The cjwplain Package∗
(PLAIN\TeX\ under \LaTeX\2ε)

Colin J. Wynne†

1998/08/31

Contents

1 Package Options 3
1.1 Regular Options ............................................. 3
1.2 Special Options ............................................. 4

2 The Code 4
2.1 Declarations .................................................. 4
2.2 Allocation Calls: outerallocs ................................ 6
2.3 Error Processing: diagnostics ............................ 6
2.4 Skips: plainskips ........................................... 6
2.5 Fonts ............................................................. 7
2.6 The \line\ Macro: strictline ............................... 7
2.7 Tab Alignments: tabbing .................................... 7
2.8 Itemising: strictitem ....................................... 8
2.9 Miscellaneous .................................................. 9
2.9.1 Sectioning .................................................. 9
2.9.2 Proclamations ............................................. 9
2.9.3 Paragraph Formatting .................................... 9
2.9.4 Accents and Miscellaneous ............................ 10
2.9.5 Ending the Document ................................... 10
2.9.6 Math Commands .......................................... 10
2.10 Math Alignment: eqalign ................................... 11
2.11 Output Routine: plainoutput ................................ 11
2.11.1 Page Numbering, Running Heads and Miscellaneous ... 13
2.11.2 Insertions .................................................. 14
2.11.3 Footnotes: strictfootnotes, altfootnotes ............... 14
2.11.4 Magnification: magnification ........................... 15

Introduction

I first started using \TeX\ some two-and-a-half years ago, having been introduced
to it by several \TeX\nophiles in my college math department. I was aware from the

∗This file has version 1.2.
†E-Mail at: cwynne@brutus.mts.jhu.edu, cwynne@jhu.edu.
start that there was a somehow ‘bastardised’ version of this very good program which went by the name ‘Lam\TeX’—invariably referred to by my \TeX mentors as ‘Lame\TeX’. Most of you have probably heard this epithet before.

Well, I count it as a good thing that I was discouraged from using L\TeX at first, as I ended up writing quite a lot of \TeX code for myself before I ever got around to actually reading The \TeXbook in its entirety, which I did only a few months ago. My first real project for \TeX was writing a large macro set for my undergraduate thesis—table of contents, marks for the running heads, chapter and section delineation and so forth. This growing library was expanded as I decided I wanted a good set of macros for writing outlines and by requirements for various papers, such as using endnotes in lieu of footnotes. This rather haphazard collection of mine underwent a major change when I found the macro package for NFSS (version 1) under Plain \TeX, of which I promptly took advantage.

This year I finally got my own computer, mainly to run \TeX. Given my newly purchased copy of The \TeXbook and some free time, I began to try to organise that cluster of code. Having learned something in the meanwhile about generic markup, and why it is preferable, I started rewriting for more generalisation. Also in the meanwhile, L\TeX\,2\,\varepsilon had come along, greatly enhancing L\TeX’s own use of generic markup. It also standardised the NFSS, which I had so come to appreciate. Basically, between the various chunks of L\TeX\,2\,\varepsilon which I had already hacked to work under my custom format and the movement towards increasingly generic code by both myself and the L\TeX folks, that a convergence was taking place, so I finally decided to give L\TeX\,2\,\varepsilon a serious looking-at.

I started by printing out the documented source code. I liked a lot of what I saw—but there were two problems. Some of my favourite bits of Plain \TeX got left by the wayside. For doing a lot of mathematics, I still find \texttt{\textbackslash eqalign} to be the easiest way of aligning a bunch of related equations. First, it involves less typing than a \texttt{\textbackslash begin}...\texttt{\textbackslash end} pair, and I don’t often need equation numbers—something not easily done away with under vanilla L\TeX. The bigger concern was that I had a bunch of source files that were written under Plain \TeX (or, rather, under my Plain \TeX), and I didn’t want to have to make the dozens of minor modifications necessary to get them to work under L\TeX.

So, I decided to learn how L\TeX does things and to do so by writing a package that would, at its simplest, let me add a \texttt{\documentclass} and a \texttt{\begin\{document\}} line to one of my existing Plain \TeX source files and get it to compile under L\TeX.

Thus, I have written my first L\TeX package. I consider the main feature to be the ability to very easily add NFSS commands to a document written under Plain \TeX. Secondly, maybe it will help convince some other Plain \TeX diehards to give L\TeX a try, inasmuch as all of their standard commands will be supported. Finally, it should let those who use L\TeX exclusively to easily deal with Plain \TeX files if the need arises.
Feel free to let me know if you find this package useful or, of course, if you find any bugs or wish to suggest improvements.

This Package
This package is built over the file \texttt{ltplain.dtx}, or, more correctly, over those parts of \texttt{ltplain.dtx} which were changes to or omissions of the original \textsc{plain} \TeX source. Some parts, specifically font changes, have not been reproduced in their entirety, due basically to the fact that such would be a pointless exercise. See the comments in Section 2.5 for the explanation.

Finally, this document prints with all source code because I feel the source itself, and the modifications to it, are the best documentation.

1 Package Options
According to the documented \LaTeX source file \texttt{ltplain.dtx},

\begin{itemize}
\item \LaTeX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain \TeX format described in Appendix B of the \TeXBook.
\end{itemize}

It seems to me that removing the qualifying 'almost' would be no bad thing.

The idea behind the available options is that a given user may need only certain aspects of \textsc{plain} \TeX added back in for a document. Furthermore, the additional code can sometimes be specified in different ways—\textit{i.e.}, either strictly according to the definitions of \textsc{plain} \TeX or in a manner syntactically identical to \textsc{plain} \TeX but functionally grounded in \LaTeX. The overall goal, though, is completeness; I have therefore included everything, in one form or another, even when I can’t think of a reason why some things would be necessary.

There are nineteen regular and two special options available for the \texttt{cjwplain} package. (All are entered in standard \LaTeX form, as optional arguments to the \texttt{\usepackage} command. I only call two of them ‘special’ so as to draw attention to them.)

1.1 Regular Options
The regular options are:

\begin{verbatim}
  \begin{verbatim}
  \bf Options\end{verbatim}
\end{verbatim}

\begin{verbatim}
  \begin{verbatim}
  outerallocs diagnostics plainskips
  outerallocsoff diagnosticsoff plainskipsoff
  strictline tabbing strictitem
  strictlineoff tabbingoff strictitemoff
  eqalign magnification plainoutput
  eqalignoff magnificationoff plainoutputoff
  strictfootnotes altfootnotes footnotesoff
\end{verbatim}
\end{verbatim}

Note that most of these options come in \texttt{(option)/\texttt{(option)}off} pairs. These are particularly useful in conjunction with the special options (1.2) or to toggle the default options. By default, the options \texttt{diagnostics}, \texttt{tabbing} and \texttt{eqalign} are active (just what I tend to use...).
The actual options will be explained in section 2. Keep in mind, though, that some options affect others—for example selecting one of strictfootnotes and altfootnotes will automatically turn the other off; you can, however, disable both forms either with footnotesoff or by giving the two separate ...off commands separately. Also, plainoutput requires strictfootnotes and magnification, but plainoutputoff itself does not disable the Plain TeX footnote macros or magnification.

1.2 Special Options

Two options, called none and all, are available to allow maximum flexibility. These function because cjwplain calls the starred command \ProcessOptions* and therefore processes options in the order specified to \usepackage, and not the package’s internal declaration order. Thus, to make only Plain TeX’s tabbing commands available, one would use the call

\usepackage[none,tabbing]{cjwplain}

and to use everything while leaving Plain TeX’s \item command alone one would enter the command

\usepackage[all,strictitemoff]{cjwplain}

in the preamble.

2 The Code

2.1 Declarations

The options are implemented as \if statements, as that seemed to me to be the easiest way of including or excluding relatively large sections of code. First we allocate the \ifs.

1 \newif\if@outerallocs \@outerallocsfalse
2 \newif\if@diagnostics \@diagnosticstrue
3 \newif\if@plainskips \@plainskipsfalse
4 \newif\if@strictline \@strictlinetrue
5 \newif\if@strictitem \@strictitemfalse
6 \newif\if@tabbing \@tabbingtrue
7 \newif\if@eqalign \@eqaligntrue
8 \newif\if@strictfootnotes \@strictfootnotesfalse
9 \newif\if@altfootnotes \@altfootnotesfalse
10 \newif\if@plainoutput \@plainoutputfalse
11 \newif\if@magnification \@magnificationfalse

Now we declare how the options affect these \if tests.

13 \DeclareOption{outerallocs}{\@outerallocstrue}
14 \DeclareOption{outerallocsoff}{\@outerallocsfalse}
15 \DeclareOption{diagnostics}{\@diagnosticstrue}
16 \DeclareOption{diagnosticsoff}{\@diagnosticsfalse}
17 \DeclareOption{eqalign}{\@eqaligntrue}
18 \DeclareOption{strictfootnotes}{\@strictfootnotestrue}
19 \DeclareOption{altfootnotes}{\@altfootnotestrue}
20 \DeclareOption{plainoutput}{\@plainoutputtrue}
21 \DeclareOption{magnification}{\@magnificationtrue
We will have two possible ways of providing a \footnote command. As these are mutually exclusive, we make sure that they cannot both be true.

To use Plain \TeX's entire output routine will require that magnification code as well as Plain \TeX style footnotes be defined.

The two special options are given.

Finally we define a default option handling routine. I prefer only a warning as opposed to an error.
Now that all the options are declared, we process them in the order specified in the package call.

\ProcessOptions

### 2.2 Allocation Calls: outerallocs

Originally Plain \TeX{} had all allocation macros (\newcount, etc.) defined as \outer. \LaTeX{} redefines several of them to be non-outer. Careful consideration has failed to yield to me why these would need to be rewritten as \outer in this package—any Plain \TeX{} file which expects \outer definitions would not call them in a non-outer position, and any other files would themselves have redefined versions of the macros.

Since, however, it is such a small change, we will provide it. Note: Using the outerallocs option will break a good deal of standard \LaTeX{} code, namely the standard macros for counters and lengths. This means you probably do not want to use it. It is here only for completeness’s sake.

\if@outerallocs
\outer\def\newcount{\alloc@0\count\countdef\insc@unt}
\outer\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
\outer\def\newskip{\alloc@2\skip\skipdef\insc@unt}
\outer\def\newbox{\alloc@4\box\chardef\insc@unt}
\outer\def\newwrite{\alloc@7\write\chardef\sixt@@n}
\outer\def\newfam{\alloc@8\fam\chardef\sixt@@n}
\fi

### 2.3 Error Processing: diagnostics

Any Plain \TeX{} afficianados using this package will feel more comfortable to have the standard values for error processing information. One change, though. \LaTeX{}\epsilon uses a counter named errorcontextlines, and not a count.

\if@diagnostics
\showboxbreadth=5
\showboxdepth=3
\setcounter{errorcontextlines}{5}
\fi

### 2.4 Skips: plainskips

When the plainskips option is selected, the three \...skip macros should unconditionally leave horizontal mode and insert a skip, like in Plain \TeX{}.

\if@plainskips
\def\smallskip{\vskip\smallskipamount}
\def\medskip{\vskip\medskipamount}
\def\bigskip{\vskip\bigskipamount}
\fi
2.5 Fonts

A package already exists whereby oldstyle font commands can be given, namely \texttt{oldlfont}. Furthermore, one can use \texttt{rawfonts}, if necessary, to load in such specific fonts as \texttt{\ninebf}, etc.

\begin{verbatim}
\%\font\tenrm=cmr10 \% roman text
...
\%\textfont	tfam=\tentt
\end{verbatim}

2.6 The \texttt{\line} Macro: \texttt{strictline}

Now we get to the first tricky part. The \texttt{\line} macro needs to be available to the \texttt{picture} environment in \LaTeX{}, as well as restoring the original Plain \TeX{} definition for our usage here. The good news is that \LaTeX{} only uses \texttt{\line} inside of the \texttt{picture} environment. So we employ the following solution: we keep the definition of \texttt{\@line} as per \LaTeX{} convention, and give in any case a user accessible \texttt{\plainline}.

\begin{verbatim}
\let\plainline\@@line
\if@strictline
\%\def\@@line{\hbox to\hsize} % Defined in \texttt{ltplain.dtx}
\fi
\end{verbatim}

Now we define an internal name for the standard \LaTeX{} macro and restore the Plain \TeX{} definition.

\begin{verbatim}
\let\latex@line\line
\let\line\@@line
\end{verbatim}

The definitions of \texttt{\leftline}, \texttt{\rightline} and \texttt{\centerline} can be left as is (though users depending upon personal redefinitions of \texttt{\line} for special effects in these macros should simply put their redefinition into the macro \texttt{\@line}).

\begin{verbatim}
\%\def\leftline#1{\@@line{#1\hss}}
\%\def\rightline#1{\@@line{\hss#1}}
\%\def\centerline#1{\@@line{\hss#1\hss}}
\end{verbatim}

Now we make a patch to the definition of \texttt{\@picture} (the workhorse macro for the \texttt{picture} environment) which will restore the \LaTeX{} definition only within that environment.

\begin{verbatim}
\def\@picture(#1,#2)(#3,#4){% 
   \let\line\latex@line%
   \@picht#2\unitlength
   \setbox\@picbox\hbox to#1\unitlength\bgroup
   \hskip -#3\unitlength
   \lower #4\unitlength\hbox\bgroup
   \ignorespaces
\fi
\end{verbatim}

2.7 Tab Alignments: \texttt{tabbing}

The tabbing macros from Plain \TeX{} use the \texttt{newif} construction, so must occur at an \texttt{\outer} level. Thus, they are included in a separate package.
LaTEX may have its own tabbing environment, but I like Plain TeX’s. The only potential conflict I saw was with the `+` macro. However, LaTEX only defines `+` inside of the tabbing environment itself, so there should be absolutely no problem.

```latex
\def\cleartabs\settabs\tabalign \+ only potential conflict I saw was with the `+` macro. However, LaTEX only defines `+` inside of the tabbing environment itself, so there should be absolutely no problem.

\def\cleartabs\settabs\tabalign \+ only potential conflict I saw was with the `+` macro. However, LaTEX only defines `+` inside of the tabbing environment itself, so there should be absolutely no problem.
```

2.8 Itemising: strictitem

Now we have another problem, namely the `\item` macro. I unfortunately see no way to get around the fact that `\item` is a general macro in \LaTeX{}, and that the
formats are completely different: i.e., \texttt{plain} \TeX{} expects the \texttt{\langle label \rangle} to be the one mandatory argument, whereas \LaTeX{}'s \texttt{\item} macro takes the \texttt{\langle label \rangle} as an optional argument. Thus, the best I can think of is the following. We redefine \texttt{plain} \TeX{}'s \texttt{\item} after standard \LaTeX{} practice,

\begin{verbatim}
\def\@@item{\par\hang\textindent}
\let\plainitem\@@item
\end{verbatim}

and we \texttt{\let} it to something accessible in normal documents, the command \texttt{\plainitem}.

The command \texttt{\itemitem} can be taken care of directly.

\begin{verbatim}
\def\itemitem{\par\indent \hangindent2\parindent \textindent}
\end{verbatim}

Now a user will have to replace all occurrences of \texttt{\item{foo}} with either \texttt{\item[foo]} or \texttt{\plainitem{foo}} (I imagine the choice will depend upon one's editor's facilities with regexps...). It's not perfect, but it's the only way I can think of to provide maximum compatibility. Of course, we will still give the option, \texttt{strictitem}, of using just the original definition, but that will probably not be terribly convenient for anyone trying to add \LaTeX{} features on top of an existing \texttt{plain} \TeX{} source. Thus, we will also provide the (slightly longwinded) replacement \texttt{\latexitem}.

\begin{verbatim}
\if@strictitem
\let\latexitem\item
\let\item\@@item
\fi
\end{verbatim}

2.9 Miscellaneous

2.9.1 Sectioning

I have personally never used the \texttt{plain} \TeX{} \texttt{\beginsection} macro, but somebody might have...

\begin{verbatim}
\outer\def\beginsection#1\par{\vskip\z@ plus.3\vsize\penalty-250
\vskip\z@ plus-.3\vsize\bigskip\vskip\parskip
\message{#1}\leftline{\bf#1}\nobreak\smallskip\noindent}
\end{verbatim}

2.9.2 Proclamations

Once again we will leave \LaTeX{}'s NFSS based redefinition, this time for the \texttt{\proclaim} command, in place.

\begin{verbatim}
%\outer\def\proclaim #1. #2\par{\medbreak
%\noindent{\bfseries#1.\enspace}{\slshape#2\par}\
%\ifdim\lastskip<\medskipamount \removelastskip\penalty55\medskip\fi}
\end{verbatim}

2.9.3 Paragraph Formatting

I have done some simple tests of \LaTeX{}'s \texttt{\raggedright} macro, and it seems to me that it mimics the functionality of the \texttt{plain} \TeX{} macro of the same name. Therefore I see no reason to redefine it as part of this package.

\begin{verbatim}
%\def\raggedright{\%\rightskip\z@ plus2em \spaceskip.3333em \xspaceskip.5em\relax}
\end{verbatim}

Another \LaTeX{} font change will also be left as is.

\begin{verbatim}
%\def\ttraggedright{\reset@font\ttfamily\rightskip\z@ plus2em \relax}
\end{verbatim}
2.9.4 Accents and Miscellaneous

These should work as is for Plain \TeX documents.

Nor do I see a reason to change these back to Plain \TeX definitions.

The LaTeX definition of \ldots is more or less identical to the Plain \TeX macro \ldots. So we will leave this alone, too.

The \LaTeX definition of \ldots is more or less identical to the Plain \TeX macro \ldots. So we will leave this alone, too.

These changes, as others before, only add functionality without seeming to limit Plain \TeX usage, so no change will be made.

2.9.5 Ending the Document

We simply add the \bye macro back in, though the \end should be changed to the \LaTeX \end{document}.

2.9.6 Math Commands

Operators and other math-mode font-related changes will be ignored with as other NFSS alterations already mentioned.
Various \texttt{matrix} type command, including \texttt{bordermatrix} and \texttt{cases} have similarly been rewritten for NFSS commands under \LaTeX{}.

### 2.10 Math Alignment: eqalign

\begin{verbatim}
\def\eqalign#1{\null\,\vcenter{\openup\jot\m@th
\ialign{\strut\hfil$\displaystyle{##}$&$\displaystyle{{}##}$\hfil
\crcr#1\crcr}}\,}
\def\eqalignno#1{\displ@y \tabskip\@centering
\halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
&$\@lign\displaystyle{{}##}$\hfil\tabskip\@centering
&\llap{$\@lign##$}\tabskip\z@skip\crcr
#1\crcr}}
\def\leqalignno#1{\displ@y \tabskip\@centering
\halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
&$\@lign\displaystyle{{}##}$\hfil\tabskip\@centering
&\kern-\displaywidth\rlap{$\@lign##$}\tabskip\displaywidth\crcr
#1\crcr}}
\else
\let\plaincentering\@centering
\if@eqalign
\def\eqalignno#1{\displ@y \tabskip\@centering
\halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
&$\@lign\displaystyle{{}##}$\hfil\tabskip\@centering
&\llap{$\@lign##$}\tabskip\z@skip\crcr
#1\crcr}}
\else
\let\plainalignment#1{\displ@y \tabskip\@centering
\halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
&$\@lign\displaystyle{{}##}$\hfil\tabskip\@centering
&\kern-\displaywidth\rlap{$\@lign##$}\tabskip\displaywidth\crcr
#1\crcr}}
\fi
\fi
\end{verbatim}

\texttt{eqnarray*} If the user does not choose this option, we will instead define an \texttt{eqnarray*} environment which does not number equations.

\begin{verbatim}
\@namedef{eqnarray*}{% \let \ \cr $$\null\,\vcenter{\openup\jot\m@th
\ialign{\strut\hfil\displaystyle{##}$&\displaystyle{##}$\hfil
\crcr}}}
\@namedef{endeqnarray*}{\crcr}
\fi
\end{verbatim}

### 2.11 Output Routine: plainoutput

The output routines also involve \texttt{newif} commands, and are therefore also relegated to a separate package.

\begin{verbatim}
\ifplainoutput
\InputIfFileExists{cjwplout.clo}{}{% \PackageWarning{cjwplain}{Option ‘cjwplout.clo’ not found.}
\@tabbingfalse
\fi
\end{verbatim}

\texttt{headline} If the user wishes to use the entire \TeX{} output routine, we first redefine

\begin{verbatim}
\headline
\footline
\pageno
\folio
\end{verbatim}
the normal versions of headline, footline and pageno, as well as related macros. We will use NFSS definitions in place of \tenrm.

\countdef\pageno=0 % first page is number 1
\newtoks\headline \headline={\hfil} % headline is normally blank
\newtoks\footline \footline={\hss\reset@font\folio\hss} % footline is normally a centered page number in font \tenrm
\def\nopagenumbers{\footline{\hfil}} % blank out the footline
\def\folio{%
  \ifnum\pageno<\z@ \romannumeral-\pageno \else \number\pageno \fi}
\def\advancepageno{%\ifnum\pageno<\z@ \global\advance\pageno\m@ne \else \global\advance\pageno\@ne \fi} % increase \pageno

We also supply the \raggedbottom and \normalbottom macros as normal.

\newif\ifr@ggedbottom
\def\raggedbottom{\topskip 10\p@ plus60\p@ \r@ggedbottomtrue}
\def\normalbottom{\topskip 10\p@ \r@ggedbottomfalse}
% undoes \raggedbottom

If the entire output routine is being used, we define the Plain \TeX insertion macros as normal.

\newinsert\topins
\newif\ifp@ge \newif\if@mid
\def\topinsert{\@midfalse\p@gefalse\@ins}
\def\midinsert{\@midtrue\@ins}
\def\pageinsert{\@midfalse\p@getrue\@ins}
\skip\topins=\z@skip % no space added when a topinsert is present
\count\topins=1000 % magnification factor (1 to 1)
\dimen\topins=\maxdimen % no limit per page
\def\@ins{\par\begingroup\setbox\z@\vbox\bgroup} % start a \vbox
\def\endinsert{\egroup % finish the \vbox
  \if@mid \dimen@\ht\z@ \advance\dimen@\dp\z@ \advance\dimen@12\p@
  \ifdim\dimen@>\pagegoal\@midfalse\p@gefalse\fi\fi
  \if@mid \bigskip\box\z@igbreak
  \else \insert\topins\penalty100 % floating insertion
  \splittopskip\z@skip
  \splitmaxdepth\maxdimen \floatingpenalty\z@ \ifp@ge \dimen@\dp\z@ \else \vbox to\vsize{\unvbox\z@\kern-\dimen@}\% depth is zero \else \box\z@\nobreak\bigskip\fi\fi\endgroup}

Now we define the main part of the output routine. We use \@plainline instead of line, since \@line is guaranteed to have the definition we want.

\def\plainoutput{\shipout\vbox{\makeheadline\pagebody\makefootline} %
\advancepageno
\ifnum\outputpenalty>\-\@MM \else\dosupereject\fi}
\def\pagebody{\vbox to\vsize{\boxmaxdepth\maxdimen \pagecontents}}
\def\makeheadline{\vbox to\z@{\vskip-22.5\p@\@@line{\vbox to8.5\p@{}
\the\headline}\vss}
\nointerlineskip}
\def\makefootline{\baselineskip24\p@\@@line{\the\footline}}
\def\dosupereject{\ifnum\insertpenalties>\z@ % something is being held over
  \@plainline{\kern-\topskip\nobreak\vfill\supereject\fi}
\else \box\z@\nobreak\bigskip\fi\fi\endgroup}
Finally, we make the \texttt{\LaTeX} output routines active again.

2.11.1 Page Numbering, Running Heads and Miscellaneous

We can make the \texttt{\LaTeX} head and foot commands accessible (after a fashion), even if the entire output routine is not being used. \texttt{\LaTeXtwo} provides the commands \texttt{@oddhead} and \texttt{@evenhead}, as well as their footnote equivalents. Therefore we can give the following versions.

\begin{verbatim}
\if@plainoutput\else
\def\footline{\@ifnextchar ={\@@footline}{\@@footline=}}
\def\@@footline=#1{\gdef\@oddfoot{#1} \gdef\@evenfoot{#1}}
\def\headline{\@ifnextchar ={\@@headline}{\@@headline=}}
\def\@@headline=#1{\gdef\@oddhead{#1} \gdef\@evenhead{#1}}
\let\nopagenumbers\relax \let\@oddfoot\relax \let\@evenfoot\relax
\fi
\end{verbatim}

Since the \texttt{\LaTeX} \texttt{\headline} and \texttt{\footline} macros are actually tokens, we have to allow that the assignments to them can be made with an optional \texttt{=}.

The use of \texttt{@ifnextchar =} nicely takes care of that. This will alas be a slightly inefficient use of the macros, as many \texttt{\LaTeX} heads and feet already test for odd and even pages—but not all of them. So, we compromise.

Now we can simply make \texttt{\pageno} an equivalent for the \texttt{\LaTeX} counter \texttt{\c@page}.

\begin{verbatim}
\newcommand{\folio}{\if@oddfoot\relax \let\@oddfoot\relax \fi}
\end{verbatim}

This means that assignments can either be made in primitive fashion directly to \texttt{\pageno} or in \texttt{\LaTeX} fashion to the \texttt{\page} counter. We would also like to have the \texttt{\LaTeX} \texttt{\folio} macro. We are going to expand upon this slightly, though. I find \texttt{\folio} generally to be a useful command, so we will write it in such a way as to make it useful with \texttt{\LaTeX} counters (as with, say, the \texttt{\arabic} macro) as well as with counts.

\begin{verbatim}
\newcommand{\folio}{\if@oddfoot\relax \let\@oddfoot\relax \fi}
\end{verbatim}

This means that assignments can either be made in primitive fashion directly to \texttt{\folio} or in \texttt{\LaTeX} fashion to the \texttt{\page} counter. We would also like to have the \texttt{\LaTeX} \texttt{\folio} macro. We are going to expand upon this slightly, though. I find \texttt{\folio} generally to be a useful command, so we will write it in such a way as to make it useful with \texttt{\LaTeX} counters (as with, say, the \texttt{\arabic} macro) as well as with counts.
This is not, unfortunately, perfect, in that it must be used with an optional argument \texttt{\textbackslash folio\{section\}} as opposed to the normal style \texttt{\textbackslash arabic\{section\}}. On the other hand, I can’t think of many applications for \texttt{\textbackslash folio} other than page numbering. At any rate, we finish page numbering with the incrementation command.

```
def\advancepageno{\ifnum\pageno<0 \global\advance\pageno\m@ne
  \else\global\advance\pageno\@ne \fi}
```

One more bit from the Plain TeX output routine needs to be dealt with. Although LATEX’s \texttt{\textbackslash raggedbottom} macro will suffice to simulate Plain TeX’s command of the same name, we need to add a \texttt{\textbackslash let} command to enable Plain TeX’s counterpart, \texttt{\textbackslash normalbottom}.

```
\let\normalbottom\flushbottom
```

### 2.11.2 Insertions

If the Plain TeX output routine is not being used, we simulate the insertions using LATEX’s \texttt{\textbackslash figure} environment.

```
def\topinsert{\begin{figure}\[t\]}
def\pageinsert{\begin{figure}\[p\]}
def\midinsert{\begin{figure}\[htpb\]}
def\endinsert{\end{figure}}
```

### 2.11.3 Footnotes: strictfootnotes, altfootnotes

We define \texttt{\textbackslash@@footnote} as Plain TeX’s footnoting macro.

```
%\newinsert\footins
\let\latex@footnote\footnote
\def\@@footnote#1{\let\@sf\empty % parameter #2 (the text) is read later
  \ifhmode\edef\@sf{\spacefactor\the\spacefactor}/\fi
  #1\@sf\vfootnote{#1}}
\def\vfootnote#1{\insert\footins\bgroup
  \interlinepenalty\interfootnotelinepenalty
  \splittopskip\ht\strutbox % top baseline for broken footnotes
  \splitmaxdepth\dp\strutbox \floatingpenalty\@MM
  \leftskip\z@skip \rightskip\z@skip \spaceskip\z@skip
  \xspaceskip\z@skip
  \textindent{#1}\footstrut\futurelet\next\fo@t}
\def\fo@t{\ifcat\bgroup\noexpand\next \let\next\f@@t
  \else\let\next\f@t\fi \next}
\def\f@@t\bgroup\aftergroup\@foot\let\next}
\def\f@t#1{#1\@foot}
\def\@foot{\strut\textbackslash}\
```

LATEX leaves these initializations for the \texttt{\textbackslash footins} insert.

```
%\skip\footins=\bigskipamount % space added when footnote is present
%\count\footins=1000 % footnote magnification factor (1 to 1)
%\dimen\footins=8in  % maximum footnotes per page
```
Now we have several options for how to really deal with footnotes. The easy answer is to do them entirely according to Plain TeX.

The second option is to just use the \footnote command. This needs no rewriting, of course. The last option is to rewrite \footnote macro to use the Plain TeX format instead of the \footnote macro, which uses an optional argument.

LaTeX keeps Plain TeX’s \footnoterule as the default.

\footnoterule
% the \hrule is .4pt high

2.11.4 Magnification: magnification

The last part of Plain TeX for which we need to account is magnification. The magnification macros are easily reinstated, either as part of the overall Plain TeX output routine or standalone. Since the \mag primitive is not disabled, it could still be used in LaTeX. However, LaTeX does not itself work with true units any usage of \magnification could do some strange things to your page layouts.

This brings us to the end of the main package.