
  }
```

We start by saving the intended width and the preamble of the `tabular*`.

```
\@namedef{@supertabular*}#1[#2]#3{%
  \ST@trace\tw@{Starting a new xtabular*}%
  \def\ST@tableformat{#3}%
  \ST@wd=#1\relax
  \global\ST@startrue
  \global\ST@mpfalse
```

Now we can call the common code for both environments.

```
  \x@supertabular
```

And we can start a normal `tabular*` environment.

```
  \expandafter\csname org@tabular*\expandafter\endcsname
  \expandafter{\expandafter\ST@wd\expandafter}%
  \expandafter{\ST@tableformat}%
  \@@tablehead}

```

`\mpxtabular` This version of the supertabular environment puts each tabular into a minipage, thus making footnotes possible. We start by looking for an optional argument, which will be ignored as it makes no sense to try and align a multipage table in the middle…

```
\def\mpxtabular{%
  \@ifnextchar[{\@mpsupertabular}%]
               {\@mpsupertabular[]}}
```

We can now save the preamble in a macro.

```
\def\@mpsupertabular[#1]#2{%
  \def\ST@tableformat{#2}%
  \ST@trace\tw@{Starting a new mpxtabular}%
```

Remember that this is not a `mpsupertabular*` environment and also note we have to close the minipage later.

```
  \global\ST@starfalse
  \global\ST@mptrue
```

Since we are about to start a minipage of `\columnwidth` the horizontal alignment will not work. We have to remember the values and then restore them inside the minipage.

```
  \ST@rightskip \rightskip
  \ST@leftskip \leftskip
  \ST@parfillskip \parfillskip
```

Call the code that is common to all the environments.

```
  \x@supertabular
```

Finally, start a normal `tabular`

```
  \minipage{\columnwidth}%
  \parfillskip\ST@parfillskip
  \rightskip \ST@rightskip
  \leftskip \ST@leftskip
  \noindent\expandafter\org@tabular\expandafter{\ST@tableformat}%
  \@@tablehead}

```

`\mpxtabular*` We start by looking for the optional argument of the tabular environment.

```
\@namedef{mpxtabular*}#1{%
  \@ifnextchar[{\@nameuse{@mpsupertabular*}{#1}}%
               {\@nameuse{@mpsupertabular*}{#1}[]}%]
}
```

Now we can save the intended width and the preamble of the `tabular*`.

```
\@namedef{@mpsupertabular*}#1[#2]#3{%
  \ST@trace\tw@{Starting a new mpxtabular*}%
  \def\ST@tableformat{#3}%
  \ST@wd=#1\relax
  \global\ST@startrue
  \global\ST@mptrue
  \ST@rightskip \rightskip
  \ST@leftskip \leftskip
  \ST@parfillskip \parfillskip
```

Now is the time to call the common code for both environments.

```
  \x@supertabular
```

And we can start a normal `tabular*` environment.

```
  \minipage{\columnwidth}%
  \parfillskip\ST@parfillskip
  \rightskip \ST@rightskip
  \leftskip \ST@leftskip
  \noindent\expandafter\csname org@tabular*\expandafter\endcsname
  \expandafter{\expandafter\ST@wd\expandafter}%
  \expandafter{\ST@tableformat}%
  \@@tablehead}%

```

`\endxtabular` `\endxtabular*` These close the `xtabular` and `xtabular*` environments.

For the extension, this macro has been modified to write out to the `.aux` file the number of pages used for the supertabular.

```
\def\endxtabular{%
  \ifx\@table@last@tail\undefined
    \@tabletail
  \else
    \@table@last@tail
  \fi
  \csname endtabular\ifST@star*\fi\endcsname
```

While studying the original code to determine where additions were needed for the extension, I realized that the last part of the `\end...` code was common to all the environments. I have broken it out into a seperate routine which also includes the modification needed for the extension.

```
  \x@endsupertabular
```

And back to the original code.

```
  \ST@trace\tw@{Ended a xtabular\ifST@star*\fi}}
```

The definition of the ending of the `xtabular*` environment is simple:

```
\expandafter\let\csname endxtabular*\endcsname\endxtabular
```

`\x@endsupertabular` This macro contains the code that is common to all the `\end...` commands. It includes the modification required for the extension.

```
\newcommand{\x@endsupertabular}{%
```

Restore the original definition of `\@tabularcr`

```
  \ST@restore
```

Check if we have to insert a caption and restore to default behaviour of putting captions at the top.

```
  \if@topcaption
  \else
```

```
    \@process@tablecaption
    \global\@topcaptiontrue
  \fi
```

Restore the meaning of `\\` to the one it had before the start of this environment. Also re-initialize some control-sequences

```
  \global\let\\=\@oldcr
  \global\let\@table@first@head\undefined
%%%  \global\let\@table@last@tail\undefined
  \global\let\@process@tablecaption\relax
```

Extension: For the extension, write the number of the last page to the `.aux` file. Also, if we are in the `iso` class, reset the 'float' flag.

```
  \PWSToplastpagenum
    \infloatfalse}

```

`\PWSToplastpagenum` *Extension:* This routine is responsible for writing the number of the last page of the supertabular to the `.aux` file.

What gets written is `\PWST@vi{4}`, assuming that the value of *c@PWSTtable* is 6 and the value of *PWSTcurpage* is 4.

```
\newcommand{\PWSToplastpagenum}{%
```

There are a number of cases to consider. The first decision is whether the current page is the previously calculated last page.

```
  \ifnum\PWSTcurpage=\PWSTlastpage
```

The current table ends on the calculated last page. There are four cases to consider:

1. The table has not overflowed (`firstcall` is TRUE) and the table is not empty — this page is still the last page.

2. The table has not overflowed (`firstcall` is TRUE) and the table is empty — this page is after the actual last page, so decrease the page number.

3. The table has overflowed (`firstcall` is FALSE) and the overflow is large enough to generate a non-empty table on the next page — increment the page number.

4. The table has overflowed (`firstcall` is FALSE) and the overflow is small enough to generate an empty table on the next page — this page is still the last page.

```
    \iffirstcall  % on last, no overflow
%%      \ifnum\PWSTlines < \PWSTlasthead % this table is empty
      \ifnum\PWSTlines < \@ne           % this table is empty
        \global\advance\PWSTcurpage \m@ne
      \fi
    \else % overflow
%%      \ifnum\PWSTlines > \tw@ % enough for another page
```

```
      \ifnum\PWSTlines > \z@    % enough for another page
        \global\advance\PWSTcurpage \@ne
      \fi
    \fi
  \else
```

The table has ended on a page that is not the calculated last page. If the table is empty, then decrement the page number, else this is the last page.

```
%%    \ifnum\PWSTlines < \PWSThead % empty table
    \ifnum\PWSTlines < \@ne        % empty table
      \global\advance\PWSTcurpage \m@ne
    \fi
  \fi
```

Finally, write out the 'new' last page number.

```
  \if@filesw\immediate\write\@auxout%
    {\gdef\string\PWST@\romannumeral\c@PWSTtable{\the\PWSTcurpage}}%
    \ST@trace\tw@{Table \the\c@PWSTtable:\MessageBreak
                  wrote \the\PWSTcurpage\space as the last page}%
  \fi}

```

`\endmpxtabular` `\endmpxtabular*` These close the `mpxtabular` and `mpxtabular*` environments.

```
\def\endmpxtabular{%
  \ifx\@table@last@tail\undefined
    \@tabletail
  \else
    \@table@last@tail
  \fi
  \csname endtabular\ifST@star*\fi\endcsname
  \endminipage
```

Now call the common code for all `\end...`.

```
  \x@endsupertabular
```

Finish per the original code.

```
  \ST@trace\tw@{Ended an mpxtabular\ifST@star*\fi}}
```

The definition of the ending of the `mpxtabular*` environment is simple:

```
\expandafter\let\csname endmpxtabular*\endcsname\endmpxtabular

```

`\ST@restore` This macro restores the original definitions of the macros that handle parbox entries and the 'end of row' macros.

```
\def\ST@restore{%
  \ifx\undefined\@classix
    \let\@tabularcr\org@tabularcr
  \else
    \let\@arraycr\org@tabularcr
  \fi
  \let\@startpbox\org@startpbox
  \let\@endpbox\org@endpbox}

```

\inner@tabular
\inner@tabular*

In order to facilitate complete `tabular` environments to be in a cell of a `supertabular` we need to adapt the definition of the original environments. For the inner `tabular` a number of definitions have to be restored.

```
\def\inner@tabular{%
  \ST@restore
  \let\\=\@oldcr
  \noindent
  \org@tabular}
\@namedef{inner@tabular*}{%
  \ST@restore
  \let\\=\@oldcr
  \noindent
  \csname org@tabular*\endcsname}

```

\ST@cr

This macro is called by each `\\` inside the tabular environment. It updates the estimate of the amount of space left on the current page and starts a new page if necessary.

```
\def\ST@cr{%
  \noalign{%
    \ST@trace\thr@@{Parbox height: \the\ST@pboxht\MessageBreak
                    Line height: \the\ST@lineht}%
    \ifnum\ST@pboxht<\ST@lineht
```

If there is a non-empty line, but an empty parbox, then `\ST@pboxht` might be non-zero, but too small thereby breaking the algorithm. Therefore we estimate the height of the line to be `\ST@lineht` in this case, and store it in `\ST@prevht`.

```
      \global\advance\ST@pageleft -\ST@lineht
      \global\ST@prevht\ST@lineht
    \else
```

When the parbox is not empty we take its height into account plus a little extra.

```
      \global\advance\ST@pboxht \PWST@xentrystretch\ST@pboxht
      \global\advance\ST@pboxht \ST@stretchht
      \ST@trace\thr@@{Added par box with height \the\ST@pboxht}%
      \global\advance\ST@pageleft -\ST@pboxht
      \global\ST@prevht\ST@pboxht
      \global\ST@pboxht\z@
    \fi
```

`\ST@toadd` is the value of the optional argument of `\\`.

```
     \global\advance\ST@pageleft -\ST@toadd
     \global\ST@toadd=\z@
     \ST@trace\thr@@{Space left for xtabular: \the\ST@pageleft}%
```

Extension: Increment the line number at this point.

```
     \global\advance\PWSTlines \@ne
```

```
    \ST@trace\thr@@{Line counter incremented by one to: \the\PWSTlines}%
  }% end of noalign
```

In general, when the `\ST@pageleft` has become negative, the last row was so high that the supertabular doesn't fit on the current page. In this case we skip the current page and start at the top of the next one; otherwise TEX will move this part of the table to a new page anyway, probably with a message about an overfull `\vbox`.

Extension: For the extension I do some special handling if we are on the last page. Essentially the idea is not to start a new page, but to continue on the current page, noting any overflow.

```
  \ifnum\PWSTcurpage=\PWSTlastpage
    \PWST@lastpagecr
  \else
```

Execute the original code.

```
    \ifnum\ST@pageleft<\z@
      \ST@skippage
    \else
```

When there is not enough space left on the current page, we start a new page. To compute the amount of space needed we use the height of the previous line (`\ST@prevht`) as an estimate of the height of the next line. If we are processing an `mpsupertabular` we also need to take the height of the footnotes into account.

```
      \noalign{\global\@tempdima\ST@tailht
        \global\advance\@tempdima\ST@prevht
      \ifST@mp
        \ifvoid\@mpfootins\else
          \global\advance\@tempdima\ht\@mpfootins
          \global\advance\@tempdima 3pt
        \fi
      \fi}% end noalign
      \ifnum\ST@pageleft<\@tempdima
        \ST@newpage
      \else
```

This line is necessary because the tablehead has to be inserted *after* the `\if\else\fi`-clause. For this purpose `\ST@next` is used. In the middle of tableprocessing it should be an *empty* macro (*not* `\relax`).

```
        \noalign{\global\let\ST@next\@empty}%
      \fi
    \fi
```

Extension: Close off the `\iflastpage`;

```
  \fi
```

and finish per the original code.

```
  \ST@next}

```

`\PWST@lastpagecr` *Extension:* This routine handles newlines on the last page of a supertabular. The idea is that when we are on the last page the table continues to be processed until the end without calling for a newpage even if the table will be too long. I do need to record whether or not the table has 'overflowed' the allowable space on the page. The code is very similar to the last part of the code for `\ST@cr`.

```
\newcommand{\PWST@lastpagecr}{%
 \noalign{%
   \ifnum\ST@pageleft<\z@
```

The table has overflowed, so record the fact.

```
     \PWST@setfirstcall
   \fi
```

Now continue along the lines of `\ST@cr`.

```
   \global\@tempdima\ST@tailht
   \global\advance\@tempdima\ST@prevht
     \ifST@mp
       \ifvoid\@mpfootins\else
         \global\advance\@tempdima\ht\@mpfootins
         \global\advance\@tempdima 3pt
       \fi
     \fi
     \ifnum\ST@pageleft<\@tempdima
```

Again, the table has overflowed.

```
       \PWST@setfirstcall
     \fi
```

Finish like `\ST@cr`.

```
     \global\let\ST@next\@empty
     }}

```

`\PWST@setfirstcall` *Extension:* This routine records that a table on the last page has overflowed by setting `firstcall` to FALSE. If it is the first such overflow it also zeroes the line counter.

```
\newcommand{\PWST@setfirstcall}{%
 \iffirstcall
   \global\firstcallfalse
   \global\PWSTlines=\z@
   \ST@trace\thr@@{Overflow on last page. Line counter set to \the\PWSTlines}%
 \fi}

```

`\ST@skipfirstpart` This macro skips the current page and moves the entire supertabular that has been built so far to the next page.

```
\def\ST@skipfirstpart{%
 \noalign{%
   \ST@trace\tw@{Tabular too high, moving to next page}%
```

In order for this to work properly we need to adapt the value of `\ST@pageleft`. When this macro is called it has a negative value. We should add the height of the next page to that (`\@colroom`). From the result the 'normal' height of the supertabular should be subtracted (`\@colroom - \pagetotal`). This could be coded as follows:

```
\ST@dimen\@colroom
\advance\ST@dimen-\pagetotal
\global\advance\ST@pageleft\@colroom
\global\advance\ST@pageleft-\ST@dimen
```

However, note that `\@colroom` is added *and* subtracted. Thus the code can be simplified to:

```
    \global\advance\ST@pageleft\pagetotal
```

Then we can set `\ST@pagesofar` to zero and start the new page.

```
    \global\ST@pagesofar\z@
    \newpage
```

Finally we make sure that this macro can only be executed once for each supertabular by changing the definition of `\ST@skippage`.

```
    \global\let\ST@skippage\ST@newpage
  }}

```

`\ST@newpage` This macro performs the actions necessary to start a new page.

This macro is also modified for the extension to supertabluar.

```
\def\ST@newpage{%
  \noalign{\ST@trace\tw@{Starting new page, writing tail}}%
```

Output `\tabletail`, close the tabular environment, close a minipage if necessary, output all material and start a fresh new page.

```
  \@tabletail
  \ifST@star
    \csname endtabular*\endcsname
  \else
    \endtabular
  \fi
  \ifST@mp
    \endminipage
  \fi
```

Then we make sure that `\ST@skippage` can no longer be executed for this supertabular by changing its definition.

```
  \global\let\ST@skippage\ST@newpage
```

On with the output.

Extension: The original code had the next line as `\newpage\@calnextpageht`. However, if the general header has a vertical height that differs from the first header, then the table on the continuation pages may run short or, more disconcerting, long. The extension, I think, cures that by using a different algorithm to calculate the height on the next page.

```
  \newpage\PWSTcalnextpageht
  \ST@trace\tw@{writing head}%
```

Extension: The original code just let `\ST@next` to `\@tablehead`. The extension has to handle the special case of of the heading on the last page.

```
  \PWSTsethead
```

Now we are back to the original supertabular code.

```
  \ifST@mp
    \noindent\minipage{\columnwidth}%
    \parfillskip\ST@parfillskip
    \rightskip \ST@rightskip
    \leftskip \ST@leftskip
  \fi
  \noindent
  \ifST@star
    \expandafter\csname org@tabular*\expandafter\endcsname
    \expandafter{\expandafter\ST@wd\expandafter}%
    \expandafter{\ST@tableformat}%
  \else
    \expandafter\org@tabular\expandafter{\ST@tableformat}%
  \fi}

```

`\PWSTsethead` *Extension:* This is more extension code for use within `\ST@newpage`. It provides the proper table head for the page about to be processed.

```
\newcommand{\PWSTsethead}{%
```

First the line counter is zeroed.

```
  \global\PWSTlines=\z@
  \ST@trace\thr@@{Newpage, line counter set to: \the\PWSTlines}%
```

The current page counter is incremented and it is checked against the old page counter to see if this is the last page of this supertabular.

```
  \global\advance\PWSTcurpage\@ne
  \ST@trace\tw@{Table \the\c@PWSTtable:\MessageBreak
                current page = \the\PWSTcurpage,\MessageBreak
                last page = \the\PWSTlastpage}%
  \ifnum\PWSTcurpage=\PWSTlastpage
    \ST@trace\tw@{Newpage is the last page}%
```

We are on the last page. If there are more than one pages and the last table heading has been specified, then the heading is set to `\@table@last@head`, otherwise it is set to `\@tablehead`.

```
    \ifnum\PWSTcurpage>\@ne
      \ifx\@table@last@head\relax
        \let\ST@next\@tablehead
        \ST@trace\tw@{Set heading to tablehead}%
```

```
638       \else
639         \let\ST@next\@table@last@head
640         \ST@trace\tw@{Set heading to tablelasthead}%
641       \fi
642     \fi
643   \else
```

We are not on the last page, so just set the heading to `\@tablehead`.

```
644     \let\ST@next\@tablehead
645     \ST@trace\tw@{Set heading to tablehead}%
646   \fi}
647
```

The end of this package

648 ⟨/xtab⟩

References

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.

[Wil96] Peter R. Wilson. *LaTeX for standards: The LaTeX package files user manual.* NIST Report NISTIR, June 1996.

Index

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