The LaTeX memoir class for configurable book typesetting: Source code*

Peter Wilson†
Herries Press
(with the assistance of Lars Madsen)

2010/04/19

Abstract

The memoir class is designed for typesetting general books such as novels, biographies, histories, and so on, although as it supports all the functionality of the standard book class it can also be used for technical writing. It provides more functions than the standard class as well as presenting a more friendly interface for the book designer. It can also simulate the typesetting style of the standard article class.

The class was first released in mid 2001 and has been well used ever since.

Contents

*This file (`memoir.dtx`) has version number v3.6, last revised 2010/04/19.

†herries dot press at earthlink dot net

List of Tables

1 Introduction

This document provides the commented source for the LaTeX memoir class, which is designed for typesetting general books such as novels, biographies, histories, and so on. It has all the functionality of the standard book class and, as well as providing some extra functions, also provides a more friendly interface for the document designer. As it can encompass everything that the book class provides it may also be used for technical writing.
The default appearance of a document typeset with this class is the same as if it had been typeset with the book class and it can be made to simulate the article class.. The class, though, includes extra facilities that make it easy to change the appearance of such things as the page headers and footers, the style of chapter and other sectional headings, and the style of captions. It also makes it easy to both change the style of the Table of Contents, List of Figures, etc., as well as creating new kinds of 'List of...'. New types of floats, if needed, can be created very simply. Epigraphs can be put into the document in a variety of styles.
The class provides a variety of page, chapter and captioning styles that you can choose from if you don't want to create your own.
Documents can be typeset in 9pt, 10pt, 11pt, 12pt, 14pt or 17pt font sizes, and if you have scaleable fonts available they can be set at any size. There is a reasonably intuitive means of setting the margins and placement of the text on a page. There is an option to put trim marks on the printed pages if the stock sheets need to be trimmed down to the final page size. For those whose publishers like a manuscript to look as though it was typewritten, there is an option to do this (double spacing, ragged right, no hyphenation, fixed width font). There is also an unsophisticated means of flagging any revisions to the text.
As this is a new class, by default it does not support the old LaTeX v2.09 font commands, namely the `\bf`, `\sl`, `\it` and `\sc` commands; it warns about using the `\em` command but does support it.
I hope that apart from the font commands the class is compatible with 'standard' LaTeX.
Development of this class would never have been started without the wonderful work done by Leslie Lamport and others [LMB99] from whom I have learned a great deal and borrowed much code.
Sections 2 through 4 describe some administrative elements and code for general use later in the specification. The macros forming the class file are defined in sections 5 through 32.
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 A driver for this document

The next series of code contains the documentation driver file for LaTeX, i.e., the file that will produce the documentation you are currently reading. This will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass[twoside]{ltxdoc}
\usepackage[T1]{fontenc}
\usepackage{url}
%%\usepackage[draft=false,
%%            plainpages=false,
%%            pdfpagelabels,
%%            bookmarksnumbered,
%%            hyperindex=true
%%           ]{hyperref}  % Doesn't work with indexing of \DescribeMacro
\providecommand{\phantomsection}{} % just in case hyperref not used
```

We do want an index, using linenumbers, but not update information.

```
\EnableCrossrefs
\CodelineIndex
%% \RecordChanges
```

We had better have page headings to aid navigation, but I don't like Uppercased titles.

```
\makeatletter
  \@mparswitchfalse
\makeatother
\renewcommand{\MakeUppercase}[1]{#1}
\pagestyle{headings}
```

We may use so many `docstrip` modules that we set the `StandardModuleDepth` counter to 1.

```
\setcounter{StandardModuleDepth}{1}
```

Some commonly used abbreviations

```
\newcommand*{\Lopt}[1]{\textsf {#1}}            % typeset an option
\newcommand*{\file}[1]{\texttt {#1}}            % typeset a file
\newcommand*{\Lcount}[1]{\textsl {\small#1}}    % typeset a counter
\newcommand*{\pstyle}[1]{\textsl {#1}}          % typeset a pagestyle
\newcommand*{\Lenv}[1]{\texttt {#1}}            % typeset an environment
\newcommand*{\Lpack}[1]{\textsf {#1}}           % typeset a package
\newcommand*{\ctt}{\textsc{ctt}}                % comp.text.tex
\newenvironment{PW}{\em}{}
\newcommand*{\theTeXbook}{\textit{The \TeX book}}
```

We want the full details printed.

```
\begin{document}
\DeleteShortVerb{\|}
\raggedbottom
\raggedright
\DocInput{memoir.dtx}
```

```
\PrintIndex
%% \PrintChanges
\end{document}
⟨/driver⟩
```

3 Identification

The memoir document class can only be used with LaTeX2e, so we make sure that an appropriate message is displayed when another TEX format is used.

```
⟨class⟩\NeedsTeXFormat{LaTeX2e}
```

Announce the name, option files and version for LaTeX2e files:

```
⟨class⟩\ProvidesClass{memoir}%
⟨class⟩  [2010/04/19 v3.6 configurable book, report, article document class]
⟨9pt⟩\ProvidesFile{mem9.clo}%
⟨9pt⟩              [2008/01/30 v0.4 memoir class 9pt size option]
⟨10pt⟩\ProvidesFile{mem10.clo}%
⟨10pt⟩              [2008/01/30 v0.3 memoir class 10pt size option]
⟨11pt⟩\ProvidesFile{mem11.clo}%
⟨11pt⟩              [2008/01/30 v0.3 memoir class 11pt size option]
⟨12pt⟩\ProvidesFile{mem12.clo}%
⟨12pt⟩              [2008/01/30 v0.4 memoir class 12pt size option]
⟨14pt⟩\ProvidesFile{mem14.clo}%
⟨14pt⟩              [2008/01/30 v0.4 memoir class 14pt size option]
⟨17pt⟩\ProvidesFile{mem17.clo}%
⟨17pt⟩              [2008/01/30 v0.3 memoir class 17pt size option]
⟨20pt⟩\ProvidesFile{mem20.clo}%
⟨20pt⟩              [2008/01/31 v0.1 memoir class 20pt size option]
⟨25pt⟩\ProvidesFile{mem25.clo}%
⟨25pt⟩              [2008/01/31 v0.1 memoir class 25pt size option]
⟨30pt⟩\ProvidesFile{mem30.clo}%
⟨30pt⟩              [2008/01/31 v0.1 memoir class 30pt size option]
⟨36pt⟩\ProvidesFile{mem36.clo}%
⟨36pt⟩              [2008/01/31 v0.1 memoir class 36pt size option]
⟨48pt⟩\ProvidesFile{mem48.clo}%
⟨48pt⟩              [2008/01/31 v0.1 memoir class 48pt size option]
⟨60pt⟩\ProvidesFile{mem60.clo}%
⟨60pt⟩              [2008/01/31 v0.1 memoir class 60pt size option]
```

4 Initial Code

```
⟨*class⟩
```

Note (2001/08/03): Old versions of the amsmath package did odd things with `\@tempa`, `\@tempb` and `\@tempc`. I have now replaced any use of these with `\@memtempa`, etc.
In this part we define a few commands that are used later on.

\@ptsize
\@memptsize

The \@ptsize control sequence is normally used to store the second digit of the pointsize we are typesetting in. So, normally, it's value is one of 0, 1 or 2. \@memptsize stores the full pointsize.

```
\newcommand*{\@ptsize}{}
\newcommand*{\@memptsize}{}
```

Any new lengths that depend on the point size option must be declared before the options are executed.

\onelineskip
\lxvchars
\xlvchars

The length \onelineskip is the vertical space taken by a normal line of text. The lengths \lxvchars and \xlvchars are the approximate lengths required for typesetting lines with either 65 or 45 characters.

```
\newlength{\onelineskip}
\newlength{\lxvchars}
\newlength{\xlvchars}
```

\@memcnta
\c@memmarkcntra

We need a scratch count register and a scratch counter.

```
\newcount\@memcnta
\newcounter{@memmarkcntra}
```

\if@restonecol

When the document has to be printed in two columns, we sometimes have to temporarily switch to one column. This switch is used to remember to switch back.

```
\newif\if@restonecol
```

\if@openright

This is TRUE if chapters are to start on righthand (recto) pages; this is the default. FALSE means chapters can start on any page.

```
\newif\if@openright
  \@openrighttrue

```

\if@openleft

This is TRUE if chapters are to start on lefthand (verso) pages.

```
\newif\if@openleft
  \@openleftfalse

```

\if@mainmatter

This is TRUE if the main part of the document is being currently procesed; this is the default.

```
\newif\if@mainmatter
  \@mainmattertrue

```

\if@memoldfont

This is TRUE if the oldfontcommands option is used.

```
\newif\if@memoldfont
  \@memoldfontfalse

```

\ifextrafontsizes This is TRUE if the extrafontsizes option is used.

```
\newif\ifextrafontsizes
  \extrafontsizesfalse

```

\@memerror \@memwarn Two macros to save some space when reporting errors or warnings. The macros take the same arguments, ignoring the first in each case, as \ClassError and \ClassWarning, e.g.,
\@memwarn{Message}
instead of
\ClassWarning{memoir}{Message}.

```
\newcommand*{\@memerror}{\ClassError{memoir}}
\newcommand*{\@memwarn}{\ClassWarning{memoir}}

```

\ifsamename \nametest The macro \nametest{⟨*name1*⟩}{⟨*name2*⟩} tests whether the characters forming the two arguments are the same or not. If they are the same then \ifsamename is set TRUE, otherwise it is set FALSE. ⟨*name1*⟩ and ⟨*name2*⟩ must both be either macro names (including the backslash) or must both not be macro names. That is, you cannot do \nametest{\cs}{cs}.

```
\newif\ifsamename
\newcommand{\nametest}[2]{%
  \samenamefalse%
  \begingroup%
  \def\@memtempa{#1}\def\@memtempb{#2}%
  \ifx \@memtempa\@memtempb%
    \endgroup%
    \samenametrue%
  \else%
    \endgroup%
  \fi}

```

\ifm@m@And \m@m@Andtrue \m@m@Andfalse \ifm@m@Or \m@m@Ortrue \m@m@Orfalse \ifm@m@Xor \m@m@Xortrue \m@m@Xorfalse These are for 'if A and B', 'if A or B' and 'if A xor B'.
For A and B:

```
\m@m@Andfalse
\ifA
  \ifB
    \m@mAndtrue
  \fi
\fi
```

For A or B:

```
\m@m@Ortrue
\ifA
\else
  \ifB
```

```
  \else
    \m@m@Orfalse
  \fi
\fi
```

For A xor B:

```
\m@m@Xortrue
\ifA
  \ifB
    \m@m@Xfalse
  \fi
\else
  \ifB
  \else
    \m@m@Xorfalse
  \fi
\fi
```

```
\newif\ifm@m@And
\newif\ifm@m@Or
\newif\ifm@m@Xor
```

`\kill@lastcounter` `\kill@lastcounter{`⟨*cntr*⟩`}`[1] kills the counter ⟨*cntr*⟩. This macro is an extension to the suggestions as it also makes the last count register reusable. *The macro must only be used when the last allocated counter is to be killed.* For example with:
`\newcounter{fred} \kill@lastcounter{fred} \newcounter{fred}`
there is no error with the second `\newcounter`.

```
\newcommand{\kill@lastcounter}[1]{%
```

Deallocate the last counter register. From *The TEXbook*, `\count10` is the number of the last register. The counter should be zeroed before being deallocated otherwise an immediatally following `\newcount` may not be zero when allocated[2].

```
  \count\count10 \z@
  \advance\count10 \m@ne
```

Set the internal counter definition to `\relax`.

```
  \expandafter\let\csname c@#1\endcsname\relax}
```

`\@name@p@xdf` `\@name@unresp@xdef` `\@namelet` `\@namelongdef` Utility definition macros, along the lines of the kernel's `\@namedef`.

```
\newcommand{\@name@p@xdef}[1]{%
  \expandafter\protected@xdef\csname #1\endcsname}
```

[1] With thanks to Stefan Ulrich (`ulrich@cis.uni-muenchen.de`) who answered a question on this on CTT, 2001/07/09. (Also answered by Heiko Oberdiek, `oberdiek@ruf.uni-freiburg.de`).

[2] Discovered and fix provided by Robert Schlicht on 2005/08/31.

```
\newcommand{\@name@unresp@xdef}[1]{%
  \expandafter\unrestored@protected@xdef\csname #1\endcsname}
\newcommand{\@namelet}[1]{%
  \expandafter\let\csname #1\endcsname}
\newcommand{\@namelongdef}[1]{%
  \long\expandafter\def\csname #1\endcsname}
```

\memletcmdtxt
\memlettxttxt
\memlettxtcmd

More utility macros (for the user but perhaps better not to tell them about it in the manual)

```
\memletcmdtxt{\cmd}{txt}  -> \let\cmd\txt
\memlettxttxt{txta}{txtb} -> \let\txta\txtb
\memlettxtcmd{txt}{\cmd}  -> \let\txt\cmd
```

```
\newcommand*{\memletcmdtxt}[2]{\expandafter\let\expandafter#1\csname#2\endcsname}
\newcommand*{\memlettxttxt}[1]{\expandafter\memletcmdtxt\csname#1\endcsname}
\newcommand*{\memlettxtcmd}[2]{\expandafter\let\csname#1\endcsname#2}
```

\@nameedef

A shorthand for using \protected@edef.

```
\newcommand{\@nameedef}[1]{%
  \expandafter\protected@edef\csname #1\endcsname}

```

\memjustarg
\memgobble

Utility macros *that are not to be changed by any user!*

```
\newcommand{\memjustarg}[1]{#1}
\newcommand{\memgobble}[1]{}

```

4.1 Classes, Packages and Files

\@memfakeusepackage

\@memfakeusepackage{⟨*pack*⟩} fools LaTeX into thinking that the ⟨*pack.sty*⟩ package has been loaded.

```
\newcommand*{\@memfakeusepackage}[1]{%
  \@namelet{ver@#1.sty}\@empty}

```

\EmulatedPackage
\EmulatedPackageWithOptions

There was some discussion on CTT[3] about how to prevent loading of a package. My response was to use \@memfakusepackage. Donald Arseneau came up with this comprehensive solution (there was a little discussion about naming with my final suggestion as here).

\EmulatedPackage{⟨*package*⟩}[⟨*date*⟩]
\EmulatedPackageWithOptions{⟨*option-list*⟩}{⟨*package*⟩}[⟨*date*⟩]

```
\providecommand*{\EmulatedPackage}{}
\renewcommand*{\EmulatedPackage}[1]{%
  \@ifnextchar[{\@emulated@package{#1}}%
               {\@emulated@package{#1}[\@empty]}%]
}
\providecommand*{\EmulatedPackageWithOptions}{}
```

[3] *Re: breakurl + pdfetex for generating .dvi*, March 2005.

```
\renewcommand*{\EmulatedPackageWithOptions}[2]{%
  \PassOptionsToPackage{#1}{#2}%
  \EmulatedPackage{#2}%
}
```

`\@emulated@package` The workhorse for the user commands.

```
\def\@emulated@package#1[#2]{%
  \expandafter\xdef\csname ver@#1.\@pkgextension\endcsname{#2}%
  \@ifundefined{opt@#1.\@pkgextension}%
    {\@namedef{opt@#1.\@pkgextension}{}}{}%
  \wlog{Package #1 \ifx\@empty#2\else[#2] \fi
        \if,\csname opt@#1.\@pkgextension\endcsname,\else
        (with options \csname opt@#1.\@pkgextension\endcsname) \fi
        emulated by \@currname.}%
}
\@onlypreamble\EmulatedPackage
\@onlypreamble\EmulatedPackageWithOptions
\@onlypreamble\@emulated@package
```

`\DisemulatePackage` `\DisemulatePackage{`⟨*package*⟩`}` undoes a previous `\EmulatedPackage...`

```
\newcommand*{\DisemulatePackage}[1]{%
  \@namelet{ver@#1.\@pkgextension}\relax}
\@onlypreamble\DisemulatePackage
```

Code, based on a hint from Morten Høgholm[4] that the scrlfile package from koma-script includes commands such as
`\AfterFile{`⟨*file*⟩`}{`⟨*code*⟩`}`
plus before and after classes and packages.
Extend the `\InputIfFileExists` macro to call hooks before and the actual inputting. The kernel version (as of 2005/11/21) is:

```
\newcommand{\InputIfFileExists}[2]{%
 \IfFileExists{#1}%
   {\#2\@addtofilelist{#1}\@@input \@filef@und}}
```

`\InputIfFileExists` Effectively all file input is handled by this macro. Extend it by adding pre- and post- input hooks.

```
\renewcommand{\InputIfFileExists}[2]{%
  \IfFileExists{#1}%
    {#2\@addtofilelist{#1}\m@matbeginf{#1}%
     \@@input \@filef@und
     \m@matendf{#1}%
     \killm@matf{#1}}}
```

[4] CTT posting 2005/11/08, *Re: Checking for packages froma class... revisited*

\m@matbeginf
\m@matendf

\m@matbeginf{⟨file⟩} calls macro \file-m@mfb if it is defined. Similarly \m@matendf{⟨file⟩} calls \file-m@mfe if it is defined.

```
\newcommand{\m@matbeginf}[1]{\@ifundefined{#1-m@mfb}{}%
  {\@nameuse{#1-m@mfb}}}
\newcommand{\m@matendf}[1]{\@ifundefined{#1-m@mfe}{}%
  {\@nameuse{#1-m@mfe}}}
```

\killm@matf

\killm@matf{⟨file⟩} undefines the \file-m@mfb and \file-m@mfe macros.

```
\newcommand*{\killm@matf}[1]{%
  \@namelet{#1-m@mfb}\relax
  \@namelet{#1-m@mfe}\relax}
```

\AtBeginFile
\AtEndFile

\AtBeginFile{⟨file⟩}{⟨code⟩} inserts ⟨code⟩ just before ⟨file⟩ is input. Similarly \AtEndFile inserts just after input.

```
\newcommand{\AtBeginFile}[2]{\@ifundefined{#1-m@mfb}%
  {\@namedef{#1-m@mfb}{#2}}%
  {\expandafter\addtodef\csname #1-m@mfb\endcsname{}{#2}}}
\newcommand{\AtEndFile}[2]{\@ifundefined{#1-m@mfe}%
  {\@namedef{#1-m@mfe}{#2}}%
  {\expandafter\addtodef\csname #1-m@mfe\endcsname{}{#2}}}
```

\AtBeginPackage
\AtEndPackage
\RequireAtEndPackage

\AtBeginPackage{⟨pack⟩}{⟨code⟩} inserts ⟨code⟩ just before the ⟨pack⟩ package is input, and \AtEndPackage is the equivalent for after input. \RequireAtEndPackage inserts ⟨code⟩ either at the end of ⟨pack⟩, or immediately if ⟨pack⟩ has already been input.

```
\newcommand{\AtBeginPackage}[2]{%
  \AtBeginFile{#1.\@pkgextension}{#2}}
\newcommand{\AtEndPackage}[2]{%
  \AtEndFile{#1.\@pkgextension}{#2}}
\newcommand{\RequireAtEndPackage}[2]{%
  \@ifpackageloaded{#1}{#2}%
  {\AtEndFile{#1.\@pkgextension}{#2}}}
```

\AtBeginClass
\AtEndClass
\RequireAtEndClass

\AtBeginClass{⟨class⟩}{⟨code⟩} and friends are the class equivalents of \AtBeginPackage and friends.

```
\newcommand{\AtBeginClass}[2]{%
  \AtBeginFile{#1.\@clsextension}{#2}}
\newcommand{\AtEndClass}[2]{%
  \AtEndFile{#1.\@clsextension}{#2}}
\newcommand{\RequireAtEndClass}[2]{%
  \@ifclassloaded{#1}{#2}%
  {\AtEndFile{#1.\@clsextension}{#2}}}
```

\phantomsection A command needed if the hyperref package is used, for putting before certain \addcontentsline commands.

```
\newcommand{\phantomsection}{}
```

\nofiles The kernel's \nofiles, which surprisingly may be called *before* the class is loaded, lets \makeindex to \relax. This could cause problems[5] with my initial version of \makeindex. Here's the kernel \nofiles definition:

```
\def\nofiles{%
  \@fileswfalse%  flag for suppressing \immediate \writes
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3%
    {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
  \let\makeindex\relax
  \let\makeglossary\relax}
```

To get it to work for when \makeindexes optional argument is used I let \makeindex itself deal with \nofiles (and also \makeglossary).

```
\renewcommand*{\nofiles}{%
  \@fileswfalse%  flag for suppressing \immediate \writes
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3%
    {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
  }
```

\memsetcounter A wrapper round \setcounter. I want to use this in the aux file.

```
\newcommand*{\memsetcounter}[2]{\setcounter{#1}{#2}}
\AtBeginDocument{\immediate\write\@mainaux{%
  \string\providecommand*{\string\memsetcounter}[2]{}}}
```

4.2 For package documentation

Some macros that may be useful for documenting LaTeX code. These have principally come from doc.dtx.

\bs Prints \

```
\def\bs{\texttt{\char`\\}}
```

\l@nohyphenation From doc.dtx

\meta \meta{⟨*arg*⟩} prints ⟨*arg*⟩.

\meta@font@select

```
\ifx\l@nohyphenation\undefined
  \newlanguage\l@nohyphenation
\fi
```

[5] As reported by Heiko Oberdiek on 2005/07/09.

```
\DeclareRobustCommand{\meta}[1]{%
  \ensuremath\langle
  \ifmmode \expandafter \nfss@text \fi
  {%
    \meta@font@select
    \edef\meta@hyphen@restore
      {\hyphenchar\the\font\the\hyphenchar\font}%
    \hyphenchar\font\m@ne
    \language\l@nohyphenation
    #1\/%
    \meta@hyphen@restore
  }\ensuremath\rangle
}
\def\meta@font@select{\itshape}

```

\marg
\oarg
\parg

Robust versions of the `doc.dtx` macros.
`\marg{`⟨*arg*⟩`}` prints {⟨*arg*⟩}
`\oarg{`⟨*arg*⟩`}` prints [⟨*arg*⟩]
`\parg{`⟨*arg*⟩`}` prints (⟨*arg*⟩)

```
\DeclareRobustCommand{\marg}[1]{%
  {\ttfamily\char`\{}\meta{#1}{\ttfamily\char`\}}}
\DeclareRobustCommand{\oarg}[1]{%
  {\ttfamily\char`\[}\meta{#1}{\ttfamily\char`\]}}
\DeclareRobustCommand{\parg}[1]{%
  {\ttfamily\char`\(}\meta{#1}{\ttfamily\char`\)}}

```

\cs

`\cs{arg}` prints \arg.

```
\DeclareRobustCommand{\cs}[1]{\texttt{\char`\\#1}}

```

\cmdprint
\cmd

From Heiko Oberdiek CTT 2001/05/26 (print and index a command)
`\cmdprint{\fred}` prints \fred
`\cmd{\fred}` prints and indexes \fred. NOTE It assumes that `?` is the 'actual' character for MakeIndex (it is normally `@` but that is not much use if a command includes `@` as part of its name).

```
\newcommand{\cmdprint}[1]{\texttt{\string#1}}
\newcommand{\cmd}[1]{\cmdprint{#1}%
  \index{\expandafter\@gobble\string#1?\string\cmdprint{\string#1}}}

```

4.3 Checking the processor

Prior to 2008/07/22 the hyperref package was modified in such a way as to produce warnings about versions of the ifpdf and ifxetex packages when hyperref was used with versions v1.618033, v1.61803, or v1.618 mempatch v4.9 of the class. hyperref's author refused to make any changes to his package, insisting that emulating a package was 'ugly hacking'. He did, though, provide an example code that could be added to memoir to use a package if it was available.

I have used it for the emulations in this section. There will be no further changes to the class to accomodate any further changes to hyperref — it's easy for a package to determine what class is being used and make arrangements accordingly, especially as there are not all that many classes. There are hundreds of packages that might be used with a particular class and it would be senseless for a class to attempt to take account of everything that might be used with it.

\ifpdf
\pdftrue
\pdffalse

This can be used to check whether or not a document is being processed by LaTeX or pdfLaTeX.

```
\newif\ifm@mifpdf
  \m@mifpdffalse
\IfFileExists{ifpdf.sty}{\RequirePackage{ifpdf}\relax}{%
  \ClassWarningNoLine{memoir}{%
    Package 'ifpdf' is not installed.\MessageBreak
    The package is being emulated}%
\m@mifpdftrue
\expandafter\newif\csname ifpdf\endcsname
  \pdffalse
\ifx\pdfoutput\undefined
\else
  \ifx\pdfoutput\@undefined
  \else
    \ifx\pdfoutput\relax
    \else
      \ifnum\pdfoutput>0\relax
        \pdftrue
      \fi
    \fi
  \fi
\fi
%%\EmulatedPackage{ifpdf}[2008/07/23]
}

```

\ifetex
\etextrue
\etexfalse

Check if etex is being used. This is based on the check for pdf(latex). Someone may have created a package for this, or perhaps not.

```
\newif\ifm@mifetex
  \m@mifetexfalse
\IfFileExists{ifetex.sty}{\RequirePackage{ifetex}\relax}{%
  \ClassInfo{memoir}{%
    An 'ifetex' package is being emulated}%
\m@mifetextrue
\newif\ifetex
  \etexfalse
\ifx\eTeXversion\undefined
\else
  \ifx\eTeXversion\@undefined
  \else
    \ifx\eTeXversion\relax
```

```
    \else
      \ifnum\eTeXversion>0\relax
        \etextrue
      \fi
    \fi
  \fi
\fi
%%\EmulatedPackage{ifetex}[2008/07/23]
}

```

Having supplied an `\ifpdf` to check if pdfLaTeX is being used, here's one for XeTeX.

`\ifxetex` Checks if XeTeX is being used.
`\xetextrue`
`\xetexfalse`
`\RequireXeTeX`

```
\newif\ifm@mifxetex
  \m@mifxetexfalse
\IfFileExists{ifxetex.sty}{\RequirePackage{ifxetex}\relax}{%
  \ClassWarningNoLine{memoir}{%
    The 'ifxetex' package is not installed.\MessageBreak
    The package is being emulated}%
\expandafter\newif\csname ifxetex\endcsname
\@ifundefined{XeTeXrevision}{\xetexfalse}{\xetextrue}
\m@mifxetextrue%
```

Check for XeTeX from the ifxetex package.

```
%%\EmulatedPackage{ifxetex}[2008/07/23]
}
\ifm@mifxetex%
% ifxetex package not found, emulate \RequireXeTeX
\def\RequireXeTeX{%
  \ifxetex\else
  \@memerror{XeTeX is required to process this document}%
             {Try again with xelatex, not (pdf)latex.\MessageBreak
              Or try removing any XeTeX package(s).}
  \fi}
\fi

```

`\ifluatex` Check if `luatex` is being used. This is based on the check for `pdf(latex)`.
`\luatextrue` Someone may have created a package for this, or perhaps not.
`\luatexfalse`

```
\newif\ifm@mifluatex
  \m@mifluatexfalse
\IfFileExists{ifluatex.sty}{\RequirePackage{ifluatex}\relax}{%
  \ClassWarningNoLine{memoir}{%
    The 'ifluatex' package is not installed.\MessageBreak
    The package is being emulated}%
\m@mifluatextrue
\expandafter\newif\csname ifluatex\endcsname
  \luatexfalse
```

```
\ifx\luatexversion\@undefined
\else
  \ifx\luatexversion\undefined
  \else
    \ifx\luatexversion\relax
    \else
      \luatextrue
    \fi
  \fi
\fi
%%\EmulatedPackage{ifluatex}[2008/07/23]
}

```

4.4 Extending an existing macro

My usual technique for modifying the code of an existing macro was to use `\renewcommand`. However it slowly became apparent that many of such modifications merely consisted of adding some code at either the start or end of the existing definition. In June 2000 I posed a question on the `comp.text.tex` newsgroup about this, and Michael Downes[6] and Heiko Oberdiek[7] were kind enough to respond. The responses arrived almost simultaneously, both supplying methods for adding code at the end of a macro defined to take one argument. Michael Downes' response included a remark about it probably being too much effort to formulate a general way of doing this. Heiko Oberdiek almost immediately replied with such a generalisation.
The following code, which I have extracted from a package that I wrote but which I did not submit to CTAN, incorporates the work of both Michael and Heiko. I have added a bit. All errors are my responsibility. Michael went on the write the patchcmd package which is a generalisation of the facilities below and so there was no need for my more restricted package.

`\wo@dmacro` Heiko Oberdiek suggested[8] that ⟨*macro*⟩ should be tested for being the name of a macro, rather than using the normal LaTeX test against it being undefined. He supplied the code for performing the macro name test.
`\wo@dmacro` is a helper for handling the string 'macro:'.

```
\edef\wo@dmacro{%
  \string m\string a\string c\string r\string o\string :%
}

```

`\wo@difmacro@begingroup` This seems to take two arguments:
`\wo@difmacro@begingroup{`⟨*macro*⟩`}{`⟨*code*⟩`}`, where ⟨*macro*⟩ should be the name of a defined macro and ⟨*code*⟩ is code to be executed if and only if ⟨*macro*⟩ *is* a defined macro.

[6] `epsmjd@ams.org`
[7] `oberdiek@ruf.uni-freiburg.de`
[8] Via email on 2000/07/12.

```
\def\wo@difmacro@begingroup#1{%
```

Start a group.

```
  \begingroup
```

Test if `#1` is a defined macro; `\wo@dparsemacro` does most of the work.

```
    \edef\x{%
      \noexpand\wo@dparsemacro\meaning#1\wo@dmacro\string -%
    }%
    \x\@nil{#1}%
}
```

`\wo@dparsemacro` This is called by `\wo@difmacro@begingroup`. If ⟨*macro*⟩ is not a defined macro it prints a warning and closes the group begun by `\wo@difmacro@begingroup`. Otherwise it process the ⟨*code*⟩ which *must* close the group.

```
\begingroup
  \edef\x{\endgroup
    \def\noexpand\wo@dparsemacro##1\wo@dmacro##2\string -}%
\x#3\@nil#4{%
  \ifx\\#3\\%
    \endgroup
    \@memwarn{\string `\string #4\string ' is not a macro}%
    \expandafter\@gobble
  \else
    \expandafter\@firstofone
  \fi
}
```

`\addtodef` `\addtodef*` `\wo@daddtodef` The command `\addtodef{`⟨*macro*⟩`}{`⟨*start-stuff*⟩`}{`⟨*end-stuff*⟩`}` adds stuff at the start and/or end of an argumentless macro. Initially the code for appending was a reimplementation of the kernel `\g@addto@macro` command (from `ltclass.dtx`). Later, at Michael Downes' suggestion, I combined appending and prepending stuff into a single command. Actually, there is no real need to have both `\addtodef` and `\addtodef*` as the body of the command being amended has no argument, but both versions are provided for consistency.
The kernel `\@star@or@long` and `\l@ngrel@x` commands (in `ltdefns.dtx`) are used to handle the potential `*` after the command name.

```
\def\addtodef{\@star@or@long\wo@daddtodef}
```

The `\@star@or@long` command dealt with a possible `*` and now `\wo@daddtodef` does the work. It picks up the three arguments that the user thinks belong to `\addtodef`, namely ⟨*macro*⟩, ⟨*start-stuff*⟩ and ⟨*end-stuff*⟩.

```
\long\def\wo@daddtodef#1#2#3{%
```

Check if ⟨*macro*⟩ has been defined.

```
  \wo@difmacro@begingroup{#1}{%
```

If ⟨*macro*⟩ is defined then store the tokens corresponding to the body of ⟨*macro*⟩ and the extra ⟨*-stuff*⟩ in token registers.

```
    \@temptokena{#2}%
    \toks@\expandafter{#1#3}%
```

Do an expanded definition for `\x`, so that calling `\x` will `\def` a new version of ⟨*macro*⟩, whose body consists of the saved tokens (i.e., the orginal body plus the extra stuff). As Michael Downes noted, single letter control sequences do not take any of TEX's hash table space. `\l@ngrel@x` has been previously set to either `\long` or `\relax` by `\@star@or@long`, and so may make the new definition of ⟨*macro*⟩ to be `\long`.

```
    \edef\x{\endgroup
        \l@ngrel@x\def\noexpand#1{\the\@temptokena \the\toks@}}%
```

Finally, call `\x` to perform the new definition.

```
    \x
  }%
}
```

`\addtoiargdef`
`\addtoiargdef*`
`\wo@daddtoiargdef`

This adds stuff at the start and/or end of a macro that takes one argument. It is a modification of Michael Downes' appending code[9] which was:

```
\def\appendef#1#2{%
  \begingroup
    \toks@\expandafter{#1{##1}#2}%
    \edef\x{\endgroup \def\noexpand#1####1{\the\toks@}}%
  \x}
```

```
\def\addtoiargdef{\@star@or@long\wo@daddtoiargdef}
\long\def\wo@daddtoiargdef#1#2#3{%
  \wo@difmacro@begingroup{#1}{%
    \@temptokena{#2}%
    \toks@\expandafter{#1{##1}#3}%
    \edef\x{\endgroup
       \l@ngrel@x\def\noexpand#1####1{\the\@temptokena \the\toks@}}%
    \x
  }%
}
```

For the record, Heiko Oberdiek produced the following version[10] which avoids the use of a token register.

```
\newcommand{\appendiargdef}[2]{%
  \long\expandafter\def\expandafter#1\expandafter
```

[9] Posted to CTT on 15 June 2000,
URL: `http://www.dejanews.com/getdoc.xp?AN=635057844`.

[10] Posted to CTT on 16 June 2000,
URL: `http://www.dejanews.com/getdoc.xp?AN=635095381`.

```
##\expandafter1\expandafter{#1{##1}#2}}
```

Continuing the saga, Michael came up with a pretty general solution for modifying a macro with any number (up to 9) of arguments which he sent to me on 2000/07/13; this became the patchcmd package which is on CTAN. I'm sure that Michael would have given me permission to include it here if he had not tragically passed away in 2003. I miss his encouragement, advice, and skills.

\patchcmd `\patchcmd{⟨macro⟩}{⟨start-stuff⟩}{⟨end-stuff⟩}` inserts ⟨*start-stuff*⟩ at the beginning of the definition of ⟨*macro*⟩ and ⟨*end-stuff*⟩ at the end of the definition. ⟨*macro*⟩ can have 0–9 arguments and be defined by \newcommand (and associates) or with \DeclareRobustCommand. For macros that use \futurelet (e.g., those with starred forms or optional arguments) only prepending works — any non-empty ⟨*end-stuff*⟩ will mess things up. \patchcmd does not work with macros that have delimited arguments.

```
%%%%%%%%%%%%%%% Michael Downes' patchcmd 2000/07/31 v1.03 %%%%%%%%
\newcommand{\patchcommand}[1]{%
  \expandafter\patchcmd@a\meaning#1??->@\@nil#1%
}
```

\patchcmd@a

```
\long\def\patchcmd@a#1#2#3->#4#5\@nil#6{%
  \ifx @#4\relax \patchcmdError#6#1%
    \expandafter\@gobbletwo % discard the other two arguments
  \else
    \if l#2\toks@{\patchcmd@e{}#6}% l in this position means \long
    \else \toks@{\patchcmd@e*#6}%   not \long
    \fi
    \patchcmd@b #3@#4#5 ? ? ? \@nil#6%
    \expandafter\the\expandafter\toks@
  \fi}
```

\patchcmd@b

```
\def\patchcmd@b#1:#2@#3#4 #5#6 #7 #8\@nil#9{%
  \if \ifx @#7@\expandafter
      \ifx\csname #6\endcsname#9T\else F\fi\else F\fi T%
    \toks@\expandafter{\expandafter\patchcommand\csname #6 \endcsname}%
  \else
    \ifx @#2@% No arguments
      \toks@\expandafter{\the\toks@ 0}%
    \else
      \patchcmd@c 0#2{\string##}0%
    \fi
  \fi}
```

\patchcmd@c

```
\def\patchcmd@c#1#2#3{%
```

```
  \if\string###2%      % yes it's a # token
    \ifodd 0#31 % and it's followed by a number
      \if 0#3\patchcmd@d#1\fi % number=0? then we're done
    \else \patchcmd@d D% # not a number: must be a delimited arg
    \fi
  \else \patchcmd@d D% not a # token: must be a delmited arg
  \fi
  \patchcmd@c#3}
```

\patchcmd@d

```
\def\patchcmd@d#1{%
  \if D#1%
%%%    \PackageError{patchcmd}{Cannot change a macro that has
%%%      delimited arguments}\@ehd
    \@memerror{%
      Cannot change a macro that has delimited arguments}{\@ehd}
  \else
    \toks@\expandafter{\the\toks@ #1}%
  \fi
  \begingroup
  \aftergroup\@gobble
  \let\patchcmd@c\endgroup}
```

\patchcmd@e

```
\def\patchcmd@e#1#2#3#4#5{%
  \begingroup
  \edef\@##1{%
    \@temptokena\noexpand\expandafter{%
      \noexpand#2%
        \ifnum#3>0 {####1}\ifnum#3>1 {####2}\ifnum#3>2 {####3}%
        \ifnum#3>3 {####4}\ifnum#3>4 {####5}\ifnum#3>5 {####6}%
        \ifnum#3>6 {####7}\ifnum#3>7 {####8}\ifnum#3>8 {####9}%
        \fi\fi\fi\fi\fi\fi\fi\fi\fi
      ##1%
    }%
  }
  \@{#5}%
  \edef\@##1{\endgroup
    \noexpand\renewcommand#1\noexpand#2\ifcase#3 \else [#3]\fi
    {##1\the\@temptokena}}%
  \@{#4}%
}
```

\patchcmdError

```
\long\def\patchcmdError#1#2{%
  \begingroup
  \toks@{Not redefinable}%
  \ifcat\relax\noexpand#1% Is it a control sequence?
    \begingroup
```

```
    \let#1=?\ifx ?\relax % Is it "\relax"?
      \endgroup % accept current value of \toks@
    \else \endgroup
      \if\ifx\relax#1u\else #2\fi u%
        \toks@{Not defined}%
      \fi
    \fi
  \fi
  \edef\@{\endgroup
%%%    \noexpand\PackageError{patchcmd}{%
%%%      \the\toks@: \string#1}\noexpand\@ehd}%
    \noexpand\@memerror{%
      \the\toks@: \string#1}\noexpand\@ehd}%
  \@}

%%%%%%%%%%%%%%%%%%%% end of patchcmd code %%%%%%%%%%%%%%%%%%%%%%%%%
```

Prevent later loading of the original patchcmd package.

```
%%\@memfakeusepackage{patchcmd}
```

4.5 Support for the bidi package (RTL typesetting)

The bidi package (system) enables bidirectional typesetting. As part of being able to accomplish this it often needs to exchange, for example, `\leftskip` and `\rightskip` depending on whether it is setting left-to-right (LTR) or right-to-left (RTL). In the case of memoir the initial bidi system implementation used a bidimemoir class developed by Vafa Khalighi. Now, with Vafa's help, the necessary changes have been incorporated in memoir; the bidimemoir class is no longer needed.

`\memRTLleftskip` `\memRTLrightskip` `\memRTLvleftskip` `\memRTLvrightskip` `\memRTLraggedright` `\memRTLraggedleft`

These are the hooks required to support the bidi package.

```
\newcommand*{\memRTLleftskip}{\leftskip}
\newcommand*{\memRTLrightskip}{\rightskip}
\newcommand*{\memRTLvleftskip}{\leftskip}
\newcommand*{\memRTLvrightskip}{\rightskip}
\newcommand*{\memRTLraggedright}{\raggedright}
\newcommand*{\memRTLraggedleft}{\raggedleft}
```

It is expected that the bidi package will redefine these as:

```
\renewcommand*{\memRTLleftskip}{\if@RTL\rightskip\else\leftskip\fi}
\renewcommand*{\memRTLrightskip}{\if@RTL\leftskip\else\rightskip\fi}
\renewcommand*{\memRTLvleftskip}{\if@RTL\vrightskip\else\vleftskip\fi}
\renewcommand*{\memRTLvrightskip}{\if@RTL\vleftskip\else\vrightskip\fi}
\renewcommand*{\memRTLraggedright}{\if@RTL\raggedleft\else\raggedright\fi}
\renewcommand*{\memRTLraggedleft}{\if@RTL\raggedright\else\raggedleft\fi}
```

5 Declaration of Options

5.1 Setting Paper Sizes

The variables \paperheight and \paperwidth should reflect the physical paper size after trimming.
Option letterpaper will be the default.

\stockheight
\stockwidth
\trimtop
\trimedge

The lengths \stockheight and \stockwidth should be the height and width of the stock sheet before trimming. For example, this is the physical size of a single sheet that might be laser-printed. The lengths \trimtop and \trimedge are the amount that will be trimmed off the top and fore edge of the physical sheet. For desk printer output the size of the trimmed sheet is often the same as the physical sheet. In other words the \paper... and \stock... sizes are the same and the trims are zero. This class assumes that this is the normal case.

```
\newlength{\stockheight}
\newlength{\stockwidth}
\newlength{\trimtop}
\newlength{\trimedge}
```

The class provides an extended range of stock sizes. It may be useful at some point to have macros for these.

\stockdbill
\stockstatement
\stockexecutive
\stockletter
\stockold
\stocklegal
\stockledger
\stockbroadsheet

US stock sizes.

```
\newcommand*{\stockdbill}     {\stockheight=7in    \stockwidth=3in}
\newcommand*{\stockstatement} {\stockheight=8.5in  \stockwidth=5.5in}
\newcommand*{\stockexecutive} {\stockheight=10.5in \stockwidth=7.25in}
\newcommand*{\stockletter}    {\stockheight=11in   \stockwidth=8.5in}
\newcommand*{\stockold}       {\stockheight=12in   \stockwidth=9in}
\newcommand*{\stocklegal}     {\stockheight=14in   \stockwidth=8.5in}
\newcommand*{\stockledger}    {\stockheight=17in   \stockwidth=11in}
\newcommand*{\stockbroadsheet}{\stockheight=22in   \stockwidth=17in}
```

\stockpottvo
\stockfoolscapvo
\stockcrownvo
\stockpostvo
\stocklargecrownvo
\stocklargepostvo
\stocksmalldemyvo

Traditional British octavo sizes.

```
\newcommand*{\stockpottvo}      {\stockheight=6.25in \stockwidth=4in}
\newcommand*{\stockfoolscapvo}  {\stockheight=6.75in \stockwidth=4.25in}
\newcommand*{\stockcrownvo}     {\stockheight=7.5in  \stockwidth=5in}
\newcommand*{\stockpostvo}      {\stockheight=8in    \stockwidth=5in}
\newcommand*{\stocklargecrownvo}{\stockheight=8in    \stockwidth=5.25in}
\newcommand*{\stocklargepostvo} {\stockheight=8.25in \stockwidth=5.25in}
\newcommand*{\stocksmalldemyvo} {\stockheight=8.5in  \stockwidth=5.675in}
```

\stockdemyvo
\stockmediumvo
\stocksmallroyalvo
\stockroyalvo
\stocksuperroyalvo
\stockimperialvo

```
\newcommand*{\stockdemyvo}      {\stockheight=8.75in  \stockwidth=5.675in}
\newcommand*{\stockmediumvo}    {\stockheight=9in     \stockwidth=5.75in}
\newcommand*{\stocksmallroyalvo}{\stockheight=9.25in  \stockwidth=6.175in}
\newcommand*{\stockroyalvo}     {\stockheight=10in    \stockwidth=6.25in}
```

```
\newcommand*{\stocksuperroyalvo}{\stockheight=10.25in \stockwidth=6.75in}
\newcommand*{\stockimperialvo}  {\stockheight=11in    \stockwidth=7.5in}
```

\stockmcrownvo
\stockmlargecrownvo
\stockmdemyvo
\stockmsmallroyalvo

Metric stock sizes.

```
\newcommand*{\stockmcrownvo}      {\stockheight=186mm \stockwidth=123mm}
\newcommand*{\stockmlargecrownvo} {\stockheight=198mm \stockwidth=129mm}
\newcommand*{\stockmdemyvo}       {\stockheight=216mm \stockwidth=138mm}
\newcommand*{\stockmsmallroyalvo} {\stockheight=234mm \stockwidth=156mm}
```

\stockao
\stockai
\stockaii
\stockaiii
\stockaiv
\stockav
\stockavi

The A series, A0 — A6.

```
\newcommand*{\stockao}  {\stockheight=1189mm \stockwidth=841mm}
\newcommand*{\stockai}  {\stockheight=841mm \stockwidth=594mm}
\newcommand*{\stockaii} {\stockheight=594mm \stockwidth=420mm}
\newcommand*{\stockaiii}{\stockheight=420mm \stockwidth=297mm}
\newcommand*{\stockaiv} {\stockheight=297mm \stockwidth=210mm}
\newcommand*{\stockav}  {\stockheight=210mm \stockwidth=148mm}
\newcommand*{\stockavi} {\stockheight=148mm \stockwidth=105mm}
```

\stockbo
\stockbi
\stockbii
\stockbiii
\stockbiv
\stockbv
\stockbvi

The B series, B0 — B6.

```
\newcommand*{\stockbo}  {\stockheight=1414mm \stockwidth=1000mm}
\newcommand*{\stockbi}  {\stockheight=1000mm \stockwidth=707mm}
\newcommand*{\stockbii} {\stockheight=707mm \stockwidth=500mm}
\newcommand*{\stockbiii}{\stockheight=500mm \stockwidth=353mm}
\newcommand*{\stockbiv} {\stockheight=353mm \stockwidth=250mm}
\newcommand*{\stockbv}  {\stockheight=250mm \stockwidth=176mm}
\newcommand*{\stockbvi} {\stockheight=176mm \stockwidth=125mm}
```

It will be useful also to provide macros that set the page sizes.

\pagedbill
\pagestatement
\pageexecutive
\pageletter
\pageold
\pagelegal
\pageledger
\pagebroadsheet

S page sizes.

```
\newcommand*{\pagedbill}     {\paperheight=7in    \paperwidth=3in}
\newcommand*{\pagestatement} {\paperheight=8.5in  \paperwidth=5.5in}
\newcommand*{\pageexecutive} {\paperheight=10.5in \paperwidth=7.25in}
\newcommand*{\pageletter}    {\paperheight=11in   \paperwidth=8.5in}
\newcommand*{\pageold}       {\paperheight=12in   \paperwidth=9in}
\newcommand*{\pagelegal}     {\paperheight=14in   \paperwidth=8.5in}
\newcommand*{\pageledger}    {\paperheight=17in   \paperwidth=11in}
\newcommand*{\pagebroadsheet}{\paperheight=22in   \paperwidth=17in}
```

\pagepottvo
\pagefoolscapvo
\pagecrownvo
\pagepostvo
\pagelargecrownvo
\pagelargepostvo
\pagesmalldemyvo

British traditional page sizes, octavo.

```
\newcommand*{\pagepottvo}      {\paperheight=6.25in \paperwidth=4in}
\newcommand*{\pagefoolscapvo}  {\paperheight=6.75in \paperwidth=4.25in}
\newcommand*{\pagecrownvo}     {\paperheight=7.5in  \paperwidth=5in}
\newcommand*{\pagepostvo}      {\paperheight=8in    \paperwidth=5in}
```

```
\newcommand*{\pagelargecrownvo}{\paperheight=8in    \paperwidth=5.25in}
\newcommand*{\pagelargepostvo} {\paperheight=8.25in \paperwidth=5.25in}
\newcommand*{\pagesmalldemyvo} {\paperheight=8.5in  \paperwidth=5.675in}
```

\pagedemyvo
\pagemediumvo
\pagesmallroyalvo
\pageroyalvo
\pagesuperroyalvo
\pageimperialvo

```
\newcommand*{\pagedemyvo}      {\paperheight=8.75in  \paperwidth=5.675in}
\newcommand*{\pagemediumvo}    {\paperheight=9in     \paperwidth=5.75in}
\newcommand*{\pagesmallroyalvo}{\paperheight=9.25in  \paperwidth=6.175in}
\newcommand*{\pageroyalvo}     {\paperheight=10in    \paperwidth=6.25in}
\newcommand*{\pagesuperroyalvo}{\paperheight=10.25in \paperwidth=6.75in}
\newcommand*{\pageimperialvo}  {\paperheight=11in    \paperwidth=7.5in}
```

\pagemcrownvo
\pagemlargecrownvo
\pagemdemyvo
\pagemsmallroyalvo

Metric sizes.

```
\newcommand*{\pagemcrownvo}      {\paperheight=186mm \paperwidth=123mm}
\newcommand*{\pagemlargecrownvo} {\paperheight=198mm \paperwidth=129mm}
\newcommand*{\pagemdemyvo}       {\paperheight=216mm \paperwidth=138mm}
\newcommand*{\pagemsmallroyalvo} {\paperheight=234mm \paperwidth=156mm}
```

\pageao
\pageai
\pageaii
\pageaiii
\pageaiv
\pageav
\pageavi

The A series, A0 — A6.

```
\newcommand*{\pageao}  {\paperheight=1189mm \paperwidth=841mm}
\newcommand*{\pageai}  {\paperheight=841mm \paperwidth=594mm}
\newcommand*{\pageaii} {\paperheight=594mm \paperwidth=420mm}
\newcommand*{\pageaiii}{\paperheight=420mm \paperwidth=297mm}
\newcommand*{\pageaiv} {\paperheight=297mm \paperwidth=210mm}
\newcommand*{\pageav}  {\paperheight=210mm \paperwidth=148mm}
\newcommand*{\pageavi} {\paperheight=148mm \paperwidth=105mm}
```

\pagebo
\pagebi
\pagebii
\pagebiii
\pagebiv
\pagebv
\pagebvi

The B series, B0 — B6.

```
\newcommand*{\pagebo}  {\paperheight=1414mm \paperwidth=1000mm}
\newcommand*{\pagebi}  {\paperheight=1000mm \paperwidth=707mm}
\newcommand*{\pagebii} {\paperheight=707mm \paperwidth=500mm}
\newcommand*{\pagebiii}{\paperheight=500mm \paperwidth=353mm}
\newcommand*{\pagebiv} {\paperheight=353mm \paperwidth=250mm}
\newcommand*{\pagebv}  {\paperheight=250mm \paperwidth=176mm}
\newcommand*{\pagebvi} {\paperheight=176mm \paperwidth=125mm}
```

Declare the stock size options.
Metric paper stock sizes.

```
\DeclareOption{a0paper}{\stockao}
\DeclareOption{a1paper}{\stockai}
\DeclareOption{a2paper}{\stockaii}
\DeclareOption{a3paper}{\stockaiii}
\DeclareOption{a4paper}{\stockaiv}
\DeclareOption{a5paper}{\stockav}
\DeclareOption{a6paper}{\stockavi}
```

```
\DeclareOption{b0paper}{\stockbo}
\DeclareOption{b1paper}{\stockbi}
\DeclareOption{b2paper}{\stockbii}
\DeclareOption{b3paper}{\stockbiii}
\DeclareOption{b4paper}{\stockbiv}
\DeclareOption{b5paper}{\stockbv}
\DeclareOption{b6paper}{\stockbvi}
\DeclareOption{mcrownvopaper}{\stockmcrownvo}
\DeclareOption{mlargecrownvopaper}{\stockmlargecrownvo}
\DeclareOption{mdemyvopaper}{\stockmdemyvo}
\DeclareOption{msmallroyalvopaper}{\stockmsmallroyalvo}
```

US paper stock sizes.

```
\DeclareOption{dbillpaper}{\stockdbill}
\DeclareOption{statementpaper}{\stockstatement}
\DeclareOption{executivepaper}{\stockexecutive}
\DeclareOption{letterpaper}{\stockletter}
\DeclareOption{oldpaper}{\stockold}
\DeclareOption{legalpaper}{\stocklegal}
\DeclareOption{ledgerpaper}{\stockledger}
\DeclareOption{broadsheetpaper}{\stockbroadsheet}
```

British octavo stock paper sizes.

```
\DeclareOption{pottvopaper}{\stockpottvo}
\DeclareOption{foolscapvopaper}{\stockfoolscapvo}
\DeclareOption{crownvopaper}{\stockcrownvo}
\DeclareOption{postvopaper}{\stockpostvo}
\DeclareOption{largecrownvopaper}{\stocklargecrownvo}
\DeclareOption{largepostvopaper}{\stocklargepostvo}
\DeclareOption{smalldemyvopaper}{\stocksmalldemyvo}
\DeclareOption{demyvopaper}{\stockdemyvo}
\DeclareOption{mediumvopaper}{\stockmediumvo}
\DeclareOption{smallroyalvopaper}{\stocksmallroyalvo}
\DeclareOption{royalvopaper}{\stockroyalvo}
\DeclareOption{superroyalvopaper}{\stocksuperroyalvo}
\DeclareOption{imperialvopaper}{\stockimperialvo}
```

Ebook.

```
\DeclareOption{ebook}
   {\setlength\stockheight {9in}%
    \setlength\stockwidth  {6in}}
```

The landscape option switches the values of the height and width, assuming that the dimensions were originally given for portrait orientation. (At the suggestion of Wilhelm Müller made this independent of the option sequence). .

```
  \ifmemlandscape
 \memlandscapetrue
\memlandscapefalse
```

```
\newif\ifmemlandscape
  \memlandscapefalse
\DeclareOption{landscape}{\memlandscapetrue}
\DeclareOption{portrait}{\memlandscapefalse}
```

5.2 Choosing the type size

The type size options are handled by defining `\@ptsize` to contain the last digit of the size in question and branching on `\ifcase` statements. This is done for historical reasons to stay compatible with other packages that use the `\@ptsize` variable to select special actions. It makes the declarations of size options less than 10pt or more than 20pt difficult. In this class `9` is used for the `9pt` option, assuming that the class will never define a `19pt` option. For larger options the full complement of digits are used.
In any event, `\@memptsize` holds the complete size.
Option `10pt` will be the default.

```
\renewcommand*{\@ptsize}{0}
\renewcommand*{\@memptsize}{10}
\DeclareOption{9pt}{\renewcommand*{\@ptsize}{9}\renewcommand*{\@memptsize}{9}}
\DeclareOption{10pt}{\renewcommand*{\@ptsize}{0}\renewcommand*{\@memptsize}{10}}
\DeclareOption{11pt}{\renewcommand*{\@ptsize}{1}\renewcommand*{\@memptsize}{11}}
\DeclareOption{12pt}{\renewcommand*{\@ptsize}{2}\renewcommand*{\@memptsize}{12}}
\DeclareOption{14pt}{\renewcommand*{\@ptsize}{4}\renewcommand*{\@memptsize}{14}}
\DeclareOption{17pt}{\renewcommand*{\@ptsize}{7}\renewcommand*{\@memptsize}{17}}
\DeclareOption{20pt}{\renewcommand*{\@ptsize}{20}\renewcommand*{\@memptsize}{20}}
\DeclareOption{25pt}{\renewcommand*{\@ptsize}{25}\renewcommand*{\@memptsize}{25}}
\DeclareOption{30pt}{\renewcommand*{\@ptsize}{30}\renewcommand*{\@memptsize}{30}}
\DeclareOption{36pt}{\renewcommand*{\@ptsize}{36}\renewcommand*{\@memptsize}{36}}
\DeclareOption{48pt}{\renewcommand*{\@ptsize}{48}\renewcommand*{\@memptsize}{48}}
\DeclareOption{60pt}{\renewcommand*{\@ptsize}{60}\renewcommand*{\@memptsize}{60}}
```

`\if@nyptsizeopt` `\@nyptsizeopttrue` `\@nyptsizeoptfalse` `\anyptfilebase` `\anyptsize`

For any point size:

```
\newif\if@nyptsizeopt
  \@nyptsizeoptfalse
\providecommand*{\anyptfilebase}{mem}
\providecommand*{\anyptsize}{10}
\DeclareOption{*pt}{\@nyptsizeopttrue}
```

5.3 Two-side or one-side printing

For two-sided printing we use the switch `\if@twoside`. In addition we have to set the `\if@mparswitch` to get any margin paragraphs into the outside margin. The default is `twoside`.

\if@twoside
\if@mparswitch
```
\DeclareOption{twoside}{\@twosidetrue  \@mparswitchtrue}
\DeclareOption{oneside}{\@twosidefalse \@mparswitchfalse}
```

5.4 Two column printing

Two-column and one-column printing is again realized via a switch which is defined in the kernel. The default is single column printing.

\if@twocolumn
```
\DeclareOption{onecolumn}{\@twocolumnfalse}
\DeclareOption{twocolumn}{\@twocolumntrue}
```

5.5 The draft option

If the user requests draft we show any overfull boxes, marginal notes are allowed, and any copyright notices are not printed. For symmetry, we also define a final option which is the default.
The user can use the \ifdraftdoc flag to add additional effects:
\ifdraftdoc <additional code> \fi

\ifdraftdoc
```
\newif\ifdraftdoc\draftdocfalse
\setlength{\overfullrule}{\z@}
\DeclareOption{final}{\setlength{\overfullrule}{\z@}
                      \draftdocfalse
                      \msdocfalse}
\DeclareOption{draft}{\setlength\overfullrule{5pt}%
                      \draftdoctrue
                      \msdocfalse}
```

5.6 The ms option

\ifmsdoc The ms option makes the document look as though it was produced on a typewriter. We use a flag for remembering this. The user may also use the flag for specifying ms effects.

```
\newif\ifmsdoc
  \msdocfalse
\DeclareOption{ms}{%
  \msdoctrue
  \draftdocfalse
  \setlength\overfullrule{\z@}
}

```

5.7 The showtrims option

The showtrims option will display crosses at the corners of the logical pages showing where the stock should be trimmed.

\ifshowtrims

```
\newif\ifshowtrims
  \showtrimsfalse
\DeclareOption{showtrims}{\showtrimstrue}
```

5.8 The article option

The article option typesets as a simulation of the article class.

\ifartopt \ifartopt is a flag (TRUE) if the article option is called.

```
\newif\ifartopt
  \artoptfalse
\DeclareOption{article}{\artopttrue}
```

5.9 The subfigure option

The subfigure option has been made a no-op since version 1.1 and finally removed in version 1.61803. It was:

```
\DeclareOption{subfigure}{%
  \ClassWarningNoLine{memoir}{The subfigure option is not required}}
```

5.10 The openright, openleft and openany options

The openright option specifies that Chapters must begin on recto pages. The openleft option specifies that Chapters must begin on verso pages and the openany option lets Chapters start on any page.

```
\DeclareOption{openright}{\@openrighttrue}
\DeclareOption{openany}{\@openrightfalse}
\DeclareOption{openleft}{\@openlefttrue}
```

\openright \openany \openleft Commands that can be used to change the option in the middle of the document[11].

```
\newcommand{\openright}{\@openrighttrue\@openleftfalse%
  \gdef\clearforchapter{\cleartorecto}}
\newcommand{\openany}{\@openrightfalse\@openleftfalse%
  \gdef\clearforchapter{\clearpage}}
\newcommand{\openleft}{\@openlefttrue
  \gdef\clearforchapter{\cleartoverso}}
```

[11] Openleft provided to meet a request by Vladimir G. Ivanovic (vladimir@acm.org) in September 2001.

5.11 Equation numbering on the left

The leqno option prints equation numbers on the left. This is implemented via an external class option file.

```
\DeclareOption{leqno}{\input{leqno.clo}}
```

5.12 Flush left math displays

The fleqn option redefines the displayed math environments so that they are left adjusted with an indent of \mathindent from the current left margin. This is implemented via an external class option file.

```
\DeclareOption{fleqn}{\input{fleqn.clo}}
```

5.13 Open bibliography

The openbib option redefines the thebibliography so that each block starts on a new line, and succeeding lines in a block are indented by \bibindent.

```
\DeclareOption{openbib}{%
  \AtEndOfClass{%
    \renewcommand\@openbib@code{%
      \advance\leftmargin\bibindent
      \itemindent -\bibindent
      \listparindent \itemindent
      \parsep \z@
     }%
    \renewcommand\newblock{\par}}}

```

5.14 Old font commands

The oldfontcommands option enables commands like \bf and friends.

```
\DeclareOption{oldfontcommands}{\@memoldfonttrue}
```

5.15 Extra font sizes

The extrafontsizes options indicates that extended font sizes (above 25pt) are available.

```
\DeclareOption{extrafontsizes}{\extrafontsizestrue}

```

5.16 ETeX

If found we automatically load the eTeX package right after executing the class options. Though, some odd LaTeX installations may have the eTeX pacakge, but is not based on eTeX (eventhough that is what the LaTeX-project have recommended for several years now), so we add a dead mans switch to disable the loading.

\ifmem@noetex
\mem@noetextrue
\mem@noetexfalse

```
\newif\ifmem@noetex
  \mem@noetexfalse
\DeclareOption{noetex}{\mem@noetextrue}

```

6 Executing Options

Here we execute the default options to initialize certain variables.

```
\ExecuteOptions{final,letterpaper,10pt,onecolumn,openright,twoside,
                portrait}
```

The \ProcessOptions command causes the execution of the code for every option foo which is declared and for which the user typed the foo option in his \documentclass command. For every option bar he typed, which is not declared, the option is assumed to be a global option. All options will be passed as document options to any \usepackage command in the document preamble. \ProcessOptions* processes the options in the order they are given in the \documentclass command, instead of the definition order.

```
\ProcessOptions*
  \ifmemlandscape
    \setlength\@tempdima  {\stockheight}
    \setlength\stockheight{\stockwidth}
    \setlength\stockwidth {\@tempdima}
  \fi

```

Load the eTeX package if found, and if the user has not explicitly asked us not to.

```
\ifmem@noetex\relax\else
  \IfFileExists{etex.sty}{\RequirePackage{etex}}{}
\fi
```

\memoirpostopthook

A user could define this *before* the \documentclass command to do something at this point. For example, to set up a new stock size:
\def\memoirpostopthook{\stockheight=44in \stockwidth=34in}

```
\providecommand*{\memoirpostopthook}{}
  \memoirpostopthook

```

\cleartorecto

A repeat of \cleardoublepage; clears to a recto (odd-numbered) page.

```
\def\cleartorecto{\clearpage\if@twoside \ifodd\c@page\else
  \hbox{}\thispagestyle{cleared}%
  \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}

```

\cleartoverso

Clears to a verso (even-numbered) page.

```
\def\cleartoverso{\clearpage\if@twoside
  \ifodd\c@page\hbox{}\thispagestyle{cleared}%
```

```
  \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
```

Set `\clearforchapter` according to the options.

```
\if@openleft
  \openleft
\else
  \if@openright
    \openright
  \else
    \openany
  \fi
\fi
```

`\@ivpt` `\@xxxpt` `\@xxxvipt` `\@xlviiipt` `\@lxpt` `\@lxxiipt` `\@lxxxivpt` `\@xcvipt` `\@cviiipt` `\@cxxpt` `\@cxxxiipt`

```
\newcommand*{\@ivpt}{4}
\newcommand*{\@xxxpt}{30}
\newcommand*{\@xxxvipt}{36}
\newcommand*{\@xlviiipt}{48}
\newcommand*{\@lxpt}{60}
\newcommand*{\@lxxiipt}{72}
\newcommand*{\@lxxxivpt}{84}
\newcommand*{\@xcvipt}{96}
\newcommand*{\@cviiipt}{108}
\newcommand*{\@cxxpt}{120}
\newcommand*{\@cxxxiipt}{132}
```

Now that all the options have been executed we can load the chosen class option file that contains all size dependent code. The larger sizes are only made available for the extrafontsizes option.

`\memfontfamily` `\memfontenc` `\memfontpack`

The default font and coding as set by the kernel is `cmr` and `OT1` respectively, giving Knuth's original Computer Modern Roman font at a set of fixed sizes (maximum of 24.88pt). If we are to have any size font we have to go to a scaleable font. As the Latin Modern font is a scaleable version of Computer Modern, and all modern LaTeX distributions include it, I am using this by default.
By defining these three macros *before* the `\documentclass` any other font and package can be used.

```
\providecommand*{\memfontfamily}{lmr}
\providecommand*{\memfontenc}{T1}
\providecommand*{\memfontpack}{lmodern}
```

Deal with the 'anyptsize' (*pt) option first. In this case the macros `\anyptfilebase` and `\anyptsize` should have been defined by the author *before* the `\documentclass` command, and a file

\anyptfilebase\anyptsize.clo should exist (if not the file will default to mem10.clo).

```
\if@nyptsizeopt
  \newcommand*{\@nyptclofile}{\anyptfilebase\anyptsize.clo}
  \IfFileExists{\@nyptclofile}{\def\@memptsize{\anyptsize}}{%
    \@mememror{You have used the `*pt' option but \MessageBreak
               file \@nyptclofile\space can't be found}%
              {I'll use mem10.clo instead}
    \renewcommand*{\@nyptclofile}{mem10.clo}%
    \def\@memptsize{10}%
  }
  \renewcommand*{\@ptsize}{\@memptsize}
  \usefont{\memfontenc}{\memfontfamily}{m}{n}
  \input{\@nyptclofile}
  \usepackage{\memfontpack}\usepackage[\memfontenc]{fontenc}
\else
```

Now for the fixed pt size options.

```
  \ifextrafontsizes
    \usefont{\memfontenc}{\memfontfamily}{m}{n}
    \input{mem\@memptsize.clo}
    \usepackage{\memfontpack}\usepackage[\memfontenc]{fontenc}
  \else
    \ifnum\@memptsize > 17\relax
      \@mememror{The `extrafontsizes' option is required to use \MessageBreak
                 the `\@memptsize pt' option}%
                {The 17pt option will be used instead}
      \input{mem17.clo}
    \else
      \ifnum\@ptsize = 9\relax
        \input{mem\@ptsize.clo}
      \else
        \input{mem1\@ptsize.clo}
      \fi
    \fi
  \fi
\fi

⟨/class⟩
```

7 Fonts and spaces

In this section we deal with most of the aspects that are related to font sizes, and spacing that is related to the size of the body font.

7.1 Fonts

LaTeX offers the user commands to change the size of the font, relative to the 'main' size. Each relative size changing command \size executes the command

Table 1: Shorthand font point size commands

				\@ivpt	4
\@vpt	5	\@vipt	6	\@viipt	7
\@viiipt	8	\@ixpt	9	\@xpt	10
\@xipt	10.95	\@xiipt	12	\@xivpt	14.4
\@xviipt	17.28	\@xxpt	20.74	\@xxvpt	24.88
\@xxxpt	30	\@xxxvipt	36	\@xlviiipt	48
\@lxpt	60	\@lxxiipt	72	\@lxxxivpt	84
\@xcvipt	96	\@cviiipt	108	\@cxxpt	120
\@cxxxiipt	132				

`\@setfontsize\size` ⟨*font-size*⟩ ⟨*baselineskip*⟩ where:

⟨***font-size***⟩ The absolute size of the font to use from now on.

⟨***baselineskip***⟩ The normal value of `\baselineskip` for the size of the font selected. (The actual value will be `\baselinestretch` * ⟨*baselineskip*⟩.)

A number of commands, defined in the LaTeX kernel, shorten the following definitions and are used throughout. They are listed in the first part of Table 1. Those in the second part are introduced by the class.

`\normalsize` `\@normalsize` The user level command for the main size is `\normalsize`. Internally LaTeX uses `\@normalsize` when it refers to the main size. `\@normalsize` will be defined to work like `\normalsize` if the latter is redefined from its default definition (that just issues an error message). Otherwise, in the standard classes `\@normalsize` simply selects a 10pt/12pt size, but here it selects among the wider range. The `\normalsize` macro also sets new values for `\abovedisplayskip`, `\abovedisplayshortskip` and `\belowdisplayshortskip`. For the larger sizes I have set `\abovedisplayskip` to the font size and `\belowdisplayshortskip` to 0.5`\onelineskip`.

```
781 ⟨*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
782 \renewcommand{\normalsize}{%
783 ⟨*9pt⟩
784    \@setfontsize\normalsize\@ixpt\@xpt
785    \abovedisplayskip 9\p@ \@plus 2\p@ \@minus 4.5\p@
786    \abovedisplayshortskip \z@ \@plus 3\p@
787    \belowdisplayshortskip 5.5\p@ \@plus 2.5\p@ \@minus 3\p@
788 ⟨/9pt⟩
789 ⟨*10pt⟩
790    \@setfontsize\normalsize\@xpt\@xiipt
791    \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
792    \abovedisplayshortskip \z@ \@plus3\p@
793    \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
794 ⟨/10pt⟩
795 ⟨*11pt⟩
```

```
    \@setfontsize\normalsize\@xipt{13.6}%
    \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
⟨/11pt⟩
⟨*12pt⟩
    \@setfontsize\normalsize\@xiipt{14.5}%
    \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
⟨/12pt⟩
⟨*14pt⟩
    \@setfontsize\normalsize\@xivpt{17.5}%
    \abovedisplayskip 14\p@ \@plus3\p@ \@minus8\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 7\p@ \@plus3.5\p@ \@minus3\p@
⟨/14pt⟩
⟨*17pt⟩
    \@setfontsize\normalsize\@xviipt{22}%
    \abovedisplayskip 15\p@ \@plus4\p@ \@minus8\p@
    \abovedisplayshortskip \z@ \@plus4\p@
    \belowdisplayshortskip 8\p@ \@plus4\p@ \@minus3\p@
⟨/17pt⟩
⟨*20pt⟩
    \@setfontsize\normalsize\@xxpt{25}%
    \abovedisplayskip 20\p@ \@plus5\p@ \@minus9\p@
    \abovedisplayshortskip \z@ \@plus5\p@
    \belowdisplayshortskip 12.5\p@ \@plus6\p@ \@minus3\p@
⟨/20pt⟩
⟨*25pt⟩
    \@setfontsize\normalsize\@xxvpt{30}%
    \abovedisplayskip 25\p@ \@plus6\p@ \@minus10\p@
    \abovedisplayshortskip \z@ \@plus6\p@
    \belowdisplayshortskip 15\p@ \@plus7.5\p@ \@minus4\p@
⟨/25pt⟩
⟨*30pt⟩
    \@setfontsize\normalsize\@xxxpt{37}%
    \abovedisplayskip 30\p@ \@plus7\p@ \@minus11\p@
    \abovedisplayshortskip \z@ \@plus7\p@
    \belowdisplayshortskip 18\p@ \@plus9\p@ \@minus4\p@
⟨/30pt⟩
⟨*36pt⟩
    \@setfontsize\normalsize\@xxxvipt{45}%
    \abovedisplayskip 36\p@ \@plus8\p@ \@minus12\p@
    \abovedisplayshortskip \z@ \@plus8\p@
    \belowdisplayshortskip 22\p@ \@plus11\p@ \@minus5\p@
⟨/36pt⟩
⟨*48pt⟩
    \@setfontsize\normalsize\@xlviiipt{60}%
    \abovedisplayskip 48\p@ \@plus9\p@ \@minus13\p@
```

```
    \abovedisplayshortskip \z@ \@plus9\p@
    \belowdisplayshortskip 30\p@ \@plus15\p@ \@minus7\p@
⟨/48pt⟩
⟨*60pt⟩
    \@setfontsize\normalsize\@lxpt{72}%
    \abovedisplayskip 60\p@ \@plus10\p@ \@minus14\p@
    \abovedisplayshortskip \z@ \@plus10\p@
    \belowdisplayshortskip 36\p@ \@plus18\p@ \@minus9\p@
⟨/60pt⟩
```

The \belowdisplayskip is always equal to the \abovedisplayskip. The parameters of the first level list are always given by \@listI.

```
    \belowdisplayskip \abovedisplayskip
    \let\@listi\@listI}
```

We initially choose the normalsize font.

```
\normalsize
```

\small This code is similar to that for \normalsize. (Robert Schlicht[12] noted that in some cases % signs were missing after \@setfontsize).
For the larger sizes I have set \topsep to 2/3\onelineskip and \parsep to 1/2\topsep.

```
\newcommand{\small}{%
⟨*9pt⟩
    \@setfontsize\small\@viiipt{9.5}%
    \abovedisplayskip 6\p@ \@plus 2\p@ \@minus 4\p@
    \abovedisplayshortskip \z@ \@plus 2\p@
    \belowdisplayshortskip 4\p@ \@plus 2\p@ \@minus 2\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 2\p@ \@plus 2\p@ \@minus 2\p@
                \parsep 1\p@ \@plus\p@ \@minus\p@
                \itemsep \parsep
%%                \itemindent\z@
                }%
⟨/9pt⟩
⟨*10pt⟩
    \@setfontsize\small\@ixpt{11}%
    \abovedisplayskip 8.5\p@ \@plus3\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus2\p@
    \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 4\p@ \@plus2\p@ \@minus2\p@
                \parsep 2\p@ \@plus\p@ \@minus\p@
                \itemsep \parsep
%%                \itemindent\z@
               }%
⟨/10pt⟩
⟨*11pt⟩
```

[12] w.m.l@gmx.net, via email on 2004/03/11.

```
    \@setfontsize\small\@xpt\@xiipt
    \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 6\p@ \@plus2\p@ \@minus2\p@
                \parsep 3\p@ \@plus2\p@ \@minus\p@
                \itemsep \parsep
%%              \itemindent\z@
               }%
⟨/11pt⟩
⟨*12pt⟩
    \@setfontsize\small\@xipt{13.6}%
    \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 9\p@ \@plus3\p@ \@minus5\p@
                \parsep 4.5\p@ \@plus2\p@ \@minus\p@
                \itemsep \parsep
%%              \itemindent\z@
               }%
⟨/12pt⟩
⟨*14pt⟩
    \@setfontsize\small\@xiipt{14.5}%
    \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 11\p@ \@plus4\p@ \@minus6\p@
                \parsep 6\p@ \@plus3\p@ \@minus\p@
                \itemsep \parsep
%%              \itemindent\z@
               }%
⟨/14pt⟩
⟨*17pt⟩
    \@setfontsize\small\@xivpt{17}%
    \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 7\p@ \@plus4\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 11\p@ \@plus4\p@ \@minus6\p@
                \parsep 6\p@ \@plus3\p@ \@minus\p@
                \itemsep \parsep
%%              \itemindent\z@
               }%
⟨/17pt⟩
⟨*20pt⟩
    \@setfontsize\small\@xviipt{22}%
    \abovedisplayskip 17\p@ \@plus4\p@ \@minus8\p@
```

```
    \abovedisplayshortskip \z@ \@plus4\p@
    \belowdisplayshortskip 11\p@ \@plus4\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 14\p@ \@plus5\p@ \@minus7\p@
                \parsep 7\p@ \@plus2\p@ \@minus3\p@
                \itemsep \parsep}%
⟨/20pt⟩
⟨*25pt⟩
    \@setfontsize\small\@xxpt{25}%
    \abovedisplayskip 20\p@ \@plus5\p@ \@minus9\p@
    \abovedisplayshortskip \z@ \@plus5\p@
    \belowdisplayshortskip 12.5\p@ \@plus6\p@ \@minus3\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 16\p@ \@plus5\p@ \@minus8\p@
                \parsep 8\p@ \@plus3\p@ \@minus4\p@
                \itemsep \parsep}%
⟨/25pt⟩
⟨*30pt⟩
    \@setfontsize\small\@xxvpt{30}%
    \abovedisplayskip 25\p@ \@plus6\p@ \@minus10\p@
    \abovedisplayshortskip \z@ \@plus6\p@
    \belowdisplayshortskip 15\p@ \@plus7.5\p@ \@minus4\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 20\p@ \@plus7\p@ \@minus10\p@
                \parsep 10\p@ \@plus4\p@ \@minus5\p@
                \itemsep \parsep}%
⟨/30pt⟩
⟨*36pt⟩
    \@setfontsize\small\@xxxpt{37}%
    \abovedisplayskip 30\p@ \@plus7\p@ \@minus11\p@
    \abovedisplayshortskip \z@ \@plus7\p@
    \belowdisplayshortskip 18\p@ \@plus9\p@ \@minus4\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 24\p@ \@plus8\p@ \@minus12\p@
                \parsep 12\p@ \@plus4\p@ \@minus6\p@
                \itemsep \parsep}%
⟨/36pt⟩
⟨*48pt⟩
    \@setfontsize\small\@xxxvipt{45}%
    \abovedisplayskip 36\p@ \@plus8\p@ \@minus12\p@
    \abovedisplayshortskip \z@ \@plus8\p@
    \belowdisplayshortskip 22\p@ \@plus11\p@ \@minus5\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 30\p@ \@plus10\p@ \@minus15\p@
                \parsep 15\p@ \@plus5\p@ \@minus7\p@
                \itemsep \parsep}%
⟨/48pt⟩
⟨*60pt⟩
    \@setfontsize\small\@xlviiipt{60}%
    \abovedisplayskip 48\p@ \@plus9\p@ \@minus13\p@
```

```
    \abovedisplayshortskip \z@ \@plus9\p@
    \belowdisplayshortskip 30\p@ \@plus15\p@ \@minus7\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 40\p@ \@plus13\p@ \@minus20\p@
                \parsep 20\p@ \@plus6\p@ \@minus10\p@
                \itemsep \parsep}%
⟨/60pt⟩
    \belowdisplayskip \abovedisplayskip
}
```

`\footnotesize` This code is similar to that for `\small`.

```
\newcommand{\footnotesize}{%
⟨*9pt⟩
     \@setfontsize\footnotesize\@viipt{8}%
     \abovedisplayskip 6\p@ \@plus 2\p@ \@minus 4\p@
     \abovedisplayshortskip \z@ \@plus 2\p@
     \belowdisplayshortskip 4\p@ \@plus 2\p@ \@minus 2\p@
     \def\@listi{\leftmargin\leftmargini
                 \topsep 2\p@ \@plus 2\p@ \@minus 2\p@
                 \parsep 1\p@ \@plus\p@ \@minus\p@
                 \itemsep \parsep
%%                 \itemindent\z@
                }%
⟨/9pt⟩
⟨*10pt⟩
    \@setfontsize\footnotesize\@viiipt{9.5}%
    \abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus\p@
    \belowdisplayshortskip 3\p@ \@plus\p@ \@minus2\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 3\p@ \@plus\p@ \@minus\p@
                \parsep 2\p@ \@plus\p@ \@minus\p@
                \itemsep \parsep
%%                \itemindent\z@
               }%
⟨/10pt⟩
⟨*11pt⟩
    \@setfontsize\footnotesize\@ixpt{11}%
    \abovedisplayskip 8\p@ \@plus2\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus\p@
    \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
    \def\@listi{\leftmargin\leftmargini
                \topsep 4\p@ \@plus2\p@ \@minus2\p@
                \parsep 2\p@ \@plus\p@ \@minus\p@
                \itemsep \parsep
%%                \itemindent\z@
               }%
⟨/11pt⟩
⟨*12pt⟩
    \@setfontsize\footnotesize\@xpt\@xiipt
```

```
1032    \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
1033    \abovedisplayshortskip \z@ \@plus3\p@
1034    \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
1035    \def\@listi{\leftmargin\leftmargini
1036                \topsep 6\p@ \@plus2\p@ \@minus2\p@
1037                \parsep 3\p@ \@plus2\p@ \@minus\p@
1038                \itemsep \parsep
1039  %%              \itemindent\z@
1040               }%
1041  ⟨/12pt⟩
1042  ⟨*14pt⟩
1043    \@setfontsize\footnotesize\@xipt{13.6}%
1044    \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
1045    \abovedisplayshortskip \z@ \@plus3\p@
1046    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
1047    \def\@listi{\leftmargin\leftmargini
1048                \topsep 6.5\p@ \@plus3.5\p@ \@minus3\p@
1049                \parsep 4\p@ \@plus3\p@ \@minus\p@
1050                \itemsep \parsep
1051  %%              \itemindent\z@
1052               }%
1053  ⟨/14pt⟩
1054  ⟨*17pt⟩
1055    \@setfontsize\footnotesize\@xiipt{14}%
1056    \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
1057    \abovedisplayshortskip \z@ \@plus3\p@
1058    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
1059    \def\@listi{\leftmargin\leftmargini
1060                \topsep 6.5\p@ \@plus3.5\p@ \@minus3\p@
1061                \parsep 4\p@ \@plus3\p@ \@minus\p@
1062                \itemsep \parsep
1063  %%              \itemindent\z@
1064               }%
1065  ⟨/17pt⟩
1066  ⟨*20pt⟩
1067    \@setfontsize\footnotesize\@xivpt{17.5}%
1068    \abovedisplayskip 14\p@ \@plus3\p@ \@minus8\p@
1069    \abovedisplayshortskip \z@ \@plus3\p@
1070    \belowdisplayshortskip 7\p@ \@plus3.5\p@ \@minus3\p@
1071    \def\@listi{\leftmargin\leftmargini
1072                \topsep 12\p@ \@plus4\p@ \@minus6\p@
1073                \parsep 6\p@ \@plus2\p@ \@minus3\p@
1074                \itemsep \parsep}%
1075  ⟨/20pt⟩
1076  ⟨*25pt⟩
1077    \@setfontsize\footnotesize\@xviipt{22}%
1078    \abovedisplayskip 17\p@ \@plus4\p@ \@minus8\p@
1079    \abovedisplayshortskip \z@ \@plus4\p@
1080    \belowdisplayshortskip 11\p@ \@plus4\p@ \@minus3\p@
1081    \def\@listi{\leftmargin\leftmargini
```

```
                \topsep 14\p@ \@plus5\p@ \@minus7\p@
                \parsep 7\p@ \@plus2\p@ \@minus3\p@
                \itemsep \parsep}%
⟨/25pt⟩
⟨*30pt⟩
   \@setfontsize\footnotesize\@xxpt{25}%
   \abovedisplayskip 20\p@ \@plus5\p@ \@minus9\p@
   \abovedisplayshortskip \z@ \@plus5\p@
   \belowdisplayshortskip 12.5\p@ \@plus6\p@ \@minus3\p@
   \def\@listi{\leftmargin\leftmargini
                \topsep 16\p@ \@plus5\p@ \@minus8\p@
                \parsep 8\p@ \@plus3\p@ \@minus4\p@
                \itemsep \parsep}%
⟨/30pt⟩
⟨*36pt⟩
   \@setfontsize\footnotesize\@xxvpt{30}%
   \abovedisplayskip 25\p@ \@plus6\p@ \@minus10\p@
   \abovedisplayshortskip \z@ \@plus6\p@
   \belowdisplayshortskip 15\p@ \@plus7.5\p@ \@minus4\p@
   \def\@listi{\leftmargin\leftmargini
                \topsep 20\p@ \@plus7\p@ \@minus10\p@
                \parsep 10\p@ \@plus4\p@ \@minus5\p@
                \itemsep \parsep}%
⟨/36pt⟩
⟨*48pt⟩
   \@setfontsize\footnotesize\@xxxpt{37}%
   \abovedisplayskip 30\p@ \@plus7\p@ \@minus11\p@
   \abovedisplayshortskip \z@ \@plus7\p@
   \belowdisplayshortskip 18\p@ \@plus9\p@ \@minus4\p@
   \def\@listi{\leftmargin\leftmargini
                \topsep 24\p@ \@plus8\p@ \@minus12\p@
                \parsep 12\p@ \@plus4\p@ \@minus6\p@
                \itemsep \parsep}%
⟨/48pt⟩
⟨*60pt⟩
   \@setfontsize\footnotesize\@xxxvipt{45}%
   \abovedisplayskip 36\p@ \@plus8\p@ \@minus12\p@
   \abovedisplayshortskip \z@ \@plus8\p@
   \belowdisplayshortskip 22\p@ \@plus11\p@ \@minus5\p@
   \def\@listi{\leftmargin\leftmargini
                \topsep 30\p@ \@plus10\p@ \@minus15\p@
                \parsep 15\p@ \@plus5\p@ \@minus7\p@
                \itemsep \parsep}%
⟨/60pt⟩
   \belowdisplayskip \abovedisplayskip
}
```

\miniscule
\scriptsize
\tiny
\large
\Large
\LARGE
\huge
\Huge
\HUGE

These are all much simpler than the previous macros, they just select a new fontsize, but leave the parameters for displays and lists alone. The class provides two additional sizes, `\miniscule` and `\HUGE`, with respect to the usual set. For

the larger sizes (e.g., 72pt and above) I have made the \baselineskip approximately 20 precent larger than the pt size.

```
⟨*9pt⟩
\ifextrafontsizes
  \newcommand*{\miniscule}{\@setfontsize\miniscule\@ivpt{5}}
\else
  \newcommand*{\miniscule}{\@setfontsize\miniscule\@vpt{6}}
\fi
\newcommand*{\tiny}{\@setfontsize\tiny\@vpt{6}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@vipt{7}}
\newcommand*{\large}{\@setfontsize\large\@xpt{12}}
\newcommand*{\Large}{\@setfontsize\Large\@xipt{13.6}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xiipt{14.5}}
\newcommand*{\huge}{\@setfontsize\huge\@xivpt{18}}
\newcommand*{\Huge}{\@setfontsize\Huge\@xviipt{22}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@xxpt{25}}
⟨/9pt⟩
⟨*10pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@vpt{6}}
\newcommand*{\tiny}{\@setfontsize\tiny\@vipt{7}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@viipt{8}}
\newcommand*{\large}{\@setfontsize\large\@xipt{13.6}}
\newcommand*{\Large}{\@setfontsize\Large\@xiipt{14.5}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xivpt{18}}
\newcommand*{\huge}{\@setfontsize\huge\@xviipt{22}}
\newcommand*{\Huge}{\@setfontsize\Huge\@xxpt{25}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@xxvpt{30}}
⟨/10pt⟩
⟨*11pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@vipt{7}}
\newcommand*{\tiny}{\@setfontsize\tiny\@viipt{8}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@viiipt{9.5}}
\newcommand*{\large}{\@setfontsize\large\@xiipt{14.5}}
\newcommand*{\Large}{\@setfontsize\Large\@xivpt{18}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xviipt{22}}
\newcommand*{\huge}{\@setfontsize\huge\@xxpt{25}}
\newcommand*{\Huge}{\@setfontsize\Huge\@xxvpt{30}}
\ifextrafontsizes
  \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxxpt{37}}
\else
  \let\HUGE=\Huge
\fi
⟨/11pt⟩
⟨*12pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@viipt{8}}
\newcommand*{\tiny}{\@setfontsize\tiny\@viiipt{9.5}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@ixpt\@xpt}
\newcommand*{\large}{\@setfontsize\large\@xivpt{18}}
\newcommand*{\Large}{\@setfontsize\Large\@xviipt{22}}
```

```
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xxpt{25}}
\newcommand*{\huge}{\@setfontsize\huge\@xxvpt{30}}
\ifextrafontsizes
  \newcommand*{\Huge}{\@setfontsize\Huge\@xxxpt{37}}
  \newcommand*{\HUGE}{\@setfontsize\HUGE\@xxxvipt{45}}
\else
  \let\Huge=\huge
  \let\HUGE=\huge
\fi
⟨/12pt⟩
⟨*14pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@viiipt{9.5}}
\newcommand*{\tiny}{\@setfontsize\tiny\@ixpt{10}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xpt{12}}
\newcommand*{\large}{\@setfontsize\large\@xviipt{22}}
\newcommand*{\Large}{\@setfontsize\Large\@xxpt{25}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xxvpt{30}}
\ifextrafontsizes
  \newcommand*{\huge}{\@setfontsize\Huge\@xxxpt{37}}
  \newcommand*{\Huge}{\@setfontsize\Huge\@xxxvipt{45}}
  \newcommand*{\HUGE}{\@setfontsize\HUGE\@xlviiipt{60}}
\else
  \let\huge=\LARGE
  \let\Huge=\LARGE
  \let\HUGE=\LARGE
\fi
⟨/14pt⟩
⟨*17pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@ixpt{10}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xpt{12}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xipt{13.6}}
\newcommand*{\large}{\@setfontsize\large\@xxpt{25}}
\newcommand*{\Large}{\@setfontsize\Large\@xxvpt{30}}
\ifextrafontsizes
  \newcommand*{\LARGE}{\@setfontsize\LARGE\@xxxpt{37}}
  \newcommand*{\huge}{\@setfontsize\huge\@xxxvipt{45}}
  \newcommand*{\Huge}{\@setfontsize\Huge\@xlviiipt{60}}
  \newcommand*{\HUGE}{\@setfontsize\HUGE\@lxpt{72}}
\else
  \let\LARGE=\Large
  \let\huge=\Large
  \let\Huge=\Large
  \let\HUGE=\Large
\fi
⟨/17pt⟩
⟨*20pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xpt{12}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xipt{13.6}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xiipt{14.5}}
\newcommand*{\large}{\@setfontsize\large\@xxvpt{30}}
```

```
\newcommand*{\Large}{\@setfontsize\Large\@xxxpt{37}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xxxvipt{45}}
\newcommand*{\huge}{\@setfontsize\huge\@xlviiipt{60}}
\newcommand*{\Huge}{\@setfontsize\Huge\@lxpt{72}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@lxxiipt{86}}
⟨/20pt⟩
⟨*25pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xipt{13.6}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xiipt{14.5}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xivpt{17.5}}
\newcommand*{\large}{\@setfontsize\large\@xxxpt{37}}
\newcommand*{\Large}{\@setfontsize\Large\@xxxvipt{45}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xlviiipt{60}}
\newcommand*{\huge}{\@setfontsize\huge\@lxpt{72}}
\newcommand*{\Huge}{\@setfontsize\Huge\@lxxiipt{86}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@lxxxivpt{100}}
⟨/25pt⟩
⟨*30pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xiipt{14.5}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xivpt{17.5}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xviipt{22}}
\newcommand*{\large}{\@setfontsize\large\@xxxvipt{45}}
\newcommand*{\Large}{\@setfontsize\Large\@xlviiipt{60}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@lxpt{72}}
\newcommand*{\huge}{\@setfontsize\huge\@lxxiipt{86}}
\newcommand*{\Huge}{\@setfontsize\Huge\@lxxxivpt{100}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@xcvipt{116}}
⟨/30pt⟩
⟨*36pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xivpt{17.5}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xviipt{22}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxpt{25}}
\newcommand*{\large}{\@setfontsize\large\@xlviiipt{60}}
\newcommand*{\Large}{\@setfontsize\Large\@lxpt{72}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@lxxiipt{86}}
\newcommand*{\huge}{\@setfontsize\huge\@lxxxivpt{100}}
\newcommand*{\Huge}{\@setfontsize\Huge\@xcvipt{116}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@cviiipt{132}}
⟨/36pt⟩
⟨*48pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xviipt{22}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xxpt{25}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxvpt{30}}
\newcommand*{\large}{\@setfontsize\large\@lxpt{72}}
\newcommand*{\Large}{\@setfontsize\Large\@lxxiipt{86}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@lxxxivpt{100}}
\newcommand*{\huge}{\@setfontsize\huge\@xcvipt{116}}
\newcommand*{\Huge}{\@setfontsize\Huge\@cviiipt{132}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@cxxpt{144}}
⟨/48pt⟩
```

```
⟨*60pt⟩
\newcommand*{\miniscule}{\@setfontsize\miniscule\@xxpt{25}}
\newcommand*{\tiny}{\@setfontsize\tiny\@xxvpt{30}}
\newcommand*{\scriptsize}{\@setfontsize\scriptsize\@xxxpt{37}}
\newcommand*{\large}{\@setfontsize\large\@lxxiipt{86}}
\newcommand*{\Large}{\@setfontsize\Large\@lxxxivpt{100}}
\newcommand*{\LARGE}{\@setfontsize\LARGE\@xcvipt{116}}
\newcommand*{\huge}{\@setfontsize\huge\@cviiipt{132}}
\newcommand*{\Huge}{\@setfontsize\Huge\@cxxpt{144}}
\newcommand*{\HUGE}{\@setfontsize\HUGE\@cxxxiipt{162}}
⟨/60pt⟩

⟨/9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
⟨*class⟩
```

`\captionsize` This internal command holds the font size for captions.

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

7.2 Paragraphing

`\lineskip` `\normallineskip` These parameters control TEX's behaviour when two lines tend to come too close together.

```
\setlength\lineskip{1\p@}
\setlength\normallineskip{1\p@}
```

`\baselinestretch` This is used as a multiplier for `\baselineskip`. The default is to *not* stretch the baselines.

```
\renewcommand{\baselinestretch}{}
```

`\parskip` `\parskip` is additional vertical space between paragraphs; default is zero.
`\onelineskip` `\onelineskip` is the default space between baselines.

```
\setlength\parskip{0\p@ \@plus \p@}
⟨/class⟩
⟨*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
⟨9pt⟩\setlength{\onelineskip}{\@xpt\p@}
⟨10pt⟩\setlength{\onelineskip}{\@xiipt\p@}
⟨11pt⟩\setlength{\onelineskip}{13.6\p@}
⟨12pt⟩\setlength{\onelineskip}{14.5\p@}
⟨14pt⟩\setlength{\onelineskip}{17.5\p@}
⟨17pt⟩\setlength{\onelineskip}{22\p@}
⟨20pt⟩\setlength{\onelineskip}{25\p@}
⟨25pt⟩\setlength{\onelineskip}{30\p@}
⟨30pt⟩\setlength{\onelineskip}{37\p@}
⟨36pt⟩\setlength{\onelineskip}{45\p@}
⟨48pt⟩\setlength{\onelineskip}{60\p@}
⟨60pt⟩\setlength{\onelineskip}{72\p@}
```

`\parindent` `\parskip` gives extra vertical space between paragraphs and `\parindent` is the width of the paragraph indentation.

```
\if@twocolumn
  \setlength\parindent{1em}
\else
⟨9pt⟩  \setlength\parindent{12\p@}
⟨10pt⟩  \setlength\parindent{15\p@}
⟨11pt⟩  \setlength\parindent{17\p@}
⟨12pt⟩  \setlength\parindent{1.5em}
⟨14pt⟩  \setlength\parindent{1.5em}
⟨17pt⟩  \setlength\parindent{1.5em}
⟨20pt⟩  \setlength\parindent{1.5em}
⟨25pt⟩  \setlength\parindent{1.5em}
⟨30pt⟩  \setlength\parindent{1.5em}
⟨36pt⟩  \setlength\parindent{1.5em}
⟨48pt⟩  \setlength\parindent{1.5em}
⟨60pt⟩  \setlength\parindent{1.5em}
\fi
⟨/9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
```

\smallskipamount
\medskipamount
\bigskipamount

The values for these three parameters are set in the LaTeX kernel. Historically they have been size invariant, but I have changed them for the larger sizes (\smallskipamount is 1/4 the fontsize, and the others each double up on the next lower).

```
⟨*9pt | 10pt | 11pt | 12pt | 14pt⟩
\setlength\smallskipamount{3\p@ \@plus 1\p@ \@minus 1\p@}
\setlength\medskipamount{6\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\bigskipamount{12\p@ \@plus 4\p@ \@minus 4\p@}
⟨/9pt | 10pt | 11pt | 12pt | 14pt⟩
⟨*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
⟨*17pt⟩
\setlength\smallskipamount{4\p@ \@plus1\p@ \@minus1\p@}
\setlength\medskipamount{8\p@ \@plus2\p@ \@minus2\p@}
\setlength\bigskipamount{17\p@ \@plus4\p@ \@minus4\p@}
⟨/17pt⟩
⟨*20pt⟩
\setlength\smallskipamount{5\p@ \@plus1\p@ \@minus1\p@}
\setlength\medskipamount{10\p@ \@plus2\p@ \@minus2\p@}
\setlength\bigskipamount{20\p@ \@plus4\p@ \@minus4\p@}
⟨/20pt⟩
⟨*25pt⟩
\setlength\smallskipamount{6\p@ \@plus1\p@ \@minus1\p@}
\setlength\medskipamount{12\p@ \@plus2\p@ \@minus2\p@}
\setlength\bigskipamount{24\p@ \@plus4\p@ \@minus4\p@}
⟨/25pt⟩
⟨*30pt⟩
\setlength\smallskipamount{5\p@ \@plus1\p@ \@minus1\p@}
\setlength\medskipamount{10\p@ \@plus2\p@ \@minus2\p@}
\setlength\bigskipamount{20\p@ \@plus4\p@ \@minus4\p@}
⟨/30pt⟩
⟨*36pt⟩
\setlength\smallskipamount{9\p@ \@plus2\p@ \@minus2\p@}
```

```
\setlength\medskipamount{18\p@ \@plus4\p@ \@minus4\p@}
\setlength\bigskipamount{36\p@ \@plus8\p@ \@minus8\p@}
⟨/36pt⟩
⟨*48pt⟩
\setlength\smallskipamount{12\p@ \@plus3\p@ \@minus3\p@}
\setlength\medskipamount{24\p@ \@plus6\p@ \@minus6\p@}
\setlength\bigskipamount{48\p@ \@plus12\p@ \@minus12\p@}
⟨/48pt⟩
⟨*60pt⟩
\setlength\smallskipamount{15\p@ \@plus4\p@ \@minus4\p@}
\setlength\medskipamount{30\p@ \@plus8\p@ \@minus8\p@}
\setlength\bigskipamount{60\p@ \@plus16\p@ \@minus16\p@}
⟨/60pt⟩
⟨/17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
```

`\@lowpenalty` `\@medpenalty` `\@highpenalty` The commands `\nopagebreak` and `\nolinebreak` put in penalties to discourage these breaks at the point they are put in. They use `\@lowpenalty`, `\@medpenalty` or `\@highpenalty`, dependent on their argument.

```
⟨*class⟩
\@lowpenalty   51
\@medpenalty  151
\@highpenalty 301
```

`\clubpenalty` `\widowpenalty` These penalties are used to discourage club and widow lines. The default values are 150 each, but we want stronger discouragement.

```
\clubpenalty  1000
\widowpenalty 1000
```

`\displaywidowpenalty` `\predisplaypenalty` `\postdisplaypenalty` Discourage, but do not prevent, widows in front of a math display and forbid breaking directly in front of a display. Allow break after a display without a penalty. The default values are used, therefore we only show them here.

```
% \displaywidowpenalty 50
% \predisplaypenalty   10000
% \postdisplaypenalty  0
```

`\interlinepenalty` Allow the breaking of a page in the middle of a paragraph.

```
% \interlinepenalty 0
```

`\brokenpenalty` We allow the breaking of a page after a hyphenated line.

```
% \brokenpenalty 100
```

7.3 Vertical spacing

`\headheight` `\headsep` `\topskip` `\footskip` The `\headheight` is the height of the box that will contain the running head. In this class it is point size dependent — `\onelineskip` (normally it is a constant 12pt).

The `\headsep` is the distance between the bottom of the running head and the top of the text. For the larger sizes this is 1.8 times the fontsize.

The `\topskip` is the `\baselineskip` for the first line on a page; LaTeX's output routine will not work properly if it has the value 0pt, so do not do that! For the lareger sizes this is the font size.
The distance from the baseline of the box which contains the running footer to the baseline of last line of text is controlled by the `\footskip`. For the larger sizes this is 2.5 times the font size.

```
⟨/class⟩
⟨*9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
⟨*9pt⟩
\setlength\headheight{11\p@}
\setlength\headsep{.225in}
\setlength\topskip{9\p@}
\setlength\footskip{.33in}
⟨/9pt⟩
⟨*10pt⟩
\setlength\headheight{12\p@}
\setlength\headsep{.25in}
\setlength\topskip{10\p@}
\setlength\footskip{.35in}
⟨/10pt⟩
⟨*11pt⟩
\setlength\headheight{13.6\p@}
\setlength\headsep{.275in}
\setlength\topskip{11\p@}
\setlength\footskip{.38in}
⟨/11pt⟩
⟨*12pt⟩
\setlength\headheight{14\p@}
\setlength\headsep{.275in}
\setlength\topskip{12\p@}
\setlength\footskip{30\p@}
⟨/12pt⟩
⟨*14pt⟩
\setlength\headheight{17.5\p@}
\setlength\headsep{.30in}
\setlength\topskip{14.4\p@}
\setlength\footskip{.4in}
⟨/14pt⟩
⟨*17pt⟩
\setlength\headheight{22\p@}
\setlength\headsep{.30in}
\setlength\topskip{14.4\p@}
\setlength\footskip{.4in}
⟨/17pt⟩
⟨*20pt⟩
\setlength\headheight{25\p@}
\setlength\headsep{36\p@}
\setlength\topskip{20\p@}
\setlength\footskip{50\p@}
```

```
⟨/20pt⟩
⟨*25pt⟩
\setlength\headheight{30\p@}
\setlength\headsep{45\p@}
\setlength\topskip{25\p@}
\setlength\footskip{62\p@}
⟨/25pt⟩
⟨*30pt⟩
\setlength\headheight{37\p@}
\setlength\headsep{54\p@}
\setlength\topskip{30\p@}
\setlength\footskip{75\p@}
⟨/30pt⟩
⟨*36pt⟩
\setlength\headheight{45\p@}
\setlength\headsep{65\p@}
\setlength\topskip{36\p@}
\setlength\footskip{90\p@}
⟨/36pt⟩
⟨*48pt⟩
\setlength\headheight{60\p@}
\setlength\headsep{86\p@}
\setlength\topskip{48\p@}
\setlength\footskip{120\p@}
⟨/48pt⟩
⟨*60pt⟩
\setlength\headheight{72\p@}
\setlength\headsep{108\p@}
\setlength\topskip{60\p@}
\setlength\footskip{150\p@}
⟨/60pt⟩
```

`\maxdepth` `\@maxdepth`

The TEX primitive register `\maxdepth` has a function that is similar to that of `\topskip`. The register `\@maxdepth` should always contain a copy of `\maxdepth`. In both plain TEX and LATEX 2.09 `\maxdepth` had a fixed value of `4pt`; in native LATEX2e mode we let the value depend on the typesize. We set it so that `\maxdepth` + `\topskip` = typesize ×1.5. As it happens, in these classes `\topskip` is equal to the typesize, therefor we set `\maxdepth` to half the value of `\topskip`.

```
\setlength\maxdepth{.5\topskip}
\setlength\@maxdepth\maxdepth
```

7.4 Footnotes

`\footnotesep`

`\footnotesep` is the height of the strut placed at the beginning of every footnote. It equals the height of a normal `\footnotesize` strut, so no extra space appears between footnotes.

```
⟨9pt⟩\setlength\footnotesep{5.6\p@}
```

```
⟨10pt⟩\setlength\footnotesep{6.65\p@}
⟨11pt⟩\setlength\footnotesep{7.7\p@}
⟨12pt⟩\setlength\footnotesep{8.4\p@}
⟨14pt⟩\setlength\footnotesep{9.5\p@}
⟨17pt⟩\setlength\footnotesep{10.15\p@}
⟨20pt⟩\setlength\footnotesep{12.6\p@}
⟨25pt⟩\setlength\footnotesep{15.4\p@}
⟨30pt⟩\setlength\footnotesep{17.5\p@}
⟨36pt⟩\setlength\footnotesep{21\p@}
⟨48pt⟩\setlength\footnotesep{25.9\p@}
⟨60pt⟩\setlength\footnotesep{31.5\p@}
```

`\footins` `\skip\footins` is the space between the last line of the main text and the top of the first footnote.

```
⟨9pt⟩\setlength{\skip\footins}{8\p@ \@plus 2\p@ \@minus 2\p@}
⟨10pt⟩\setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
⟨11pt⟩\setlength{\skip\footins}{10\p@ \@plus 4\p@ \@minus 2\p@}
⟨12pt⟩\setlength{\skip\footins}{10.8\p@ \@plus 4\p@ \@minus 2\p@}
⟨14pt⟩\setlength{\skip\footins}{13\p@ \@plus 4\p@ \@minus 2\p@}
⟨17pt⟩\setlength{\skip\footins}{16\p@ \@plus 5\p@ \@minus 3\p@}
⟨20pt⟩\setlength{\skip\footins}{19\p@ \@plus 7\p@ \@minus 3\p@}
⟨25pt⟩\setlength{\skip\footins}{24\p@ \@plus 8\p@ \@minus 4\p@}
⟨30pt⟩\setlength{\skip\footins}{29\p@ \@plus 10\p@ \@minus 5\p@}
⟨36pt⟩\setlength{\skip\footins}{35\p@ \@plus 12\p@ \@minus 6\p@}
⟨48pt⟩\setlength{\skip\footins}{47\p@ \@plus 16\p@ \@minus 8\p@}
⟨60pt⟩\setlength{\skip\footins}{59\p@ \@plus 20\p@ \@minus 10\p@}
```

7.5 Floats

Floats on a text page

`\floatsep` `\textfloatsep` `\intextsep` When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode.

`\floatsep` is the space between adjacent floats that are moved to the top or bottom of the text page. For the larger sizes this is `\bigskip`.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page. For the larger sizes this is 1.45`\onelineskip`.

`\intextsep` is the space between in-text floats and the text.

```
⟨*9pt⟩
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
⟨/9pt⟩
⟨*10pt⟩
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
```

```
⟨/10pt⟩
⟨*11pt⟩
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
⟨/11pt⟩
⟨*12pt⟩
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {14\p@ \@plus 4\p@ \@minus 4\p@}
⟨/12pt⟩
⟨*14pt⟩
\setlength\floatsep    {14\p@ \@plus 4\p@ \@minus 4\p@}
\setlength\textfloatsep{20\p@ \@plus 4\p@ \@minus 4\p@}
\setlength\intextsep   {14\p@ \@plus 4\p@ \@minus 4\p@}
⟨/14pt⟩
⟨*17pt⟩
\setlength\floatsep    {15\p@ \@plus 4\p@ \@minus 4\p@}
\setlength\textfloatsep{25\p@ \@plus 4\p@ \@minus 5\p@}
\intextsep \floatsep
⟨/17pt⟩
⟨*20pt⟩
\setlength\floatsep    {20\p@ \@plus 4\p@ \@minus 4\p@}
\setlength\textfloatsep{36\p@ \@plus 4\p@ \@minus 8\p@}
\intextsep \floatsep
⟨/20pt⟩
⟨*25pt⟩
\setlength\floatsep    {24\p@ \@plus 4\p@ \@minus 4\p@}
\setlength\textfloatsep{43\p@ \@plus 4\p@ \@minus 8\p@}
\intextsep \floatsep
⟨/25pt⟩
⟨*30pt⟩
\setlength\floatsep    {30\p@ \@plus 6\p@ \@minus 6\p@}
\setlength\textfloatsep{54\p@ \@plus 6\p@ \@minus 12\p@}
\intextsep \floatsep
⟨/30pt⟩
⟨*36pt⟩
\setlength\floatsep    {36\p@ \@plus 8\p@ \@minus 8\p@}
\setlength\textfloatsep{65\p@ \@plus 8\p@ \@minus 16\p@}
\intextsep \floatsep
⟨/36pt⟩
⟨*48pt⟩
\setlength\floatsep    {48\p@ \@plus 12\p@ \@minus 12\p@}
\setlength\textfloatsep{87\p@ \@plus 12\p@ \@minus 24\p@}
\intextsep \floatsep
⟨/48pt⟩
⟨*60pt⟩
\setlength\floatsep    {60\p@ \@plus 16\p@ \@minus 16\p@}
\setlength\textfloatsep{104\p@ \@plus 16\p@ \@minus 32\p@}
\intextsep \floatsep
```

```
1536 ⟨/60pt⟩
```

\dblfloatsep
\dbltextfloatsep

When floating objects that span the whole \textwidth are placed on a text page and LaTeX is in twocolumn mode the separation between the float and the text is controlled by \dblfloatsep and \dbltextfloatsep.
\dblfloatsep is the space between adjacent floats that are moved to the top or bottom of the text page.
\dbltextfloatsep is the space between the main text and floats at the top or bottom of the page.

```
1537 ⟨*9pt⟩
1538 \setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
1539 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1540 ⟨/9pt⟩
1541 ⟨*10pt⟩
1542 \setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
1543 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1544 ⟨/10pt⟩
1545 ⟨*11pt⟩
1546 \setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
1547 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1548 ⟨/11pt⟩
1549 ⟨*12pt⟩
1550 \setlength\dblfloatsep    {14\p@ \@plus 2\p@ \@minus 4\p@}
1551 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1552 ⟨/12pt⟩
1553 ⟨*14pt⟩
1554 \setlength\dblfloatsep    {14\p@ \@plus 2\p@ \@minus 4\p@}
1555 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1556 ⟨/14pt⟩
1557 ⟨*17pt⟩
1558 \setlength\dblfloatsep    {15\p@ \@plus 4\p@ \@minus 4\p@}
1559 \setlength\dbltextfloatsep{25\p@ \@plus 5\p@ \@minus 5\p@}
1560 ⟨/17pt⟩
1561 ⟨*20pt⟩
1562 \dblfloatsep \floatsep
1563 \dbltextfloatsep \textfloatsep
1564 ⟨/20pt⟩
1565 ⟨*25pt⟩
1566 \dblfloatsep \floatsep
1567 \dbltextfloatsep \textfloatsep
1568 ⟨/25pt⟩
1569 ⟨*30pt⟩
1570 \dblfloatsep \floatsep
1571 \dbltextfloatsep \textfloatsep
1572 ⟨/30pt⟩
1573 ⟨*36pt⟩
1574 \dblfloatsep \floatsep
1575 \dbltextfloatsep \textfloatsep
1576 ⟨/36pt⟩
```

```
⟨*48pt⟩
\dblfloatsep \floatsep
\dbltextfloatsep \textfloatsep
⟨/48pt⟩
⟨*60pt⟩
\dblfloatsep \floatsep
\dbltextfloatsep \textfloatsep
⟨/60pt⟩
```

Floats on their own page or column

`\@fptop` `\@fpsep` `\@fpbot`

When floating objects are placed on separate pages the layout of such pages is controlled by these parameters. At the top of the page `\@fptop` amount of stretchable whitespace is inserted, at the bottom of the page we get an `\@fpbot` amount of stretchable whitespace. Between adjacent floats the `\@fpsep` is inserted. For the larger sizes `\@fpsep` is .7`\onelineskip`.

These parameters are used for the placement of floating objects in one column mode, or in single column floats in two column mode.

Note that at least one of the two parameters `\@fptop` and `\@fpbot` should contain a `plus ...fil` to allow filling the remaining empty space.

```
⟨*9pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{9\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/9pt⟩
⟨*10pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/10pt⟩
⟨*11pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/11pt⟩
⟨*12pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{10\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/12pt⟩
⟨*14pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{10\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/14pt⟩
⟨*17pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{12\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
```

```
⟨/17pt⟩
⟨*20pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{17\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/20pt⟩
⟨*25pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{21\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/25pt⟩
⟨*30pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{26\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/30pt⟩
⟨*36pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{31\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/36pt⟩
⟨*48pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{42\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/48pt⟩
⟨*60pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{50\p@ \@plus 2fil}
\@fpbot \@fptop
⟨/60pt⟩
```

`\@dblfptop` `\@dblfpsep` `\@dblfpbot` Double column floats in two column mode are handled with similar parameters.

```
⟨*9pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{7\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/9pt⟩
⟨*10pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/10pt⟩
⟨*11pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/11pt⟩
⟨*12pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
```

```
\setlength\@dblfpsep{10\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/12pt⟩
⟨*14pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{12\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/14pt⟩
⟨*17pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{12\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/17pt⟩
⟨*20pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/20pt⟩
⟨*25pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/25pt⟩
⟨*30pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/30pt⟩
⟨*36pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/36pt⟩
⟨*48pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/48pt⟩
⟨*60pt⟩
\@dblfptop \@fptop
\@dblfpsep \@fpsep
\@dblfpbot \@dblfptop
⟨/60pt⟩
```

7.6 The measure

The width of a line of text (and therefore the text block) is termed the *measure*.

`\lxvchars` The length `\lxvchars` is the approximate length of a normal text line containing 65 characters (a typesetters rule of thumb is that there should be about 60–70

characters per line).

```
⟨9pt⟩\setlength\lxvchars{276\p@}  %
⟨10pt⟩\setlength\lxvchars{300\p@} % standard 345pt
⟨11pt⟩\setlength\lxvchars{324\p@} % standard 360pt
⟨12pt⟩\setlength\lxvchars{336\p@} % standard 390pt
⟨14pt⟩\setlength\lxvchars{408\p@} %
⟨17pt⟩\setlength\lxvchars{444\p@} %
⟨20pt⟩\setlength\lxvchars{528\p@} %
⟨25pt⟩\setlength\lxvchars{626\p@} %
⟨30pt⟩\setlength\lxvchars{748\p@} %
⟨36pt⟩\setlength\lxvchars{891\p@} %
⟨48pt⟩\setlength\lxvchars{1177\p@} %
⟨60pt⟩\setlength\lxvchars{1463\p@} %
```

`\xlvchars` The length `\xlvchars` is the approximate length of a normal double column text line containing 45 characters (a typesetters rule of thumb is that there should be about 40–50 characters per column line).

```
⟨9pt⟩\setlength\xlvchars{192\p@}  %
⟨10pt⟩\setlength\xlvchars{204\p@} %
⟨11pt⟩\setlength\xlvchars{216\p@} %
⟨12pt⟩\setlength\xlvchars{240\p@} %
⟨14pt⟩\setlength\xlvchars{288\p@} %
⟨17pt⟩\setlength\xlvchars{312\p@} %
⟨20pt⟩\setlength\xlvchars{365\p@} %
⟨25pt⟩\setlength\xlvchars{438\p@} %
⟨30pt⟩\setlength\xlvchars{518\p@} %
⟨36pt⟩\setlength\xlvchars{617\p@} %
⟨48pt⟩\setlength\xlvchars{815\p@} %
⟨60pt⟩\setlength\xlvchars{1014\p@} %
```

`\marginparsep` `\marginparsep` is the horizontal space between the text block and marginal
`\marginparpush` notes, while `\marginparpush` is the minimum vertical separation between the notes.

```
\if@twocolumn
⟨*9pt | 10pt | 11pt | 12pt | 14pt⟩
   \setlength\marginparsep{10\p@}
⟨/9pt | 10pt | 11pt | 12pt | 14pt⟩
⟨*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
   \setlength\marginparsep{1em}
⟨/17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
\else
⟨*9pt | 10pt | 11pt | 12pt | 14pt⟩
   \setlength\marginparsep{7\p@}
⟨/9pt | 10pt | 11pt | 12pt | 14pt⟩
⟨*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
   \setlength\marginparsep{0.7em}
⟨/17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
\fi
⟨9pt | 10pt | 11pt⟩\setlength{\marginparpush}{5\p@}
```

```
⟨12pt | 14pt⟩\setlength{\marginparpush}{7\p@}
⟨*17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
  \setlength\marginparpush{0.5em}
⟨/17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩

⟨/9pt | 10pt | 11pt | 12pt | 14pt | 17pt | 20pt | 25pt | 30pt | 36pt | 48pt | 60pt⟩
⟨*class⟩
```

8 Page Layout

8.1 The typeblock and margins

All margin dimensions are measured from a point one inch from the top and lefthand side of the page.

`\setlxvchars` `\setxlvchars`

These were suggested and supplied by Morten Høgholm (18 May 2003).
`\setlxvchars[`⟨*fontspec*⟩`]` sets `\lxvchars` to the length of a line containing 65 characters in the ⟨*fontspec*⟩.
Similarly `\setxlvchars[`⟨*fontspec*⟩`]` sets `\xlvchars` for 45 characters.

```
\newcommand*{\setlxvchars}[1][\normalfont]{\begingroup
  #1
  \settowidth{\lxvchars}{abcdefghijklmnopqrstuvwxyz}%
  \setlength{\lxvchars}{2.042\lxvchars}%
  \addtolength{\lxvchars}{33.41pt}%
  \global\lxvchars=\lxvchars
  \endgroup}
\newcommand*{\setxlvchars}[1][\normalfont]{\begingroup
  #1
  \settowidth{\xlvchars}{abcdefghijklmnopqrstuvwxyz}%
  \setlength{\xlvchars}{1.415\xlvchars}%
  \addtolength{\xlvchars}{23.03pt}%
  \global\xlvchars=\xlvchars
  \endgroup}

```

`\setrectanglesize`

The macro `\setrectanglesize{`⟨*H*⟩`}{`⟨*W*⟩`}{`⟨*r*⟩`}` calculates the height and width of a rectangle given any two out of the three arguments. An unvalued argument is denoted by `*`.
Table 2 shows the argument combinations and the result for each combination. The calculated height and width are stored in `\@tempdima` and `\@tempdimb` respectively. Both lengths are set to zero if there is an error.

```
\newcommand*{\setrectanglesize}[3]{%
  \nametest{#1}{*}%
  \ifsamename                           % H = *
    \nametest{#2}{*}%
    \ifsamename                         % W = *
      \@mememerror{%
        The combination of argument values is ambiguous.\MessageBreak
```

Table 2: Arguments and results for \setrectanglesize

H	W	r	Result
*	W	r	$H = rW$
*	W	*	$H = W$
*	*	r	ambiguous
*	*	*	ambiguous
H	W	r	H, W
H	W	*	H, W
H	*	r	$W = rH$
H	*	*	$W = H$

```
1773        The lengths will be set to zero}{\@ehd}%
1774      \setlength{\@tempdima}{0pt}%
1775      \setlength{\@tempdimb}{0pt}%
1776    \else                                % W
1777      \nametest{#3}{*}%
1778      \ifsamename                        % r = *
1779        \setlength{\@tempdimb}{#2}%
1780        \setlength{\@tempdima}{\@tempdimb}%
1781      \else                              % r
1782        \setlength{\@tempdimb}{#2}%
1783        \setlength{\@tempdima}{#3\@tempdimb}%
1784      \fi
1785    \fi
1786  \else                                  % H
1787    \nametest{#2}{*}%
1788    \ifsamename                          % W = *
1789      \nametest{#3}{*}%
1790      \ifsamename                        % r = *
1791        \setlength{\@tempdima}{#1}%
1792        \setlength{\@tempdimb}{\@tempdima}%
1793      \else                              % r
1794        \setlength{\@tempdima}{#1}%
1795        \setlength{\@tempdimb}{#3\@tempdima}%
1796      \fi
1797    \else                                % W
1798      \setlength{\@tempdima}{#1}%
1799      \setlength{\@tempdimb}{#2}%
1800    \fi
1801  \fi}
1802
```

\setfillsize Consider a set of 4 lengths, T, L, C, and R, such that $T = L + C + R$, where T is a fixed size and normally C is also fixed. Also L and R may be specified independently of each other or as a ratio (i.e., $L = rR$ or $R = rL$). The macro \setfillsize{⟨T⟩}{⟨C⟩}{⟨L⟩}{⟨R⟩}{⟨r⟩} maintains these constraints among the variables, where an unvalued argument is denoted by *.

Table 3: Arguments and results for `\setfillsize`

C	L	R	r	Result	
*	*	R	r	$L = rR$,	$C = T - L - R$
*	*	R	*	$L = R$,	$C = T - L - R$
*	*	*	r	ambigous	
*	*	*	*	ambiguous	
*	L	R	r	L, R,	$C = T - L - R$
*	L	R	*	L, R,	$C = T - L - R$
*	L	*	r	$R = rL$,	$C = T - L - R$
*	L	*	*	$R = L$,	$C = T - L - R$
C	*	R	r	$L = T - C - R$	C
C	*	R	*	$L = T - C - R$,	C
C	*	*	r	$L + R = T - C$, $R = rL$,	C
C	*	*	*	$L + R = T - C$, $R = L$,	C
C	L	R	r	ambiguous	C
C	L	R	*	ambiguous	C
C	L	*	r	$R = T - C - L$,	C
C	L	*	*	$R = T - C - L$,	C

Table 3 shows the argument combinations and the result for each combination. The calculated values of `C`, `L` and `R` are stored `\@tempdimc`, `\@tempdima` and `\@tempdimb` respectively. If there is an error the lengths are set to zero.

```
\newcommand*{\setfillsize}[5]{%
  \nametest{#2}{*}%
  \ifsamename                                   % C = *
    \nametest{#3}{*}%
    \ifsamename                                 % L = *
      \nametest{#4}{*}%
      \ifsamename                               % R = *
        \@memerror{%
          The combination of argument values is ambiguous.\MessageBreak
          The lengths will be set to zero}{\@ehd}
        \setlength{\@tempdima}{0pt}%
        \setlength{\@tempdimb}{0pt}%
        \setlength{\@tempdimc}{0pt}%
      \else                                     % R
        \nametest{#5}{*}%
        \ifsamename                             % r = *
          \setlength{\@tempdimb}{#4}%
          \setlength{\@tempdima}{\@tempdimb}%
          \setlength{\@tempdimc}{#1}%
          \advance\@tempdimc -\@tempdima
          \advance\@tempdimc -\@tempdimb
        \else                                   % r
          \setlength{\@tempdimb}{#4}%
          \setlength{\@tempdima}{#5\@tempdimb}%
```

```
1827          \setlength{\@tempdimc}{#1}%
1828          \advance\@tempdimc -\@tempdima
1829          \advance\@tempdimc -\@tempdimb
1830        \fi
1831      \fi
1832    \else                                   % L
1833      \nametest{#4}{*}%
1834      \ifsamename                           % R = *
1835        \nametest{#5}{*}%
1836        \ifsamename                         % r = *
1837          \setlength{\@tempdima}{#3}%
1838          \setlength{\@tempdimb}{\@tempdima}
1839          \setlength{\@tempdimc}{#1}%
1840          \advance\@tempdimc -\@tempdima
1841          \advance\@tempdimc -\@tempdimb
1842        \else                               % r
1843          \setlength{\@tempdima}{#3}%
1844          \setlength{\@tempdimb}{#5\@tempdima}
1845          \setlength{\@tempdimc}{#1}%
1846          \advance\@tempdimc -\@tempdima
1847          \advance\@tempdimc -\@tempdimb
1848        \fi
1849      \else                                 % R
1850        \setlength{\@tempdima}{#3}%
1851        \setlength{\@tempdimb}{#4}%
1852        \setlength{\@tempdimc}{#1}%
1853        \advance\@tempdimc -\@tempdima
1854        \advance\@tempdimc -\@tempdimb
1855      \fi
1856    \fi
1857  \else                                     % C is valued
1858    \nametest{#3}{*}%
1859    \ifsamename                             % L = *
1860      \nametest{#4}{*}%
1861      \ifsamename                           % R = *
1862        \nametest{#5}{*}%
1863        \ifsamename                         % r = *
1864          \setlength{\@tempdimc}{#2}%
1865          \setlength{\@tempdima}{#1}%
1866          \advance\@tempdima -\@tempdimc
1867          \@tempdima = 0.5\@tempdima
1868          \@tempdimb = \@tempdima
1869        \else                               % r (CODE PERHAPS FIXED)
1870          \setlength{\@tempdimc}{#2}        % C
1871          \setlength{\@tempdimb}{#1}        % T
1872          \advance\@tempdimb -\@tempdimc % T - C
1873          \@tempdima = 1000sp
1874          \setlength{\@tempdima}{#5\@tempdima}        % 1000r sp
1875          \advance\@tempdima by 1000sp   % 1000(1+r)sp
1876          \@tempcnta = \@tempdima        % 1000(1+r)
```

```
          \@tempdima = \@tempdimb        % T - C
          \divide\@tempdima by \@tempcnta  % (T-C)/1000(1+r) pts
          \@tempdima = 1000\@tempdima      % (T-C)/(1+r)  pts = L
          \advance\@tempdimb by -\@tempdima % = R
        \fi
      \else                                % R
        \setlength{\@tempdimc}{#2}%
        \setlength{\@tempdimb}{#4}%
        \setlength{\@tempdima}{#1}%
        \advance\@tempdima -\@tempdimc
        \advance\@tempdima -\@tempdimb
      \fi
    \else                                  % L
      \nametest{#4}{*}%
      \ifsamename                          % R = *
        \setlength{\@tempdimc}{#2}%
        \setlength{\@tempdima}{#3}%
        \setlength{\@tempdimb}{#1}%
        \advance\@tempdimb -\@tempdimc
        \advance\@tempdimb -\@tempdima
      \else                                % R
        \@memerror{%
          The combination of argument values is ambiguous.\MessageBreak
          The lengths will be set to zero}{\@ehd}%
        \setlength{\@tempdima}{0pt}%
        \setlength{\@tempdimb}{0pt}%
        \setlength{\@tempdimc}{#2}%
      \fi
    \fi
  \fi}

```

`\setstocksize` `\settrims` `\settrimmedsize`

`\setstocksize{`⟨*height*⟩`}{`⟨*width*⟩`}` sets the height and width of the stock material and `\settrims{`⟨*top*⟩`}{`⟨*edge*⟩`}` sets the trim lengths for the top and side (fore edge) of the stock. The macro `\settrimmedsize{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` sets the size for the trimmed page, based on `\setrectanglesize`.

```
\newcommand{\setstocksize}[2]{%
  \setlength{\stockheight}{#1}%
  \setlength{\stockwidth}{#2}}
\newcommand{\settrims}[2]{%
  \setlength{\trimtop}{#1}%
  \setlength{\trimedge}{#2}}
\newcommand{\settrimmedsize}[3]{%
  \setrectanglesize{#1}{#2}{#3}%
  \setlength{\paperheight}{\@tempdima}%
  \setlength{\paperwidth}{\@tempdimb}}

```

`\settypeblocksize` `\settypeblocksize{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` calulates the `\textheight` and

\textwidth from two out of the three arguments.

```
\newcommand{\settypeblocksize}[3]{%
  \setrectanglesize{#1}{#2}{#3}%
  \setlength{\textheight}{\@tempdima}%
  \setlength{\textwidth}{\@tempdimb}}
```

\binding The length \binding is an allowance on the spine margin for binding.
\setbinding \setbinding{⟨*length*⟩} sets the \binding.

```
\newlength{\binding}
\newcommand*{\setbinding}[1]{\setlength{\binding}{#1}}
  \setbinding{0pt}
```

\spinemargin \setlrmargins{⟨*L*⟩}{⟨*R*⟩}{⟨*r*⟩} sets the Left (spine) and Right (fore edge)
\foremargin margins with constant typeblock.
\setlrmargins

```
\newlength{\spinemargin}
\newlength{\foremargin}
\newcommand{\setlrmargins}[3]{%
  \advance\paperwidth -\binding
  \setfillsize{\paperwidth}{\textwidth}{#1}{#2}{#3}%
  \setlength{\textwidth}{\@tempdimc}%
  \setlength{\spinemargin}{\@tempdima}%
  \setlength{\foremargin}{\@tempdimb}%
  \advance\paperwidth \binding
  \advance\spinemargin \binding}
```

\setlrmarginsandblock \setlrmarginsandblock{⟨*L*⟩}{⟨*R*⟩}{⟨*r*⟩} sets the Left (spine) and Right (fore edge) margins with variable typeblock.

```
\newcommand{\setlrmarginsandblock}[3]{%
  \advance\paperwidth -\binding
  \setfillsize{\paperwidth}{*}{#1}{#2}{#3}%
  \setlength{\textwidth}{\@tempdimc}%
  \setlength{\spinemargin}{\@tempdima}%
  \setlength{\foremargin}{\@tempdimb}%
  \advance\paperwidth \binding
  \advance\spinemargin \binding}
```

\uppermargin \setulmargins{⟨*L*⟩}{⟨*R*⟩}{⟨*r*⟩} sets the Left (upper) and Right (lower) margins
\lowermargin with constant typeblock.
\setulmargins

```
\newlength{\uppermargin}
\newlength{\lowermargin}
\newcommand{\setulmargins}[3]{%
  \setfillsize{\paperheight}{\textheight}{#1}{#2}{#3}%
  \setlength{\textheight}{\@tempdimc}%
  \setlength{\uppermargin}{\@tempdima}%
  \setlength{\lowermargin}{\@tempdimb}}
```

```
1955
```

`\setulmarginsandblock` `\setulmarginsandblock{⟨L⟩}{⟨R⟩}{⟨r⟩}` sets the Left (upper) and Right (lower) margins with variable typeblock.

```
1956 \newcommand{\setulmarginsandblock}[3]{%
1957   \setfillsize{\paperheight}{*}{#1}{#2}{#3}%
1958   \setlength{\textheight}{\@tempdimc}%
1959   \setlength{\uppermargin}{\@tempdima}%
1960   \setlength{\lowermargin}{\@tempdimb}}
1961
```

`\headdrop` `\setheaderspaces` `\setheaderspaces{⟨L⟩}{⟨R⟩}{⟨r⟩}` sets the Left (head margin) and Right (headsep) spacing with constant headheight.

```
1962 \newlength{\headdrop}
1963 \newcommand{\setheaderspaces}[3]{%
1964   \setfillsize{\uppermargin}{\headheight}{#1}{#2}{#3}%
1965   \setlength{\headheight}{\@tempdimc}%
1966   \setlength{\headdrop}{\@tempdima}%
1967   \setlength{\headsep}{\@tempdimb}}
1968
```

`\setheadfoot` `\setheadfoot{⟨headheight⟩}{⟨footskip⟩}` sets the headheight and the footskip.

```
1969 \newcommand{\setheadfoot}[2]{%
1970   \setlength{\headheight}{#1}%
1971   \setlength{\footskip}{#2}}
1972
```

`\setcolsepandrule` `\setcolsepandrule{⟨colsep⟩}{⟨thickness⟩}` sets the column separation and the rule thickness.

```
1973 \newcommand{\setcolsepandrule}[2]{%
1974   \setlength{\columnsep}{#1}%
1975   \setlength{\columnseprule}{#2}}
1976
```

`\setmarginnotes` `\setmarginnotes{⟨sep⟩}{⟨width⟩}{⟨push⟩}` sets the marginpar parameters.

```
1977 \newcommand{\setmarginnotes}[3]{%
1978   \setlength{\marginparsep}{#1}%
1979   \setlength{\marginparwidth}{#2}%
1980   \setlength{\marginparpush}{#3}}
1981
```

Initialise the paper size and trimming to their default values.

```
1982 \settrimmedsize{\stockheight}{\stockwidth}{*}
1983 \settrims{\z@}{\z@}
1984
```

What now follows is the standard class's method for setting up the dimensions. Set `\@tempdimb` to size-dependent initial line length and set `\@tempdima` to the maximum textwidth for the paper width, an inch margin on either side. In the standard classes the initial line length is about 14% greater than `\lxvchars`.

```
\setlength{\@tempdimb}{1.14\lxvchars}
\setlength\@tempdima{\paperwidth}
  \addtolength\@tempdima{-2in}
```

\textwidth Now set the \textwidth depending on the number of columns. In twocolumn mode each column should be no wider than \@tempdimb.

```
\if@twocolumn
  \ifdim\@tempdima>2\@tempdimb\relax
    \setlength\textwidth{2\@tempdimb}
  \else
    \setlength\textwidth{\@tempdima}
  \fi
```

In onecolumn the text should not be wider than the minumum of the paperwidth (less 2in for the margins) and the maximum length of the character line.

```
\else
  \ifdim\@tempdima>\@tempdimb\relax
    \setlength\textwidth{\@tempdimb}
  \else
    \setlength\textwidth{\@tempdima}
  \fi
\fi
```

Adjust the width to be a whole number of points.

```
\@settopoint\textwidth

```

\textheight The \textheight is the height of the text block, excluding headers and footers. This is set according to the \paperheight, to an integral number of lines, and allowing a 1in margin at the top and bottom and a further 1.5in for headers and footers.

```
\setlength\@tempdima{\paperheight}
  \addtolength\@tempdima{-3.5in}
```

Divide this height by the \baselineskip to get the number of lines. Then (re)calculate the \textheight and finally add the \topskip.

```
  \divide\@tempdima\baselineskip
\@tempcnta=\@tempdima
\setlength\textheight{\@tempcnta\baselineskip}
  \addtolength\textheight{\topskip}

```

The margins are calculated.

\oddsidemargin \marginparwidth \evensidemargin The margins depend on the paper size, also for two sided printing the inner margin is made smaller than the outer.

```
\if@twoside
  \setlength\@tempdima       {\paperwidth}
  \addtolength\@tempdima     {-\textwidth}
  \setlength\oddsidemargin   {.4\@tempdima}
```

```
  \addtolength\oddsidemargin {-1in}
  \setlength\marginparwidth  {.6\@tempdima}
  \addtolength\marginparwidth{-\marginparsep}
  \addtolength\marginparwidth{-0.4in}
\else
  \setlength\@tempdima       {\paperwidth}
  \addtolength\@tempdima     {-\textwidth}
  \setlength\oddsidemargin   {.5\@tempdima}
  \addtolength\oddsidemargin {-1in}
  \setlength\marginparwidth  {.5\@tempdima}
  \addtolength\marginparwidth{-\marginparsep}
  \addtolength\marginparwidth{-0.8in} % don't know why this isn't .4
\fi
\ifdim\marginparwidth>2in
  \setlength\marginparwidth{2in}%
\fi
```

Set these values to integer numbers of points, and afterwards calculate the `\evensidemargin`. Jonathon Stickel (`jjstickel@vcn.com`) on 2008/05/30 noted that `\marginparwidth` had to be positive for the initial setting of the sidebar geometry through `\setsidebars`.

```
\@settopoint\oddsidemargin
\@settopoint\marginparwidth
\ifdim\marginparwidth<1pt \setlength\marginparwidth{1pt}\fi

\setlength\evensidemargin  {\paperwidth}
\addtolength\evensidemargin{-2in}
\addtolength\evensidemargin{-\textwidth}
\addtolength\evensidemargin{-\oddsidemargin}
\@settopoint\evensidemargin
```

`\topmargin` The `\topmargin` is the distance below the top of the printable area (1in below the top of the paper) and the top of the box containing the running head.

```
\setlength\topmargin  {\paperheight}
\addtolength\topmargin{-2in}
\addtolength\topmargin{-\headheight}
\addtolength\topmargin{-\headsep}
\addtolength\topmargin{-\textheight}
\addtolength\topmargin{-\footskip}
\addtolength\topmargin{-.5\topmargin}
\@settopoint\topmargin

```

That is the end of the classical algorithm. Now calculate the user-friendly dimensions. The calculations are simpler than in the general case as the `\paperwidth` and `\paperheight` is the same as the `\stockwidth` and `\stockheight`.

We can get the spine and edge margins from the `\oddsidemargin`.

```
\setlength{\spinemargin}{\oddsidemargin}
```

```
\addtolength{\spinemargin}{1in}
\setlrmargins{\spinemargin}{*}{*}
```

Similarly we can get the upper and lower margins from the \topmargin, \headheight and \headskip.

```
\setlength{\uppermargin}{\topmargin}
\addtolength{\uppermargin}{1in}
\addtolength{\uppermargin}{\headheight}
\addtolength{\uppermargin}{\headsep}
\setulmargins{\uppermargin}{*}{*}
```

\@memznegtest DA suggested this in a private email (2003/002/13) to make error checking and reporting a bit more (space) efficient. Use like \@memznegtest{\marginparsep} instead of

```
\ifdim\marginparsep>\z@\else
   \@memerror{\protect\marginparsep\space is zero or negative}{\@ehd}%
\fi
```

If its length variable argument is zero or less it reports an error.

```
\newcommand*{\@memznegtest}[1]{%
  \ifdim#1>\z@\else
 %%%%   \@memerror{\protect#1\space is zero or negative}{\@ehd}%
    \@memwarn{\protect#1\space is zero or negative}%
  \fi}
```

\@memnegtest Reports an error if its length variable argument is negative.

```
\newcommand*{\@memnegtest}[1]{%
  \ifdim#1<\z@
%%%%    \@memerror{\protect#1\space is negative}{\@ehd}%
    \@memwarn{\protect#1\space is negative}%
  \fi}
```

\m@mclassicht The classic adjustment of the \textheight to get an integral number of lines (given an integral number of baselineskips returns a height giving one more line in the block).

```
\newcommand*{\m@mclassicht}{%
  \setlength{\@tempdima}{\textheight}%
  \divide\@tempdima \baselineskip
  \@tempcnta=\@tempdima
  \setlength{\textheight}{\@tempcnta\baselineskip}%
  \addtolength{\textheight}{\topskip}}
```

\m@mlinesht The adjustment of the \textheight to get an integral number of lines (given an inegral number of baselineskips returns a height giving that number of lines).

```
\newcommand*{\m@mlinesht}{%
  \setlength{\@tempdima}{\textheight}%
  \advance\@tempdima -\baselineskip
  \divide\@tempdima \baselineskip
  \@tempcnta=\@tempdima
  \setlength{\textheight}{\@tempcnta\baselineskip}%
  \addtolength{\textheight}{\topskip}}

```

`\m@mnearestht` The adjustment of the `\textheight` to get an integral number of lines with the calculated height being as the closest to the given height. Algorithm supplied by Lars Madsen and Morten Høgholm on 2006/07/27.

```
\newcommand*{\m@mnearestht}{%
  \setlength{\@tempdima}{\textheight}%
  \advance\@tempdima -\topskip
  \advance\@tempdima 0.5\baselineskip
  \divide\@tempdima \baselineskip
  \@tempcnta=\@tempdima
  \setlength{\textheight}{\@tempcnta\baselineskip}%
  \addtolength{\textheight}{\topskip}}

```

`\checkthelayout` `\checkthelayout[`⟨*text*⟩`]` is the user level macro for checking the layout. The ⟨*text*⟩ argument controls which algorithm should be used to calculate the `\textheight`.

```
\newcommand*{\checkthelayout}[1][classic]{%
```

First check the dimensions are not (zero or) negative.

```
  \@memnegtest{\trimedge}
  \@memnegtest{\trimtop}
  \@memznegtest{\stockwidth}
  \@memznegtest{\paperwidth}
  \@memznegtest{\textwidth}
%%  \@memznegtest{\spinemargin}
  \@memnegtest{\spinemargin}
%%  \@memznegtest{\foremargin}
  \@memnegtest{\foremargin}
  \@memznegtest{\marginparsep}
  \@memznegtest{\marginparwidth}
  \@memznegtest{\stockheight}
  \@memznegtest{\paperheight}
  \@memznegtest{\textheight}
%%  \@memznegtest{\uppermargin}
  \@memnegtest{\uppermargin}
%%  \@memznegtest{\lowermargin}
  \@memnegtest{\lowermargin}
%%  \@memznegtest{\headheight}
  \@memnegtest{\headheight}
%%  \@memznegtest{\headsep}
  \@memnegtest{\headsep}
```

```
%%%  \@memznegtest{\footskip}
  \@memnegtest{\footskip}
```

Carry on regardless. We may need to adjust the `\textheight` to get an integral number of lines.

```
  \nametest{#1}{classic}%
  \ifsamename
    \m@mclassicht
  \else
    \nametest{#1}{lines}%
    \ifsamename
      \m@mlinesht
    \else
      \nametest{#1}{nearest}%
      \ifsamename
        \m@mnearestht
      \else
        \nametest{#1}{fixed}
        \ifsamename
        \else%    not classic, lines, nearest, or fixed
          \@memerror{Optional argument is not one of:\MessageBreak
                     classic, fixed, lines, or nearest. \MessageBreak
                     I will assume the default}%
                    {\@ehc}%
        \fi
      \fi
    \fi
  \fi
  \setulmargins{\uppermargin}{*}{*}
```

Check that all the sums add up correctly, or at least to within a small (`\@tempdimb`) error.

```
  \@tempdimb = -1pt
  \@tempdima=\stockwidth
  \advance\@tempdima -\trimedge
  \advance\@tempdima -\paperwidth
  \ifdim\@tempdima<\@tempdimb
    \@tempdima = -\@tempdima
    \@memerror{\protect\paperwidth\space (\the\paperwidth) and/or
                         \protect\trimedge\space (\the\trimedge)
                         are too large for \protect\stockwidth\space (\the\stockwidth)
                         by \the\@tempdima}%
                        {\@ehd}
  \fi
  \@tempdima = \paperwidth
  \advance\@tempdima -\foremargin
  \advance\@tempdima -\textwidth
  \advance\@tempdima -\spinemargin
  \ifdim\@tempdima<\@tempdimb
    \@tempdima = -\@tempdima
```

```
2160    \@memerror{\protect\spinemargin\space (\the\spinemargin) and/or
2161                        \protect\textwidth\space (\the\textwidth) and/or
2162                        \protect\foremargin\space (\the\foremargin)
2163                        are too large for \protect\paperwidth\space (\the\paperwidth)
2164                        by \the\@tempdima}%
2165                       {\@ehd}
2166  \fi
2167  \@tempdima = \stockheight
2168  \advance\@tempdima -\trimtop
2169  \advance\@tempdima -\paperheight
2170  \ifdim\@tempdima<\@tempdimb
2171    \@tempdima = -\@tempdima
2172    \@memerror{\protect\paperheight\space (\the\paperheight) and/or
2173                        \protect\trimtop\space (\the\trimtop)
2174                        are too large for \protect\stockheight\space (\the\stockheight
2175                        by \the\@tempdima}%
2176                       {\@ehd}
2177  \fi
2178  \@tempdima = \paperheight
2179  \advance\@tempdima -\uppermargin
2180  \advance\@tempdima -\textheight
2181  \advance\@tempdima -\lowermargin
2182  \ifdim\@tempdima<\@tempdimb
2183    \@tempdima = -\@tempdima
2184    \@memerror{\protect\uppermargin\space (\the\uppermargin) and/or
2185                        \protect\textheight\space (\the\textheight) and/or
2186                        \protect\lowermargin\space (\the\lowermargin)
2187                        are too large for \protect\paperheight\space (\the\paperheight
2188                        by \the\@tempdima}%
2189                       {\@ehd}
2190  \fi
2191  \@tempdima = \uppermargin
2192  \advance\@tempdima -\headheight
2193  \advance\@tempdima -\headsep
2194  \ifdim\@tempdima<\@tempdimb
2195    \@tempdima = -\@tempdima
2196    \@memerror{\protect\headheight\space (\the\headheight) and/or
2197                        \protect\headsep\space (\the\headsep)
2198                        are too large for \protect\uppermargin\space (\the\uppermargin
2199                        by \the\@tempdima}%
2200                       {\@ehd}
2201  \fi
2202  \@tempdima = \lowermargin
2203  \advance\@tempdima -\footskip
2204  \ifdim\@tempdima<\z@
2205    \@tempdima = -\@tempdima
2206    \@memerror{\protect\footskip\space (\the\footskip)
2207                        is too large for \protect\lowermargin\space (\the\lowermargin)
2208                        by \the\@tempdima}%
2209                       {\@ehd}
```

```
2210  \fi}
2211
```

`\fixthelayout` Calculate the normal LaTeX page layout parameter values. We'll do the heights first as they are independent of the number of columns and the side printing.

```
2212 \newcommand*{\fixthelayout}{%
2213   \topmargin = \trimtop
2214     \advance\topmargin \uppermargin
2215     \advance\topmargin -\headsep
2216     \advance\topmargin -\headheight
2217     \advance\topmargin -1in\relax
```

Now the `\oddsidemargin`.

```
2218   \oddsidemargin = \stockwidth
2219     \advance\oddsidemargin -\trimedge
2220     \advance\oddsidemargin -\paperwidth
2221     \advance\oddsidemargin \spinemargin
2222     \advance\oddsidemargin -1in\relax
```

And the `\evensidemargin`.

```
2223   \evensidemargin = \trimedge
2224     \advance\evensidemargin \foremargin
2225     \advance\evensidemargin -1in\relax
```

Set the values to the nearest whole point.

```
2226   \@settopoint\textwidth
2227   \@settopoint\oddsidemargin
2228   \@settopoint\evensidemargin
```

Fix standard page layouts after possible change of `\textwidth`.

```
2229   \fixheaderwidths}
2230
```

`\typeoutlayout` Why not type out the calculated versions of the designed values?

```
2231 \newcommand*{\typeoutlayout}{%
2232   \typeout{}
2233   \typeout{****************************************************}
2234   \typeout{Stock height and width:
2235                 \the\stockheight\space by \the\stockwidth}
2236   \typeout{Top and edge trims:
2237                 \the\trimtop\space and \the\trimedge}
2238   \typeout{Page height and width:
2239                 \the\paperheight\space by \the\paperwidth}
2240   \typeout{Text height and width:
2241                 \the\textheight\space by \the\textwidth}
2242   \typeout{Spine and edge margins:
2243                 \the\spinemargin\space and \the\foremargin}
2244   \typeout{Upper and lower margins:
2245                 \the\uppermargin\space and \the\lowermargin}
2246   \typeout{Headheight and headsep:
2247                 \the\headheight\space and \the\headsep}
```

```
  \typeout{Footskip:
                \the\footskip}
  \typeout{Columnsep and columnseprule:
                \the\columnsep\space and \the\columnseprule}
  \typeout{Marginparsep and marginparwidth:
                \the\marginparsep\space and \the\marginparwidth}
  \typeout{Sidecapsep and sidecapwidth:
                 \the\sidecapsep\space and \the\sidecapwidth}
  \typeout{Sidebarhsep and sidebarwidth:
                 \the\sidebarhsep\space and \the\sidebarwidth}
  \typeout{Sidebarvsep and sidebartopsep:
                 \the\sidebarvsep\space and \the\sidebartopsep}
  \typeout{Sidebarheight:
                 \the\dimen\sideins}
  \typeout{Sidefoothsep and sidefootwidth:
                 \the\sidefoothsep\space and \the\sidefootwidth}
  \typeout{Sidefootvsep and sidefootheight:
                 \the\sidefootvsep\space and \the\sidefootheight}
  \typeout{**************************************************}
  \typeout{}}
```

`\checkandfixthelayout` This macro checks and fixes the layout, and reports the result. It takes the same optional argument as `\checkthelayout`.

```
% \changes{v1.61803}{2008/01/30}{Changed \cs{checkandfixthelayout}
%          for the extended \cs{checkthelayout} (mempatch v4.5)}
%    \begin{macrocode}
\newcommand*{\checkandfixthelayout}[1][classic]{%
  \checkthelayout[#1]%
  \fixthelayout
  \typeoutlayout}
```

`\fixpdflayout` `\fixdvipslayout` Page layout with pdfLaTeX seems a bit iffy. At the suggestion of Lars Madsen, help with setting viewer (e.g., ghostview) window sizes for dvi/ps. `\fixdvipslayout` does for `dvi` output as `\fixpdflayout` does for `pdf` output.

```
\newcommand*{\fixpdflayout}{%
  \pdfpageheight=\the\stockheight
  \pdfpagewidth=\the\stockwidth
  \ifxetex\else
  \ifdim\pdfvorigin=0pt\pdfvorigin=1in\fi
  \ifdim\pdfhorigin=0pt\pdfhorigin=1in\fi
  \fi}
\newcommand*{\fixdvipslayout}{%
  \AtBeginDvi{\special{papersize=\the\stockwidth,\the\stockheight}}}

\AtBeginDocument{%
  \ifxetex
    \fixpdflayout
```

```
\else
  \ifpdf
    \ifnum\pdfoutput<\@ne
      \fixdvipslayout
    \else
      \fixpdflayout
    \fi
  \else
    \fixdvipslayout
  \fi
\fi}
```

With a landscape document when going `latex -> dvips` the resulting `.ps` file may apear upside down in `ghostview`. If this happens, try putting the following in the document preamble:

```
\addtodef{\fixdvipslayout}{}{%
  \special{!TeXDict begin /landplus90{true}store end }}
```

See `http://www.radicaleye.com.dvips.html` (DVIPS Home Page) for an explanation.
Some other potential specials for PostScript printing may be (at least for an HP 5SiMx LaserJet duplex printer):

```
\special{!TeXDict begin <</Duplex true>>
                        setpagedevice end} % duplex
\special{!TeXDict begin <</Tumble true>>
                        setpagedevice end} % short side binding
```

`\typeoutstandardlayout` Types out the current values of the standard page layout parameters.

```
\newcommand{\typeoutstandardlayout}{%
  \typeout{}
  \typeout{*****************************************************}
  \typeout{Page height and width:
                \the\paperheight\space by \the\paperwidth}
  \typeout{Text height and width:
              \the\textheight\space by \the\textwidth}
  \typeout{Oddside and evenside margins:
                \the\oddsidemargin\space and \the\evensidemargin}
  \typeout{Topmargin and footskip:
                \the\topmargin\space and \the\footskip}
  \typeout{Headheight and headsep:
                \the\headheight\space and \the\headsep}
  \typeout{Columnsep and columnseprule:
                \the\columnsep\space and \the\columnseprule}
  \typeout{Marginparsep and marginparwidth:
                \the\marginparsep\space and \the\marginparwidth}
  \typeout{*****************************************************}
```

```
  \typeout{}
}
```

8.2 Some predefined layouts

A few predefined layouts for the textblock are presented. The `\checkandfixthelayout` macro *must* be called afterwards.

`\medievalpage` This implements Jan Tschichold's reconstruction of the page and textblock layout used by medieval scribes and the early printers []. The spine, top, edge and bottom margins are in the ratios 2:3:4:6.
`\medievalpage[`⟨*num*⟩`]` positions the typeblock on the page with the margins in the above ratios. The spine margin is (page width)/⟨*num*⟩ (default 9). This must be an integer.

```
%%%% s = w/#1, t = 1.5s, e = 2s, f = 3s
\newcommand*{\medievalpage}[1][9]{%
  \spinemargin=\paperwidth
  \divide\spinemargin #1\relax
  \uppermargin = 1.5\spinemargin
  \setlrmarginsandblock{\spinemargin}{*}{2}
  \setulmarginsandblock{\uppermargin}{*}{2}}

```

`\isopage` An implementation of Bringhurst's layout for ISO proportioned pages. It works for any page though. The edge margin is twice the spine, and the bottom margin is twice the top.
`\isopage[`⟨*num*⟩`]` positions and sizes the typeblock on the page according to the above ratios. The spine is (page width)/⟨*num*⟩ and the top margin id (page height)/⟨*num*⟩. ⟨*num*⟩ must be an integer.

```
% s = w/#1, e = 2s, t = h/#1, f = 2h
\newcommand*{\isopage}[1][9]{%
  \spinemargin=\paperwidth
  \divide\spinemargin #1\relax
  \uppermargin=\paperheight
  \divide\uppermargin #1\relax
  \setlrmarginsandblock{\spinemargin}{*}{2}
  \setulmarginsandblock{\uppermargin}{*}{2}}

```

`\semiisopage` An variation on Bringhurst's layout for ISO proportioned pages. It works for any page though. The top margin is the same as the spine, and the edge and bottom margins are twice the spine.
`\semiisopage[`⟨*num*⟩`]` positions and sizes the typeblock on the page according to the above ratios. The spine is (page width)/⟨*num*⟩. ⟨*num*⟩ must be an integer.

```
%%% s = w/#1, t = s, e = 2s, f = e
\newcommand*{\semiisopage}[1][9]{%
  \spinemargin=\paperwidth
```

```
2343   \divide\spinemargin #1\relax
2344   \uppermargin=\spinemargin
2345   \setlrmarginsandblock{\spinemargin}{*}{2}
2346   \setulmarginsandblock{\uppermargin}{*}{2}}
2347
```

\setpagebl
\setpageml
\setpagetl
\setpagetm

`\setpagebl{`⟨*height*⟩`}{`⟨*width*⟩`}{`⟨*ratio*⟩`}` sets a page smaller than the stock size at the bottom left of the stock. The arguments are the the same as for `\settrimmedsize`; ⟨*height*⟩, ⟨*width*⟩ and ⟨*ratio*⟩ of height and width (choose any two) of the desired page size. The trims are adjusted to suit. `\setpagetl` puts the page at the top left, `\setpageml` at the middle left, and `\setpagetm` at the top middle.

```
2348 \newcommand*{\setpagebl}[3]{%
2349   \settrimmedsize{#1}{#2}{#3}%
2350   \trimtop=\stockheight \advance\trimtop -\paperheight
2351   \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2352 \newcommand*{\setpageml}[3]{%
2353   \settrimmedsize{#1}{#2}{#3}
2354   \trimtop=\stockheight \advance\trimtop -\paperheight
2355     \advance\trimtop -0.5\trimtop
2356   \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2357 \newcommand*{\setpagetl}[3]{%
2358   \settrimmedsize{#1}{#2}{#3}
2359   \trimtop=0pt
2360   \trimedge=\stockwidth \advance\trimedge -\paperwidth}
2361 \newcommand*{\setpagetm}[3]{%
2362   \settrimmedsize{#1}{#2}{#3}%
2363   \trimtop=0pt
2364   \trimedge=\stockwidth \advance\trimedge -\paperwidth
2365   \advance\trimedge -0.5\trimedge}
2366
```

\setpagetr
\setpagemr
\setpagebr
\setpagebm
\setpagecc

Similar to those above, these macros set the page on the stock at the top right, middle right, bottom right, bottom middle, and centered.

```
2367 \newcommand*{\setpagetr}[3]{%
2368   \settrimmedsize{#1}{#2}{#3}%
2369   \trimtop=0pt
2370   \trimedge=0pt}
2371 \newcommand*{\setpagemr}[3]{%
2372   \settrimmedsize{#1}{#2}{#3}%
2373   \trimtop=\stockheight \advance\trimtop -\paperheight
2374     \advance\trimtop -0.5\trimtop
2375   \trimedge=0pt}
2376 \newcommand*{\setpagebr}[3]{%
2377   \settrimmedsize{#1}{#2}{#3}%
2378   \trimtop=\stockheight  \advance\trimtop -\paperheight
2379   \trimedge=0pt}
2380 \newcommand*{\setpagebm}[3]{%
2381   \settrimmedsize{#1}{#2}{#3}%
```

```
  \trimtop=\stockheight \advance\trimtop -\paperheight
  \trimedge=\stockwidth \advance\trimedge -\paperwidth
    \advance\trimedge -0.5\trimedge}
\newcommand*{\setpagecc}[3]{%
  \settrimmedsize{#1}{#2}{#3}%
  \trimtop=\stockheight \advance\trimtop -\paperheight
    \advance\trimtop -0.5\trimtop
  \trimedge=\stockwidth \advance\trimedge -\paperwidth
    \advance\trimedge -0.5\trimedge}

```

8.3 Float placement parameters

All float parameters are given default values in the LaTeX kernel. For this reason counters only need to be set with `\setcounter` and other parameters are set using `\renewcommand`.

Limits for the placement of floating objects The settings here make it easier to place floats than with the standard classes.

`\c@topnumber` The *topnumber* counter holds the maximum number of floats that can appear on the top of a text page (classically 2)

```
\setcounter{topnumber}{3}
```

`\topfraction` This indicates the maximum part of a text page that can be occupied by floats at the top (classically 0.7).

```
\renewcommand{\topfraction}{.85}
```

`\c@bottomnumber` The *bottomnumber* counter holds the maximum number of floats that can appear on the bottom of a text page (classically 1).

```
\setcounter{bottomnumber}{2}
```

`\bottomfraction` This indicates the maximum part of a text page that can be occupied by floats at the bottom (classically 0.3).

```
\renewcommand{\bottomfraction}{.5}
```

`\c@totalnumber` This indicates the maximum number of floats that can appear on any text page (classically 3).

```
\setcounter{totalnumber}{4}
```

`\textfraction` This indicates the minimum part of a text page that has to be occupied by text (classically 0.2).

```
\renewcommand{\textfraction}{.1}
```

`\floatpagefraction` This indicates the minimum part of a page that has to be occupied by floating objects before a 'float page' is produced (classically 0.5).

```
\renewcommand{\floatpagefraction}{.7}
```

`\c@dbltopnumber` The *dbltopnumber* counter holds the maximum number of two column floats that can appear on the top of a two column text page (classically 2).

```
2399 \setcounter{dbltopnumber}{3}
```

`\dbltopfraction` This indicates the maximum part of a two column text page that can be occupied by two column floats at the top (classically 0.7).

```
2400 \renewcommand{\dbltopfraction}{.85}
```

`\dblfloatpagefraction` This indicates the minimum part of a page that has to be occupied by two column wide floating objects before a 'float page' is produced (classically 0.5).

```
2401 \renewcommand{\dblfloatpagefraction}{.7}
```

9 Page Styles

The page style *foo* is defined by defining the command `\ps@foo`. This command should make only local definitions. There should be no stray spaces in the definition, since they could lead to mysterious extra spaces in the output.

`\@evenhead` `\@oddhead` `\@evenfoot` `\@oddfoot` The `\ps@...` command defines the macros `\@oddhead`, `\@oddfoot`, `\@evenhead`, and `\@evenfoot` to define the running heads and feet—e.g., `\@oddhead` is the macro to produce the contents of the heading box for odd-numbered pages. It is called inside an `\hbox` of width `\textwidth`.

9.1 Marking conventions

To make headings determined by the sectioning commands, the page style defines the commands `\chaptermark`, `\sectionmark`, ...,
where `\chaptermark{`⟨*TEXT*⟩`}` is called by `\chapter` to set a mark, and so on. The `\...mark` commands and the `\...head` macros are defined with the help of the following macros. (All the `\...mark` commands should be initialized to no-ops.)
LaTeX extends TeX's `\mark` facility by producing two kinds of marks, a 'left' and a 'right' mark, using the following commands:

`\markboth{`⟨*LEFT*⟩`}{`⟨*RIGHT*⟩`}`: Adds both marks.
`\markright{`⟨*RIGHT*⟩`}`: Adds a 'right' mark.
`\leftmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current 'left' mark. `\leftmark` works like TeX's `\botmark` command.
`\rightmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current 'right' mark. `\rightmark` works like TeX's `\firstmark` command.

The marking commands work reasonably well for right marks 'numbered within' left marks–e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if two `\markboth`'s occur on the same page.

Commands like \tableofcontents that should set the marks in some page styles use a \@mkboth command, which is \let by the pagestyle command (\ps@...) to \markboth for setting the heading or to \@gobbletwo to do nothing.

9.2 Defining the page styles

This class provides a set of commands for the user to define new pagestyles. Essentially defining a pagestyle consists of defining the macros \@evenhead, \@oddhead, \@evenfoot, and \@oddfoot. For this class, each header and footer is treated as three parts: a left, center, and right part. In this case, defining a pagestyle consists of specifying these 12 portions of the running headers and footers. The width of the headers/footers may also be specified, rules may be drawn below the headers and/or above the footers, and the complete header and/or footer may be offset with respect to the textblock when the width is not the same as the textwidth.
In the following ⟨*style*⟩ is the name of a pagestyle being defined (e.g., ruled).

\mem@set@ps@xtra@info
\mem@ps@find@real
\mem@ps@safe@change

The class supports an aliasing feature, where a page style name can actually call the code from another. Quite handy. But There might be problems if a use try to modify an alias page style. We care for this by storing some information about each page style, and throwing an error if the user atempts to alter a page style marked as an alias.
\mem@set@ps@xtra@info store two things: the name of the style we are an alias for (if we are not an alias it will be blank) and an indicator whether we are an alias (00 if we are and 01 if we are not).

```
\newcommand\mem@set@ps@xtra@info[3]{%
  \@namedef{ps@#1@aliasfor}{#2}%
  \@namedef{ps@#1@isalias}{#3}}

```

\mem@ps@find@real starts at a given page style, if it is marked as an alias, it will recursively go down the chain of aliases and save the name of the first real page style in \@tempa.

```
\newcommand\mem@ps@find@real[1]{%
  \if\@nameuse{ps@#1@isalias}\relax
  \mem@ps@find@real{\@nameuse{ps@#1@aliasfor}}
  \else\def\@tempa{#1}\fi}

```

\mem@ps@safe@change takes a page style name, checks to see if it is defined, if it is, it checks to see if it is safe to change it.

```
\newcommand\mem@ps@safe@change[1]{%
 \@ifundefined{ps@#1}{%
    \@mememrror{Undefined pagestyle '#1', so I cannot change it}{}}{}
 \if\@nameuse{ps@#1@isalias}\relax \mem@ps@find@real{#1}
 \@mememrror{The pagestyle '#1' is marked as an alias page style.^^J
   Modifying an alias page style may give unexpected results.^^J The
   alias chain resolves to the real page style '\@tempa', so try
```

```
2418     issuing^^J \string\copypagestyle{#1}{\@tempa}^^J before before
2419     modifying '#1'}{} \fi }
2420
```

\makeevenhead The command \makeevenhead{⟨*style*⟩}{⟨*left*⟩}{⟨*center*⟩}{⟨*right*⟩} specifies that the left, center and right portions of the even header for pagestyle ⟨*style*⟩ are defined as the other three arguments, respectiveley. Internally it defines the commands \styleeheadl, \styleeheadc and \styleeheadr to be ⟨*left*⟩, ⟨*center*⟩ and ⟨*right*⟩ respectively.

```
2421 \newcommand{\makeevenhead}[4]{%
2422   \mem@ps@safe@change{#1}
2423   \@namedef{#1eheadl}{#2}
2424   \@namedef{#1eheadc}{#3}
2425   \@namedef{#1eheadr}{#4}
2426 }
```

\makeoddhead \makeevenfoot \makeoddfoot These three macros are similar to \makeevenhead except that they are for the oddhead, evenfoot and oddfoot.

```
2427 \newcommand{\makeoddhead}[4]{%
2428   \mem@ps@safe@change{#1}
2429   \@namedef{#1oheadl}{#2}
2430   \@namedef{#1oheadc}{#3}
2431   \@namedef{#1oheadr}{#4}
2432 }
2433 \newcommand{\makeevenfoot}[4]{%
2434   \mem@ps@safe@change{#1}
2435   \@namedef{#1efootl}{#2}
2436   \@namedef{#1efootc}{#3}
2437   \@namedef{#1efootr}{#4}
2438 }
2439 \newcommand{\makeoddfoot}[4]{%
2440   \mem@ps@safe@change{#1}
2441   \@namedef{#1ofootl}{#2}
2442   \@namedef{#1ofootc}{#3}
2443   \@namedef{#1ofootr}{#4}
2444 }
2445
```

\makerunningwidth \makerunningheadwidth \makerunningfootwidth \m@mhfstyle \m@mopthfwidth The macro \makerunningwidth{⟨*style*⟩}[⟨*ftlength*⟩]{⟨*length*⟩} sets the width of the headers and footers of pagestyle ⟨*style*⟩ to be ⟨*length*⟩, but if ⟨*ftlength*⟩ is present the footer width is set to ⟨*ftlength*⟩. The lengths are stored as the macros \styleheadrunwidth and \stylefootrunwidth. The two widths can be set individually using \makerunningheadwidth{⟨*style*⟩}{⟨*length*⟩} and \makerunningfootwidth{⟨*style*⟩}{⟨*length*⟩}. .

```
2446 \newcommand*{\makerunningwidth}[1]{%
2447   \mem@ps@safe@change{#1}%
2448   \def\m@mhfstyle{#1}%
2449   \m@mopthfwidth}
2450 \newcommand*{\m@mopthfwidth}[2][\@mpty]{%
```

```
% \setlength\@tempdima{#2}
  \@namedef{\m@mhfstyle headrunwidth}{#2}%
% \expandafter\edef\csname \m@mhfstyle headrunwidth\endcsname{\the\@tempdima}
  \ifx\@mpty #1
    \@namedef{\m@mhfstyle footrunwidth}{#2}%
%   \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}
  \else
    \@namedef{\m@mhfstyle footrunwidth}{#1}%
%   \setlength\@tempdima{#1}
%   \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}
  \fi}
\newcommand*{\makerunningheadwidth}[2]{%
% \setlength\@tempdima{#2}%
% \expandafter\edef\csname \m@mhfstyle headrunwidth\endcsname{\the\@tempdima}%
  \mem@ps@safe@change{#1}%
  \@namedef{#1headrunwidth}{#2}%
}
\newcommand*{\makerunningfootwidth}[2]{%
% \setlength\@tempdima{#2}%
  \mem@ps@safe@change{#1}%
  \@namedef{#1footrunwidth}{#2}%
% \expandafter\edef\csname \m@mhfstyle footrunwidth\endcsname{\the\@tempdima}%
}
```

\normalrulethickness
\footruleheight
\footruleskip
\makeheadrule
\makefootrule

`\normalrulethickness` is the thickness of a normal horizontal or vertical rule.
`\footruleheight` is the height of a normal rule above a footer (actually zero).
`\footruleskip` is a distance sufficient to ensure that a foot rule will appear between the bottom of the textblock and above any actual footer.
(There was a CTT thread *ngerman, fancyhdr and* `\footrulewidth` *— bug?* in December 2002 that bears on the definitions below).

```
\newlength{\normalrulethickness}
  \setlength{\normalrulethickness}{0.4pt}
\newcommand{\footruleheight}{0pt}
\newcommand{\footruleskip}{0.3\normalbaselineskip}
```

The macro `\makeheadrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*height*⟩`}` specifies the width and height of the header rule for ⟨*style*⟩.
Similarly `\makefootrule{`⟨*style*⟩`}{`⟨*width*⟩`}{`⟨*height*⟩`}{`⟨*skip*⟩`}` specifies the width, height and skip for the footrule.

```
\newcommand{\makeheadrule}[3]{%
  \mem@ps@safe@change{#1}%
  \@namedef{#1headrule}{%
    \hrule\@width #2\@height #3 \vskip-#3}}
\newcommand{\makefootrule}[4]{%
  \mem@ps@safe@change{#1}%
  \@namedef{#1footrule}{%
    \vskip-#4\vskip-#3
    \hrule\@width #2\@height #3 \vskip #4}}
```

\makeheadposition `\makeheadposition{⟨style⟩}{⟨eheadpos⟩}{⟨oheadpos⟩}{⟨efootpos⟩}{⟨ofootpos⟩}` specifies the horizontal positioning of the even and odd headers and footers, respectively, for the pagestyle ⟨*style*⟩. Each of the ⟨*...pos*⟩ arguments may be either `flushleft`, `center` or `flushright`, with the obvious meanings. An empty argument (or an unrecognised one) is equivalent to `center`.

```
\newcommand{\makeheadposition}[5]{%
  \mem@ps@safe@change{#1}%
```

Do the even head position first.

```
  \nametest{flushleft}{#2}
  \ifsamename
    \@namedef{#1evenhpl}{\relax} \@namedef{#1evenhpr}{\hss}
  \else
    \nametest{flushright}{#2}
    \ifsamename
      \@namedef{#1evenhpl}{\hss} \@namedef{#1evenhpr}{\relax}
    \else
      \@namedef{#1evenhpl}{\hss} \@namedef{#1evenhpr}{\hss}
    \fi
  \fi
```

And similarly for the odd head and even & odd footers.

```
  \nametest{flushleft}{#3}
  \ifsamename
    \@namedef{#1oddhpl}{\relax} \@namedef{#1oddhpr}{\hss}
  \else
    \nametest{flushright}{#3}
    \ifsamename
      \@namedef{#1oddhpl}{\hss} \@namedef{#1oddhpr}{\relax}
    \else
      \@namedef{#1oddhpl}{\hss} \@namedef{#1oddhpr}{\hss}
    \fi
  \fi
  \nametest{flushleft}{#4}
  \ifsamename
    \@namedef{#1evenfpl}{\relax} \@namedef{#1evenfpr}{\hss}
  \else
    \nametest{flushright}{#4}
    \ifsamename
      \@namedef{#1evenfpl}{\hss} \@namedef{#1evenfpr}{\relax}
    \else
      \@namedef{#1evenfpl}{\hss} \@namedef{#1evenfpr}{\hss}
    \fi
  \fi
  \nametest{flushleft}{#5}
  \ifsamename
    \@namedef{#1oddfpl}{\relax} \@namedef{#1oddfpr}{\hss}
  \else
    \nametest{flushright}{#5}
    \ifsamename
```

```
2530      \@namedef{#1oddfpl}{\hss} \@namedef{#1oddfpr}{\relax}
2531    \else
2532      \@namedef{#1oddfpl}{\hss} \@namedef{#1oddfpr}{\hss}
2533    \fi
2534  \fi}
2535
```

`\makepsmarks` `\makepshook` The macro `\makepsmarks{`⟨*style*⟩`}{`⟨*code*⟩`}` specifies that ⟨*code*⟩ is the definition of the hook for pagestyle ⟨*style*⟩. `\makepshook` was a deprecated version of `\makepsmarks` and is now deleted.

```
2536 \newcommand{\makepsmarks}[2]{\mem@ps@safe@change{#1}\@namedef{#1pshook}{#2}}
2537
```

`\m@mhe@dreset` To cater for potential baselineskip changes, such as `\linespread{2}`, we have to ensure they don't percolate into the header/footer. (See CTT thread *memoir + linespread*, 2004/02/11)

```
2538 \newcommand*{\m@mhe@dreset}{\def\baselinestretch{1}\normalsize}
2539
```

`\makeheadfootvposition` The headers and footers are typeset inside `\parbox`'es, using `\makeheadfootvposition{`⟨*style*⟩`}{`⟨*headvpos*⟩`}{`⟨*footvpost*⟩`}` users can specify whether the alignment should be b,t or c.The default is b.

```
2540 \newcommand*\makeheadfootvposition[3]{%
2541   \mem@ps@safe@change{#1}%
2542   \@namedef{#1headvplacement}{#2}\@namedef{#1footvplacement}{#3}}
2543
```

`\makepagestyle` At last we can put everything together for defining a new pagestyle, via the macro `\makepagestyle{`⟨*style*⟩`}` which initially sets up a new pagestyle ⟨*style*⟩ corresponding the the LaTeX `empty` pagestyle. The prior macros can then be used to make modifications to the style. We also make sure to specify that this is a 'real' page style, and thus sets the empty alias name and the alias test to 01.

```
2544 \newcommand{\makepagestyle}[1]{%
2545   \mem@set@ps@xtra@info{#1}{}{01}%
```

First define the macro `\ps@style`, which in turn defines the macros `\@evenhead`, `\@oddhead`, `\@evenfoot` and `\@oddfoot`.

```
2546   \@namedef{ps@#1}{%
2547     \@namedef{#1@evenhead}{%
```

The code for the definition of `\@evenhead` and friends is based on code from Piet van Oostrum's fancyhdr package [Oos96]. The three parts of the header are put into parboxes, with fills between them, and the whole lot is put into a box the width of the header. Fillers are put before and after the main box which control the header position.
First the left filler which is either `\relax` or `\hss`, and then start the main box.

```
2548       \@nameuse{#1evenhpl}\hb@xt@\@nameuse{#1headrunwidth}{\m@mhe@dreset%
2549         \vbox{\hbox{%
```

The left part of the header.

```
2550        \rlap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2551          \raggedright\@nameuse{#1eheadl}\strut}}\hfill
```

The center part of the header.

```
2552              \parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2553          \centering\@nameuse{#1eheadc}\strut}\hfill
```

The right part of the header.

```
2554        \llap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2555          \raggedleft\@nameuse{#1eheadr}\strut}}}%
```

Finally, the header rule and finish with the right filler, which is either `\relax` or `\hss`.

```
2556        \@nameuse{#1headrule}}}\@nameuse{#1evenhpr}}%
```

The code for the `\@oddhead`, `\@evenfoot` and `\@oddfoot` follows a similar pattern. Here is `\@oddhead`.

```
2557    \@namedef{#1@oddhead}{%
2558      \@nameuse{#1oddhpl}\hb@xt@\@nameuse{#1headrunwidth}{\m@mhe@dreset%
2559        \vbox{\hbox{%
2560        \rlap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2561          \raggedright\@nameuse{#1oheadl}\strut}}\hfill
2562              \parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2563          \centering\@nameuse{#1oheadc}\strut}\hfill
2564        \llap{\parbox[\@nameuse{#1headvplacement}]{\@nameuse{#1headrunwidth}}{%
2565          \raggedleft\@nameuse{#1oheadr}\strut}}}%
2566        \@nameuse{#1headrule}}}\@nameuse{#1oddhpr}}%
```

And `\@evenfoot`. For the footers the rules come *before* any foot entries.

```
2567    \@namedef{#1@evenfoot}{%
2568      \@nameuse{#1evenfpl}\hb@xt@\@nameuse{#1footrunwidth}{\m@mhe@dreset%
2569        \vbox{\@nameuse{#1footrule}\hbox{%
2570        \rlap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2571          \raggedright\@nameuse{#1efootl}\strut}}\hfill
2572              \parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2573          \centering\@nameuse{#1efootc}\strut}\hfill
2574        \llap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2575          \raggedleft\@nameuse{#1efootr}\strut}}}%
2576        }}\@nameuse{#1evenfpr}}%
```

Lastly the `\@oddfoot`.

```
2577    \@namedef{#1@oddfoot}{%
2578      \@nameuse{#1oddfpl}\hb@xt@\@nameuse{#1footrunwidth}{\m@mhe@dreset%
2579        \vbox{\@nameuse{#1footrule}\hbox{%
2580        \rlap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2581          \raggedright\@nameuse{#1ofootl}\strut}}\hfill
2582              \parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2583          \centering\@nameuse{#1ofootc}\strut}\hfill
2584        \llap{\parbox[\@nameuse{#1footvplacement}]{\@nameuse{#1footrunwidth}}{%
2585          \raggedleft\@nameuse{#1ofootr}\strut}}}%
2586        }}\@nameuse{#1oddfpr}}%
```

Now we define \@evenhead etc., in terms of #1@evenhead.

```
\def\@evenhead{\@nameuse{#1@evenhead}}%
\def\@oddhead{\@nameuse{#1@oddhead}}%
\def\@evenfoot{\@nameuse{#1@evenfoot}}%
\def\@oddfoot{\@nameuse{#1@oddfoot}}%
```

To finish off the definition of \ps@style, add in a hook which can be defined so that it adds additional code, if required.

```
\@nameuse{#1pshook}}%
```

The final part of setting up the new pagestyle is defining all the macros called by \ps@style, and giving them default values. Make the headers and footers empty.

```
\makeevenhead{#1}{}{}{}%
\makeoddhead{#1}{}{}{}%
\makeevenfoot{#1}{}{}{}%
\makeoddfoot{#1}{}{}{}%
```

Make the headers/footers the same width as the \textwidth, center them, and ensure that the rules have zero height so that they will be invisible.

```
\makerunningwidth{#1}{\textwidth}%
\makeheadposition{#1}{}{}{}{}%
\makeheadrule{#1}{\textwidth}{0pt}%
\makefootrule{#1}{\textwidth}{\footruleheight}{\footruleskip}%
```

Set the initial vertical header and footer positions.

```
\makeheadfootvposition{#1}{b}{b}%
```

Finally, there is no addtional code needed, so make the hook empty, and we are done.

```
\makepsmarks{#1}{}}

```

\aliaspagestyle The command \aliaspagestyle{⟨*alias*⟩}{⟨*original*⟩} defines the ⟨*alias*⟩ pagestyle to be an alias for the ⟨*original*⟩ pagestyle. We remember to set the stored alias name, and sets the alias test to true (00).

```
\newcommand{\aliaspagestyle}[2]{%
  \mem@set@ps@xtra@info{#1}{#2}{00}%
  \@namedef{ps@#1}{\@nameuse{ps@#2}}}

```

\copypagestyle The command \copypagestyle{⟨*new*⟩}{⟨*original*⟩} defines the ⟨*new*⟩ pagestyle to be a copy of the ⟨*original*⟩ pagestyle.
It first makes the ⟨*new*⟩ (empty) pagestyle then defines the internals in terms of the ⟨*original*⟩ internals.

```
\newcommand{\copypagestyle}[2]{%
  \makepagestyle{#1}%
```

Do the headers and footers.

```
  \makeevenhead{#1}{\@nameuse{#2eheadl}}%
                   {\@nameuse{#2eheadc}}{\@nameuse{#2eheadr}}%
  \makeoddhead{#1}{\@nameuse{#2oheadl}}%
```

```
                {\@nameuse{#2oheadc}}{\@nameuse{#2oheadr}}%
  \makeevenfoot{#1}{\@nameuse{#2efootl}}%
                {\@nameuse{#2efootc}}{\@nameuse{#2efootr}}%
  \makeoddfoot{#1}{\@nameuse{#2ofootl}}%
               {\@nameuse{#2ofootc}}{\@nameuse{#2ofootr}}%
```

Set the width.

```
  \makerunningwidth{#1}[\@nameuse{#2footrunwidth}]{\@nameuse{#2headrunwidth}}%
```

Specify the `\headposition`.

```
  \@namedef{#1evenhpl}{\@nameuse{#2evenhpl}}%
  \@namedef{#1oddhpl}{\@nameuse{#2oddhpl}}%
  \@namedef{#1evenhpr}{\@nameuse{#2evenhpr}}%
  \@namedef{#1oddhpr}{\@nameuse{#2oddhpr}}%
```

Also vertically

```
  \makeheadfootvposition{#1}{\@nameuse{#2headvplacement}}{\@nameuse{#2footvplacement}}%
```

Specify the feet

```
  \@namedef{#1evenfpl}{\@nameuse{#2evenfpl}}%
  \@namedef{#1oddfpl}{\@nameuse{#2oddfpl}}%
  \@namedef{#1evenfpr}{\@nameuse{#2evenfpr}}%
  \@namedef{#1oddfpr}{\@nameuse{#2oddfpr}}%
```

Specify the head and foot rules.

```
  \@namedef{#1headrule}{\@nameuse{#2headrule}}%
  \@namedef{#1footrule}{\@nameuse{#2footrule}}%
```

And pick up the hook.

```
  \makepsmarks{#1}{\@nameuse{#2pshook}}}

```

`\ifonlyfloats` `\ifonlyfloats{`⟨*yes*⟩`}{`⟨*no*⟩`}` processes ⟨*yes*⟩ if the macro is called on a page consisting only of floats, otherwise ⟨*no*⟩ is processed. `\if@fcolmade` is specified in `ltoutput.dtx`.

```
\newcommand{\ifonlyfloats}[2]{\if@fcolmade #1\else #2\fi}

```

`\mergepagefloatstyle` `\mergepagefloatstyle{`⟨*style*⟩`}{`⟨*textstyle*⟩`}{`⟨*floatstyle*⟩`}` makes a new page style ⟨*style*⟩ that is ⟨*textstyle*⟩ on normal pages but uses ⟨*floatstyle*⟩ on float-only pages. Both ⟨*textstyle*⟩ and ⟨*floatstyle*⟩ must have been previously defined.

```
\newcommand{\mergepagefloatstyle}[3]{%
```

Make sure that the two styles have been called, otherwise we get some undefined errors.

```
  \@nameuse{ps@#3}\@nameuse{ps@#2}%
```

Specify the pagestyle's headers and footers.

```
  \@namedef{ps@#1}{%
  \def\@evenhead{\ifonlyfloats{\@nameuse{#3@evenhead}}%
                  {\@nameuse{#2@evenhead}}}%
  \def\@oddhead{\ifonlyfloats{\@nameuse{#3@oddhead}}%
```

```
                {\@nameuse{#2@oddhead}}%
  \def\@evenfoot{\ifonlyfloats{\@nameuse{#3@evenfoot}}%
                 {\@nameuse{#2@evenfoot}}}%
  \def\@oddfoot{\ifonlyfloats{\@nameuse{#3@oddfoot}}%
                {\@nameuse{#2@oddfoot}}}%
```

Set the hook to the ⟨*textstyle*⟩ on the assumption that that is more complex that required for a float page.

```
  \@namedef{#1pshook}{\@nameuse{#2pshook}}%
```

That's it.

```
}}
```

The pagestyles *empty* and *plain* are defined in `latex.dtx`. However, I will redefine them here, just in case someone takes a fancy to modifying them.

`\ps@empty` The *empty* pagestyle is simple, it's just what we get when we call `\makepagestyle`.

```
\makepagestyle{empty}
```

`\ps@plain` The *plain* pagestyle is also simple, it just puts the page number at the bottom middle of the page. We call `\makepagestyle{plain}` and then adjust as required.

```
\makepagestyle{plain}
  \makeevenfoot{plain}{}{\thepage}{}
  \makeoddfoot{plain}{}{\thepage}{}
```

`\ps@simple` The *simple* page style simply puts the page number in the header at the outer margin.

```
\makepagestyle{simple}
  \makeevenhead{simple}{\thepage}{}{}
  \makeoddhead{simple}{}{}{\thepage}
```

`\nouppercaseheads` `\uppercaseheads` `\memUChead` Spurred by Lars Madsen's memexsupp v0.5 package here is a way of switching uppercasing in the headings pagestyle.

```
\newcommand*{\nouppercaseheads}{\let\memUChead\relax}
\newcommand*{\uppercaseheads}{\let\memUChead\MakeUppercase}
\uppercaseheads
```

The macros `\createplainmark`, `\createmark` and `\addtopsmarks` are modified versions of code supplied by Lars Madsen.

`\createplainmark` `\createplainmark{`⟨*type*⟩`}{`⟨*marks*⟩`}{`⟨*text*⟩`}` creates a `\typemark` (⟨*type*⟩ is `toc`, `bib`, etc) with ⟨*marks*⟩ (left, both, right) whose contents are ⟨*text*⟩. For example `\createplainmark{toc}{both}{\contentsname}`. The difference

between plain and regular marks, is that plain marks use a fixed text, whereas regular marks recieve dynamic text and thus is given an argument.

```
\newcommand*{\createplainmark}[3]{%
  \nametest{#2}{left}%
  \ifsamename
    \@namedef{#1mark}{\markboth{\memUChead{#3}}{}}%
  \else
    \nametest{#2}{right}%
    \ifsamename
      \@namedef{#1mark}{\markright{\memUChead{#3}}}%
    \else
      \nametest{#2}{both}%
      \ifsamename\else
        \@memerror{%
          Unknown mark setting type `#2' for #1mark}{%
          I expected `left', `both' or `right'. \MessageBreak
          I will assume you meant `both'}%
      \fi
      \@namedef{#1mark}{\markboth{\memUChead{#3}}{\memUChead{#3}}}%
    \fi
  \fi}

```

`\createmark` `\createmark{`⟨*division*⟩`}{`⟨*marks*⟩`}{`⟨*show*⟩`}{`⟨*prefix*⟩`}{`⟨*postfix*⟩`}` creates a `\divisionmark` with ⟨*marks*⟩ (= `left`, `both` or `right`) controlling which marks are set, ⟨*show*⟩ (= `shownumber` or `nonumber`) controls the display of the divison number in `\mainmatter`, ⟨*prefix*⟩ is text before the number and ⟨*postfix*⟩ is text after the number.

```
\newcommand\createmark[5]{%
```

Handle `(show/no)number`, fails to `shownumber`.

```
  \def\@tempa{00}
  \nametest{#3}{nonumber}%
  \ifsamename
    \def\@tempa{01}%
  \else
    \nametest{#3}{shownumber}
    \ifsamename\else
      \@memerror{Unknown numbering value `#3' for #1mark}%
      {I expected `shownumber' or `nonumber'.\MessageBreak
       I will assume you meant `shownumber'}%
    \fi
  \fi
  \expandafter\if\@tempa%          compares the two \@tempa digits
    \@namedef{#1marksn}##1{##1}%
  \else
    \@namedef{#1marksn}{\@gobble}%
  \fi
```

The three ⟨*marks*⟩ cases, `left`, `both` and `right`; fails to `both`.

```
\nametest{#2}{left}%
\ifsamename
  \@namedef{#1mark}##1{%
    \@setclcnt{#1}{@memmarkcntra}%
    \advance\c@@memmarkcntra\m@ne
    \markboth{%
      \memUChead{%
        \ifnum \c@secnumdepth > \c@@memmarkcntra
          \if@mainmatter
            \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
          \fi
        \fi
        ##1}}{}}%
\else
  \nametest{#2}{right}
  \ifsamename
    \@namedef{#1mark}##1{%
      \@setclcnt{#1}{@memmarkcntra}
      \advance\c@@memmarkcntra\m@ne
      \markright{%
        \memUChead{%
          \ifnum \c@secnumdepth > \c@@memmarkcntra
            \if@mainmatter%
              \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
            \fi%
          \fi%
          ##1}}}%
  \else
    \nametest{#2}{both}%
    \ifsamename\else
      \@memerror{%
      Unknown mark setting type `#2' for #1mark}{%
      I expected `left', `both' or `right'. \MessageBreak
      I will assume you meant `both'}%
    \fi
  \@namedef{#1mark}##1{%
    \@setclcnt{#1}{@memmarkcntra}
    \advance\c@@memmarkcntra\m@ne
    \markboth{%
      \memUChead{%
        \ifnum \c@secnumdepth > \c@@memmarkcntra
          \if@mainmatter
            \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
          \fi
        \fi
        ##1}}{%
      \memUChead{%
        \ifnum \c@secnumdepth > \c@@memmarkcntra
          \if@mainmatter
            \@nameuse{#1marksn}{#4\@nameuse{the#1}#5}%
```

```
2749            \fi
2750          \fi
2751          ##1}}}%
2752      \fi
2753  \fi}
2754
```

\addtopsmarks `\addtopsmarks{⟨pagestyle⟩}{⟨prepend⟩}{⟨append⟩}` inserts ⟨*prepend*⟩ and ⟨*append*⟩ at the start and end of the current definition of `\pagestylepshook`.

```
2755 \newcommand\addtopsmarks[3]{%
2756   \mem@ps@safe@change{#1}%
2757   \expandafter\addtodef\expandafter{\csname #1pshook\endcsname}{#2}{#3}}
```

\clearplainmark \clearmark For some page styles it is handy to be able to make sure that no section or otherwise add data to the `\left-` and `\rightmark`. The macros `\clearplainmark` and `\clearmark` will take one argument specifying which mark to clear, and then redefine this mark to do nothing or to gobble its given argument.

```
2758 \newcommand\clearplainmark[1]{%
2759   \@namedef{#1mark}{}}
2760 \newcommand\clearmark[1]{%
2761   \@namedef{#1mark}{\@gobble}}
```

\ps@headings The *headings* pagestyle behaves differently for twosided and onesided printing. This is a rewrite of the standard style.

```
2762 \if@twoside
```

The footer is empty and the header contains the page number and one of the marks.

```
2763   \makepagestyle{headings}
2764     \makepsmarks{headings}{%
2765       \def\chaptermark##1{%
2766         \markboth{\memUChead{%
2767           \ifnum \c@secnumdepth >\m@ne
2768             \if@mainmatter
2769               \@chapapp\ \thechapter. \ %
2770             \fi
2771           \fi
2772           ##1}}{}}%
2773       \def\tocmark{\markboth{\memUChead{\contentsname}}{\memUChead{\contentsname}}}%
2774       \def\lofmark{\markboth{\memUChead{\listfigurename}}{\memUChead{\listfigurename}}}%
2775       \def\lotmark{\markboth{\memUChead{\listtablename}}{\memUChead{\listtablename}}}%
2776       \def\bibmark{\markboth{\memUChead{\bibname}}{\memUChead{\bibname}}}%
2777       \def\indexmark{\markboth{\memUChead{\indexname}}{\memUChead{\indexname}}}%
2778       \def\sectionmark##1{%
2779         \markright{\memUChead{%
2780           \ifnum \c@secnumdepth > \z@
2781             \thesection. \ %
2782           \fi
2783           ##1}}}}
```

And here's a version of the above \makepsmarks using \createmark and \createplainmark.

```
    \makepsmarks{headings}{%
      \createmark{chapter}{left}{shownumber}{\@chapapp\ }{. \ }
      \createmark{section}{right}{shownumber}{}{. \ }
      \createplainmark{toc}{both}{\contentsname}
      \createplainmark{lof}{both}{\listfigurename}
      \createplainmark{lot}{both}{\listtablename}
      \createplainmark{bib}{both}{\bibname}
      \createplainmark{index}{both}{\indexname}
      \createplainmark{glossary}{both}{\glossaryname}
    }
    \makeevenhead{headings}{\thepage}{}{\slshape\leftmark}
    \makeoddhead{headings}{\slshape\rightmark}{}{\thepage}
\else
```

For one sided printing even and odd pages are treated the same, so no need to bother with the evenhead, and just the \rightmark is used.

```
  \makepagestyle{headings}
    \makepsmarks{headings}{%
      \def\chaptermark##1{%
        \markright{\memUChead{%
          \ifnum \c@secnumdepth >\m@ne
            \if@mainmatter
              \@chapapp\ \thechapter. \ %
            \fi
          \fi
          ##1}}}%
      \def\tocmark{\markright{\memUChead{\contentsname}}}%
      \def\lofmark{\markright{\memUChead{\listfigurename}}}%
      \def\lotmark{\markright{\memUChead{\listtablename}}}%
      \def\bibmark{\markright{\memUChead{\bibname}}}%
      \def\indexmark{\markright{\memUChead{\indexname}}}}
```

And here's a version of the above \makepsmarks using \createmark and \createplainmark.

```
    \makepsmarks{headings}{%
      \createmark{chapter}{right}{shownumber}{\@chapapp\ }{. \ }
      \createplainmark{toc}{right}{\contentsname}
      \createplainmark{lof}{right}{\listfigurename}
      \createplainmark{lot}{right}{\listtablename}
      \createplainmark{bib}{right}{\bibname}
      \createplainmark{index}{right}{\indexname}
      \createplainmark{glossary}{right}{\glossaryname}
    }
    \makeoddhead{headings}{\slshape\rightmark}{}{\thepage}
\fi

```

\ps@myheadings The *myheadings* page style is simpler because the user has to specify the

contents using \markboth and \markright commands. This is the definition used in the standard classes.

```
}
 \newcommand{\ps@myheadings}{%
   \let\@oddfoot\@empty\let\@evenfoot\@empty
   \def\@evenhead{\thepage\hfil{\slshape\leftmark}}%
   \def\@oddhead{{\slshape\rightmark}\hfil\thepage}%
   \let\@mkboth\@gobbletwo
   \let\chaptermark\@gobble
   \let\sectionmark\@gobble
 }
```

Translating that into our terms we get:

```
\makepagestyle{myheadings}
  \makepsmarks{myheadings}{%
    \let\chaptermark\@gobble
    \let\sectionmark\@gobble
    \def\tocmark{}%
    \def\lofmark{}%
    \def\lotmark{}%
    \def\bibmark{}%
    \def\indexmark{}%
    \def\glossarymark{}}
  \makeevenhead{myheadings}{\thepage}{}{\slshape\leftmark}
  \makeoddhead{myheadings}{\slshape\rightmark}{}{\thepage}

```

(Kai von Fintel (fintel@mit.edu) emailed me on 2003/02/24 saying that the original \let\tocmark\@gobble in the myheadings pagestyle did nasty things to the ToC, etc. Now using \def\tocmark{}.)

\ps@chapter
\ps@part
\ps@cleared

The standard classes use the *plain* pagestyle for the first page of a chapter. This class uses the *chapter* instead, which is aliased to *plain*. Similarly for parts. Further, \cleardoublepage uses whatever pagestyle is in effect for the empty verso page. I find that this looks odd if the header contains a chapter name. This class uses the *cleared* pagestyle in this case. I have aliased this to *empty*; the *plain* would be another reasonable choice.

```
\aliaspagestyle{chapter}{plain}
\aliaspagestyle{part}{plain}
\aliaspagestyle{cleared}{empty}

```

\cleardoublepage

A slight adjustment to the kernel definition to set a pagestyle.

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
  \hbox{}\thispagestyle{cleared}%
  \newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}

```

`\ps@ruled` Partly to show how it is done, the class provides a *ruled* pagestyle. In this style the headers and footers are the same width as the textblock, there is a rule under the header, page numbers are set in the footers at the outside of the page. Even page headers have the chapter number and title at the left, and odd page headers have the section title at the right.
Start by making the (empty) *ruled* pagestyle.

```
\makepagestyle{ruled}
```

There is no need to change the default width (which is the `\textwidth`), nor the default positions (centered), nor to make the footrule visible. We do, though, have to put the page numbers into the footers.

```
\makeevenfoot{ruled}{\thepage}{}{}
\makeoddfoot{ruled}{}{}{\thepage}
```

Make the header rule visible and equal to the `\textwidth`.

```
\makeheadrule{ruled}{\textwidth}{\normalrulethickness}
```

`\@ruledmarks` We have to make sure that the `\chapter` and `\section` commands make the appropriate marks for use in the headers. We use the hook for this. Note that contrary to normal LaTeX practice, the titles are not automatically upper-cased. The marks for the tocbibinds also need adjusting.

```
\newcommand{\@ruledmarks}{%
  \def\chaptermark##1{%
    \markboth{%
      \ifnum \c@secnumdepth >\m@ne
        \if@mainmatter
          \thechapter. \ %
        \fi
      \fi
      ##1}{}}
  \def\sectionmark##1{\markright{##1}}
  \def\tocmark{\markboth{\contentsname}{}}
  \def\lofmark{\markboth{\listfigurename}{}}
  \def\lotmark{\markboth{\listtablename}{}}
  \def\bibmark{\markboth{\bibname}{}}
  \def\indexmark{\markboth{\indexname}{}}
  \def\glossarymark{\markboth{\glossaryname}{}}
}
```

And here's a version using `\createmark` and friends.

```
\renewcommand*{\@ruledmarks}{%
  \nouppercaseheads
  \createmark{chapter}{left}{shownumber}{}{. \space}
  \createmark{section}{right}{shownumber}{}{. \space}
  \createplainmark{toc}{both}{\contentsname}
  \createplainmark{lof}{both}{\listfigurename}
  \createplainmark{lot}{both}{\listtablename}
  \createplainmark{bib}{both}{\bibname}
  \createplainmark{index}{both}{\indexname}
  \createplainmark{glossary}{both}{\glossaryname}}
```

```
\makepsmarks{ruled}{\@ruledmarks}
```

We can now define the even page header which is to have the chapter title at the left. As the chapter mark did no upper-casing we will print it using small caps, but just use the normal font for section title on the odd page header.

```
\makeevenhead{ruled}{\scshape\leftmark}{}{}
\makeoddhead{ruled}{}{}{\rightmark}

```

This is all that we need to do for the *ruled* pagestyle.

`\ps@Ruled` Also define a *Ruled* pagestyle similar to *ruled* except that the headers and footers are 10% wider than the textblock, sticking out into the fore edge.

```
\makepagestyle{Ruled}
\makerunningwidth{Ruled}{1.1\textwidth}
\makeheadposition{Ruled}{flushright}{flushleft}{flushright}{flushleft}
\makeevenfoot{Ruled}{\thepage}{}{}
\makeoddfoot{Ruled}{}{}{\thepage}
\makeheadrule{Ruled}{1.1\textwidth}{\normalrulethickness}
\makepsmarks{Ruled}{\@ruledmarks}
\makeevenhead{Ruled}{\scshape\leftmark}{}{}
\makeoddhead{Ruled}{}{}{\rightmark}

```

`\headwidth` `\ps@companion` A *companion* pagestyle like the one in the *LaTeX Companion* series. We need the `\headwidth` length for this.

```
\newlength{\headwidth}

\makepagestyle{companion}
\setlength{\headwidth}{\textwidth}
  \addtolength{\headwidth}{\marginparsep}
  \addtolength{\headwidth}{\marginparwidth}
\makerunningwidth{companion}{\headwidth}
\makeheadrule{companion}{\headwidth}{\normalrulethickness}
\makeheadposition{companion}{flushright}{flushleft}{}{}
\makepsmarks{companion}{%
  \def\chaptermark##1{\markboth{##1}{##1}}    % left mark & right marks
  \def\sectionmark##1{\markright{%
    \ifnum \c@secnumdepth>\z@
      \thesection. \ %
    \fi
    ##1}}
  \def\tocmark{\markboth{\contentsname}{\contentsname}}
  \def\lofmark{\markboth{\listfigurename}{\listfigurename}}
  \def\lotmark{\markboth{\listtablename}{\listtablename}}
  \def\bibmark{\markboth{\bibname}{\bibname}}
  \def\indexmark{\markboth{\indexname}{\indexname}}}
```

And here's a version of the above `\makepsmarks` using `\createmark` and `\createplainmark`.

```
\makepsmarks{companion}{%
  \nouppercaseheads
  \createmark{chapter}{both}{nonumber}{}{}
  \createmark{section}{right}{shownumber}{}{. \space}
  \createplainmark{toc}{both}{\contentsname}
  \createplainmark{lof}{both}{\listfigurename}
  \createplainmark{lot}{both}{\listtablename}
  \createplainmark{bib}{both}{\bibname}
  \createplainmark{index}{both}{\indexname}
  \createplainmark{glossary}{both}{\glossaryname}}
\makeevenhead{companion}{\normalfont\bfseries\thepage}{}%
                        {\normalfont\bfseries\leftmark}
\makeoddhead{companion}{\normalfont\bfseries\rightmark}{}%
                       {\normalfont\bfseries\thepage}

```

Another pagestyle called *showlocs* can be used to show the locations of the header, footer and textblock. I would expect that this would be mainly used as an example for authors to create their own similar styles.

`\ifshowheadfootloc` `\showheadfootloctrue` `\showheadfootlocfalse` `\showheadfootlocon` `\showheadfootlocoff` `\ifshowtextblockloc` `\showtextblocklocfalse` `\showtextblocklocon` `\showtextblocklocoff`

Booleans controlling the appearance, or not, of the header/footer lines and text frame in *showlocs*.

```
\newif\ifshowheadfootloc
  \showheadfootloctrue
\newcommand*{\showheadfootlocon}{\showheadfootloctrue}
\newcommand*{\showheadfootlocoff}{\showheadfootlocfalse}
\newif\ifshowtextblockloc
  \showtextblocklocfalse
\newcommand*{\showtextblocklocon}{\showtextblocklocfalse}
\newcommand*{\showtextblocklocoff}{\showtextblocklocfalse}

```

`\framepichead`

For producing a zero-sized picture of a line at the base of the header. It is meant to be used as the left part of the header for a pagestyle.

```
\newcommand*{\framepichead}{%
\ifshowheadfootloc
  \begin{picture}(0,0)
    \unitlength 1pt
    \put(0,0){\line(1,0){\strip@pt\textwidth}}
  \end{picture}%
\fi}

```

`\framepictextfoot`

For producing a zero-sized picture of a line at the base of the footer and a frame around the text block. It is meant to be used as the left part of a footer for a pagestyle.

```
\newcommand*{\framepictextfoot}{%
  \begin{picture}(0,0)
    \unitlength 1pt
```

```
    \ifshowheadfootloc
      \put(0,0){\line(1,0){\strip@pt\textwidth}}
    \fi
    \ifshowtextblockloc
      \put(0,\strip@pt\footskip)%
        {\framebox(\strip@pt\textwidth,\strip@pt\textheight){}}
    \fi
  \end{picture}}

```

`\ps@showlocs` The *showlocs* pagestyle. This is more to show what can be done.

```
\makepagestyle{showlocs}
\makeevenhead{showlocs}{\framepichead\thepage}{\thepage}{\thepage}
\makeoddhead{showlocs}{\framepichead\thepage}{\thepage}{\thepage}
\makeevenfoot{showlocs}{\framepictextfoot\thepage}{\thepage}{\thepage}
\makeoddfoot{showlocs}{\framepictextfoot\thepage}{\thepage}{\thepage}

```

`\fixheaderwidths` The companion pagestyle, at least, needs adjusting (at `\fixthelayout` time) if the `\textwidth` has changed from its initial value.

```
\newcommand*{\fixheaderwidths}{%
  % companion pagestyle
  \setlength{\headwidth}{\textwidth}
    \addtolength{\headwidth}{\marginparsep}
    \addtolength{\headwidth}{\marginparwidth}
  \makerunningwidth{companion}{\headwidth}
  \makeheadrule{companion}{\headwidth}{\normalrulethickness}
  \makefootrule{companion}{\textwidth}{\footruleheight}{\footruleskip}
}

```

9.3 Page numbering

The kernel includes the `\pagenumbering` command for setting the style (arabic, roman, etc.) of the page numbers, and at the same time it resets the page counter. I want a version that resets the style but not the number[13].

`\pagenumbering` `\pagenumbering{`⟨*style*⟩`}` is the normal version whereas the starred version does
`\pagenumbering*` not reset the counter.

```
\renewcommand{\pagenumbering}{%
  \@ifstar{\@smempnum}{\@mempnum}}

```

`\@smempnum`

```
\newcommand{\@smempnum}[1]{%
  \gdef\thepage{\csname @#1\endcsname \c@page}}
```

[13] Added to meet a request by Daniel Richard G. (`skunk@mit.edu`) September 2001.

\@mempnum

```
\newcommand{\@mempnum}[1]{%
  \@smempnum{#1}\global\c@page \@ne}
```

\c@storedpagenumber A counter to store the page number.

```
\newcounter{storedpagenumber}
  \setcounter{storedpagenumber}{1}
```

\savepagenumber \restoregenumber `\savepagenumber` saves the current page number and `\restorepagenumber` sets the page number to the stored value.

```
\newcommand{\savepagenumber}{\global\c@storedpagenumber \c@page}
\newcommand{\restorepagenumber}{\global\c@page \c@storedpagenumber}
```

10 Non-traditional spacing

10.1 Double spacing

This is an embedding and extension of the code from the setspace package, with names changed.

\setSpacing `\setSpacing{`⟨*num*⟩`}` effectively increases the `\baselineskip` to ⟨*num*⟩`*\baselineskip`. (In package was `\setstretch`).

```
\newcommand*{\setSpacing}[1]{%
  \def\baselinestretch{#1}%
  \@currsize}
```

\setSingleSpace `\setSingleSpace{`⟨*num*⟩`}` effectively increases the `\baselineskip` for single spacing to ⟨*num*⟩`*\baselineskip` (⟨*num*⟩ should be close to 1.0). (In package was `\SetSinglespace`).

```
\newcommand*{\setSingleSpace}[1]{%
  \def\m@m@singlespace{#1}}
\setSingleSpace{1}
```

\SingleSpacing `\SingleSpacing` starts single spacing. (In package was `\singlespacing`).

```
\newcommand*{\SingleSpacing}{%
  \setSpacing{\m@m@singlespace}%
  \vskip\baselineskip% correction for coming into single spacing
}
\SingleSpacing
```

\OnehalfSpacing `\OnehalfSpacing` starts 'one and a half' spacing, which to most thesis nitpickers will look like double spacing. (In package was `\onehalfspacing`).

```
\newcommand*{\OnehalfSpacing}{
  \setSpacing{1.25}% default (10pt)
  \ifcase \@ptsize \relax   % 10pt
    \setSpacing{1.25}%
  \or%  11pt
    \setSpacing{1.213}%
  \or%  12pt
    \setSpacing{1.241}%
  \or\or% 14pt
    \setSpacing{1.20}%
  \or\or\or% 17pt
    \setSpacing{1.16}%
  \or\or% 9pt
    \setSpacing{1.35}%
  \else% the extended sizes
    \setSpacing{1.16}%
  \fi}

```

`\DoubleSpacing` `\DoubleSpacing` starts double spacing, which to most thesis nitpickers will look far too spaced out. (In package was `\doublespacing`).

```
\newcommand*{\DoubleSpacing}{
  \setSpacing{1.667}% default (10pt)
  \ifcase \@ptsize \relax   % 10pt
    \setSpacing{1.667}%
  \or%  11pt
    \setSpacing{1.618}%
  \or%  12pt
    \setSpacing{1.655}%
  \or\or% 14pt
    \setSpacing{1.60}%
  \or\or\or% 17pt
    \setSpacing{1.545}%
  \or\or% 9pt
    \setSpacing{1.8}%
  \else%       larger sizes
    \setSpacing{1.5}%
  \fi}

```

`\@setsize` Modify the kernel command. check what this does!!!!!!!!

```
\renewcommand*{\@setsize}[4]{%
  \@nomath#1%
  \let\@currsize#1%
  \baselineskip #2%
  \baselineskip \baselinestretch\baselineskip
  \parskip \baselinestretch\parskip
  \setbox\strutbox \hbox{%
    \vrule height.7\baselineskip
           depth .3\baselineskip
```

```
3042            width \z@}%
3043   \skip\footins \baselinestretch\skip\footins
3044   \normalbaselineskip\baselineskip#3#4}
3045
```

`SingleSpace` Environment form of `\SingleSpacing`. (In package was `singlespace`).

```
3046 \newenvironment{SingleSpace}{%
3047   \vskip\baselineskip
3048   \setSpacing{\m@m@singlespace}%
3049   \vskip -\baselineskip
3050 }{\par}
3051
```

`SingleSpace*` Don't use this; it's only here to match the setspace package. (In package was `singlespace*`).

```
3052 \newenvironment{SingleSpace*}{%
3053 %%  \vskip\baselineskip
3054   \setSpacing{\m@m@singlespace}%
3055   \vskip 0.5\baselineskip
3056 }{\vskip -0.5\baselineskip}
3057
```

`\m@mrestore@spacing`

```
3058 \newcommand*{\m@mrestore@spacing}{%
3059   \par
3060   \vskip \parskip
3061   \vskip \baselineskip
3062   \endgroup
3063   \vskip -\parskip
3064   \vskip -\baselineskip}
3065
```

`Spacing` `\begin{Spacing}{num}` increases the `\baselineskip` to `num*\baselineskip`. (In package was `spacing`).

```
3066 \newenvironment{Spacing}[1]{%
3067   \par
3068   \begingroup
3069     \setSpacing{#1}}{\m@mrestore@spacing}
3070
```

`OnehalfSpace` Environment form of `\OnehalfSpacing`. (In package was `onehalfspace`).

```
3071 \newenvironment{OnehalfSpace}{%
3072   \begingroup
3073     \OnehalfSpacing}{\m@mrestore@spacing}
3074
```

`DoubleSpace` Environment form of `\DoubleSpacing`. (In package was `doublespace`).

```
3075 \newenvironment{DoubleSpace}{%
3076   \begingroup
```

```
3077     \DoubleSpacing}{\m@mrestore@spacing}
3078
```

Deal with spacing around displays.

`\memdskipstretch` `\setDisplayskipStretch{`⟨*num*⟩`}` changes space around displays by the factor
`\setDisplayskipStretch` (1+⟨*num*⟩). `\noDisplaydkipStretch` keeps the regular spacing around displays.
`\noDisplayskipStretch` (In package were `\setdisplayskipstretch` and the `nodisplayskipstretch` option).

```
3079 \newcommand*{\memdskipstretch}{0.0}
3080 \newcommand*{\setDisplayskipStretch}[1]{%
3081   \renewcommand*{\memdskipstretch}{#1}}
3082 \newcommand*{\noDisplayskipStretch}{\setDisplayskipStretch{0.0}}
3083
```

`\memdskips` Macro added to the kernel hook `\everydisplay`, changing the settings of displayskips

```
3084 \newcommand*{\memdskips}{%
3085   \advance\abovedisplayskip \memdskipstretch\abovedisplayskip
3086   \advance\belowdisplayskip \memdskipstretch\belowdisplayskip
3087   \advance\abovedisplayshortskip \memdskipstretch\abovedisplayshortskip
3088   \advance\belowdisplayshortskip \memdskipstretch\belowdisplayshortskip}
```

`\everydisplay`

```
3089 \everydisplay\expandafter{%
3090   \the\everydisplay
3091   \memdskips}
3092
```

`\@xfloat` Reset `\baselinestretch` in floats.

```
3093 \let\m@m@xfloat\@xfloat
3094 \def\@xfloat #1[#2]{%
3095   \m@m@xfloat #1[#2]%
3096   \def\baselinestretch{\m@m@singlespace}%
3097   \normalsize}
3098
```

The extra spacing does not add space before and after a minipage. The solution, hinted at by Donald Arseneau, is to create a new environment. I came up with a solution which DA then much improved upon (CTT *setspace and minipages*, 2006/11/28). This is like minipage from an author's view, except that it is always top positioned and acts like a paragraph.

`\memPD`

```
3099 \newdimen\memPD
```

`vminipage` The user view, just as for `minipage`, but ensures the `t` position and as a new paragraph. It is like `minipage` but with better fore and aft spacing.

```
\newenvironment{vminipage}{%
  \par
  \@ifnextchar[%]
    \@ivminipage
    {\@iiiminipage t\relax[s]}
}{%
  \par\global\memPD=\prevdepth
  \endminipage
  \par
  \kern-\memPD%     no pagebreak allowed here
  \hbox{\vrule depth \memPD width \z@}}

```

`\@ivminipage` Deal with first optional argument to `vminipage`.

```
\def\@ivminipage[#1]{%
 \@ifnextchar[%]
   {\@iiminipage{t}}{\@iiiminipage{t}\relax[s]}}
```

10.2 Abnormal parskips

Non-zero, positive `\parskip`, which is not to be encouraged. Code based on Robin Fairbairns parskip package and the NTG classes.

`\ifm@mnzpskip` `\ifm@mnzpskip` is TRUE if `\parskip` set to anything but 0pt.
`\m@mnzpskiptrue`
`\m@mnzpskipfalse`
`\traditionalparskip` `\traditionalparskip` sets `\parskip` to 0pt. `\abnormalparskip{`⟨*length*⟩`}` sets
`\m@mabparskip` `\parskip` to ⟨*length*⟩. `\nonzeroparskip` sets `\parskip` to a non-zero value that
`\abnormalparskip` might be not too bad (any non-zero `\parskip` is not good).
`\nonzeroparskip`

```
\newif\ifm@mnzpskip
\newcommand*{\traditionalparskip}{%
  \setlength\parskip{0\p@ \@plus \p@}
  \m@mnzpskipfalse}
\newskip\m@mabparskip
\newcommand*{\abnormalparskip}[1]{%
  \setlength{\parskip}{#1}\m@mabparskip=#1\relax
  \m@mnzpskiptrue}
\newcommand*{\nonzeroparskip}{\abnormalparskip{%
  0.5\baselineskip
  \@plus .1\baselineskip \@minus .1\baselineskip% NTG
%%  0.5/baselineskip \@plus 2pt% RF
}}
\traditionalparskip

```

`\@minpagerestore` This needs extending: here's what NTG does, but I've put the appropriate code later.

```
\providecommand*{\@minipagerestore}{%
  \parskip=.5\baselineskip \@plus .1\baselineskip \@minus .1\baselineskip}
```

11 Titles

For books the title is usually designed specifically for the particular work, so originally the class did not provide a `\maketitle` command or any of the `\title` and other commands that normally go along with this. After some thought I decided to add in the code from the titling package [Wil01g]. However, in this case life is a bit simpler as there is no pre-existing class code.

`\pretitle` `\@bspretitle` `\posttitle` `\@bsposttitle` `\preauthor` `\@bspreauthor` `\postauthor` `\@bspostauthor` `\predate` `\@bspredate` `\postdate` `\@bspostdate`

To provide some flexibilty in the titling style of the document, user level commands are provided that can be changed to reconfigure the appearance resulting from `\maketitle`.

```
\newcommand{\pretitle}[1]{\def\@bspretitle{#1}}
\newcommand{\posttitle}[1]{\def\@bsposttitle{#1}}
\newcommand{\preauthor}[1]{\def\@bspreauthor{#1}}
\newcommand{\postauthor}[1]{\def\@bspostauthor{#1}}
\newcommand{\predate}[1]{\def\@bspredate{#1}}
\newcommand{\postdate}[1]{\def\@bspostdate{#1}}
```

These are defined initially to mimic the normal LaTeX style.

```
  \pretitle{\begin{center}\LARGE}
  \posttitle{\par\end{center}\vskip 0.5em}
  \preauthor{\begin{center}
    \large \lineskip .5em%
    \begin{tabular}[t]{c}}
  \postauthor{\end{tabular}\par\end{center}}
  \predate{\begin{center}\large}
  \postdate{\par\end{center}}
```

`\maketitlehooka` `\maketitlehookb` `\maketitlehookc` `\maketitlehookd`

The four hooks which will be called by `\maketitle`. These are initially vacuous.

```
\newcommand{\maketitlehooka}{}
\newcommand{\maketitlehookb}{}
\newcommand{\maketitlehookc}{}
\newcommand{\maketitlehookd}{}
```

`\thanksmarkseries` `\@bsmarkseries` `\symbolthanksmark`

These are for specifying the kind of series for thanks markers.

```
\newcommand{\thanksmarkseries}[1]{%
  \def\@bsmarkseries{\renewcommand{\thefootnote}%
                      {\@nameuse{#1}{footnote}}}}
\newcommand{\symbolthanksmark}{\thanksmarkseries{\fnsymbol}}
```

`\continuousmarks` `\@bscontmark`

These are for (non) zeroing of the footnote counter.

```
\newcommand{\@bscontmark}{\setcounter{footnote}{0}}
\newcommand{\continuousmarks}{\def\@bscontmark{}}
```

`\thanksheadextra` `\@bsthanksheadpre` `\@bsthanksheadpost`

These are for inserting stuff before and after a mark in the titling.

```
\newcommand{\thanksheadextra}[2]{%
  \def\@bsthanksheadpre{#1}%
  \def\@bsthanksheadpost{#2}}
```

`\thanksmark` This adds a thanks mark. The `\footnotemark` could have been used directly but it is fragile in a moving argument.

```
\DeclareRobustCommand{\thanksmark}[1]{\footnotemark[#1]}
```

`\thanksgap` This specifies some horizontal space.

```
\newcommand{\thanksgap}[1]{\hspace{#1}}
```

`\tamark` This stores the current definition of `\@thefnmark`. For some reason using `\@thefnmark` directly only gave the last value.

```
\newcommand{\tamark}{\@thefnmark}

```

`\thanksmarkwidth` A length determining the size of the box for typesetting a thanks marker.

```
\newlength{\thanksmarkwidth}
```

`\thanksmarksep` A length determining the inset of thanks footnotes.

```
\newlength{\thanksmarksep}
```

`\thanksmarkstyle` `\thanksscript` is a wrapper round the actual mark stuff to be typeset. The user can define this via `\thanksmarkstyle{`⟨*code*⟩`}`. The default is a superscript mark.

```
\newcommand{\thanksmarkstyle}[1]{\def\thanksscript##1{#1}}
\thanksmarkstyle{\textsuperscript{#1}}
```

`\makethanksmarkhook` A vacuous macro used as a hook into `\makethanksmark`.

```
\newcommand{\makethanksmarkhook}{}

```

`\thanksfootmark` This typesets the thanks footnote mark.

```
\newcommand{\thanksfootmark}{%
  \ifdim\thanksmarkwidth < \z@
```

Negative width, mark is in the margin.

```
    \llap{\hb@xt@ -\thanksmarkwidth{%
          \hss\normalfont\thanksscript{\tamark}}%
          \hspace*{-\thanksmarkwidth}}%
  \else
    \ifdim\thanksmarkwidth = \z@
```

Zero width, mark is at (inside) the margin.

```
      {\normalfont\thanksscript{\tamark}}%
    \else
```

Positive width.

```
      \hb@xt@\thanksmarkwidth{\hss\normalfont\thanksscript{\tamark}}%
    \fi
  \fi}
```

`\makethanksmark` This sets the general indentations for the thanks footnote, and typesets the mark. The code is a simplified version of that for typesetting ToC entries.

```
\newcommand{\makethanksmark}{%
  \leavevmode%
  \parindent 1em\noindent
%%%  \leftskip\thanksmarksep\relax
  \memRTLleftskip\thanksmarksep\relax
%%%  \advance\leftskip\thanksmarkwidth
  \advance\memRTLleftskip\thanksmarkwidth
%%%  \null\nobreak\hskip-\leftskip\relax
  \null\nobreak\hskip-\memRTLleftskip\relax
  \makethanksmarkhook\relax
  \thanksfootmark}
```

`\usethanksrule` Simple macros that let `\footnoterule` to another rule definition.

`\cancelhanksrule`

```
\newcommand{\usethanksrule}{\let\footnoterule\thanksrule}
\newcommand{\cancelthanksrule}{\let\footnoterule\@bsfootnoterule}
```

Now set up the rest of the thanks defaults, styling having been done earlier.

```
\thanksmarkseries{fnsymbol}  % symbols
\thanksheadextra{}{}
\setlength{\thanksmarkwidth}{1.8em}
\setlength{\thanksmarksep}{-\thanksmarkwidth}
```

`\thanksrule` These are saved versions of the `\footnoterule` definition as it is at the end of the preamble.

`\@bsfootnoterule`

```
\AtBeginDocument{%
  \let\thanksrule\footnoterule
  \let\@bsfootnoterule\footnoterule
}
```

`\droptitle` A titling block has `\droptitle` amount of additional vertical space above it (normally zero).

```
\newlength{\droptitle}
\setlength{\droptitle}{0pt}
```

`\maketitle` The following is a modification of `\maketitle` as in the article, report, and book classes. It sets the pagestyle to *title*.

```
\newcommand{\maketitle}{\par
  \begingroup
    \@bsmarkseries
    \def\@makefnmark{\@textsuperscript{%
       \normalfont\@bsthanksheadpre \tamark \@bsthanksheadpost}}%
    \long\def\@makefntext##1{\makethanksmark ##1}
    \if@twocolumn
      \ifnum \col@number=\@ne
        \@maketitle
      \else
        \twocolumn[\@maketitle]%
      \fi
    \else
      \ifdim\pagetotal>\z@
        \newpage
      \fi
      \global\@topnum\z@
      \@maketitle
    \fi
    \thispagestyle{title}\@thanks
  \endgroup
  \@bscontmark  %  \setcounter{footnote}{0}%
  }
\aliaspagestyle{title}{plain}

```

`\@mem@titlefootkill` `\@mem@titlefootkill{`⟨*note*⟩`}` Warn about footnotes in titles.

```
\newcommand*{\@mem@titlefootkill}[1]{%
  \@memwarn{Do not use \string\footnote\space in
            \string\maketitle.\MessageBreak
            Use \protect\thanks\space instead}}

```

`\@maketitle` Our version of `\@maketitle`. Footnotes are killed in the title; see the thread '`\title`, `\author` and `\footnote` feature in memoir class', February 2003.

```
\newcommand{\@maketitle}{%
  \let\footnote\@mem@titlefootkill
  \ifdim\pagetotal>\z@
    \newpage
  \fi
  \null
  \vskip 2em%
        \vspace*{\droptitle}
  \maketitlehooka
  {\@bspretitle \@title \@bsposttitle}
  \maketitlehookb
  {\@bspreauthor \@author \@bspostauthor}
  \maketitlehookc
  {\@bspredate \@date \@bspostdate}
  \maketitlehookd
```

```
  \par
  \vskip 1.5em}
```

`titlingpage` The `titlingpage` environment sets the pagestyle to be *titlingpage*, disables the footnote rule and ensures that the page is single column. At the end it switches back to twocolumn if neccessary, and then starts a new page as number 1.

```
\newenvironment{titlingpage}%
  {\let\footnoterule\relax
   \let\footnotesize\small
   \if@twocolumn
     \@restonecoltrue\onecolumn
   \else
     \@restonecolfalse
   \fi
   \thispagestyle{titlingpage}%
   \setcounter{page}{\@ne}%
  }{%
   \thispagestyle{titlingpage}%
   \if@restonecol \twocolumn \fi
   \if@twoside \cleardoublepage \else \clearpage \fi
   \setcounter{page}{\@ne}}
\aliaspagestyle{titlingpage}{empty}
```

`\emptythanks` This macro discards all prior `\thanks` texts.

```
\newcommand{\emptythanks}{\global\let\@thanks\@empty}
```

`\andnext` The kernel `\and` macro puts space between author's names. The `\andnext` macro puts a newline between the names.

```
\newcommand*{\andnext}{%
  \end{tabular}\\ \begin{tabular}[t]{c}}
```

`\@bsmtitlempty` `\@bsmtitlempty` is a helper macro to save some macro space. It empties some elements of `\maketitle`.

```
\newcommand{\@bsmtitlempty}{%
  \global\let\maketitle\relax
  \global\let\@maketitle\relax
  \global\let\title\relax
  \global\let\author\relax
  \global\let\date\relax
  \global\let\thanksmarkseries\relax
  \global\let\thanksheadextra\relax
  \global\let\thanksfootextra\relax
  \global\let\thanksmark\relax
  \global\let\thanksgap\relax}
```

\keepthetitle This macro undefines all the titling commands except for \thetitle, \theauthor and \thedate.

```
\newcommand{\keepthetitle}{%
  \@bsmtitlempty
  \global\let\thanks\relax
  \global\let\and\relax
  \global\let\andnext\relax
  \global\let\@thanks\@empty
  \global\let\@title\@empty
  \global\let\@author\@empty
  \global\let\@date\@empty}

```

\killtitle \killtitle undefines the remaining macros of \maketitle.

```
\newcommand{\killtitle}{%
  \keepthetitle
  \global\let\thetitle\relax
  \global\let\theauthor\relax
  \global\let\thedate\relax}

```

\thetitle \theauthor \thedate In order to make the \title, etc., values available for printing their definitions (in ltsect.dtx) need extending to save their arguments. We have to make sure that extraneous material, like \thanks, is excluded from the saved texts.

```
\addtoiargdef{\title}{%
  \begingroup\let\footnote\@gobble}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\thanksmark}[1]{}
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\thetitle{#1}
  \endgroup\endgroup}
\addtoiargdef{\author}{%
  \begingroup\let\footnote\@gobble}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\and}{\unskip, }
    \renewcommand{\andnext}{\unskip, }
    \renewcommand{\thanksmark}[1]{}
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\theauthor{#1}
  \endgroup\endgroup}
\addtoiargdef{\date}{%
  \begingroup\let\footnote\@gobble}{%
  \begingroup
    \renewcommand{\thanks}[1]{}
    \renewcommand{\thanksmark}[1]{}
    \renewcommand{\thanksgap}[1]{}
    \protected@xdef\thedate{#1}
```

```
3334  \endgroup\endgroup}
3335
```

12 Parts, chapters and other divisions

12.1 Building blocks

The definitions in this part of a class file usually make use of two internal macros, `\@startsection` and `\secdef`. To understand what is going on here, we describe their syntax.

`\@startsection` The macro `\@startsection` has 6 required arguments, optionally followed by a $*$, an optional argument and a required argument:
`\@startsection`⟨*name*⟩⟨*level*⟩⟨*indent*⟩⟨*beforeskip*⟩⟨*afterskip*⟩⟨*style*⟩ optional *

`[`⟨*altheading*⟩`]`⟨*heading*⟩
It is a generic command to start a section, the arguments have the following meaning:

⟨***name***⟩ The name of the user level command, e.g., 'section'.

⟨***level***⟩ A number, denoting the depth of the section — e.g., chapter = 0, section = 1, etc. A section number will be printed if and only if ⟨*level*⟩ <= the value of the *secnumdepth* counter.

⟨***indent***⟩ The indentation of the heading from the left margin

⟨***beforeskip***⟩ The absolute value of this argument gives the skip to leave above the heading. If it is negative, then the paragraph indent of the text following the heading is suppressed.

⟨***afterskip***⟩ If positive, this gives the skip to leave below the heading, else it gives the skip to leave to the right of a run-in heading.

⟨***style***⟩ Commands to set the style of the heading.

$*$ When this is missing the heading is numbered and the corresponding counter is incremented.

⟨***altheading***⟩ Gives an alternative heading to use in the table of contents and in the running heads. This should be present when the $*$ form is used.

⟨***heading***⟩ The heading of the new section.

A sectioning command is normally defined to `\@startsection` and its first six arguments.

`\secdef` The macro `\secdef` can be used when a sectioning command is defined without using `\@startsection`. It has two arguments:
`\secdef`⟨*unstarcmds*⟩⟨*starcmds*⟩

⟨***unstarcmds***⟩ Used for the normal form of a sectioning command.

⟨***starcmds***⟩ Used for the ∗-form of a sectioning command.

You can use `\secdef` as follows:

```
\def\chapter { ... \secdef \CMDA \CMDB }
\def\CMDA    [#1]#2{ ... }  % Command to define
                            % \chapter[...]{...}
\def\CMDB    #1{ ... }      % Command to define
                            % \chapter*{...}
```

`\@hangfrom` Internally the `\@startsection` macro uses `\@hangfrom{NUM}`, where `NUM` is the sectional number, to produce a hanging paragraph. That is, the second and later lines of a multiline title are indented from the left margin by the width of the number. The definition of `\@hangfrom`, from `ltsect.dtx`, is:

```
\def\@hangfrom#1{\setbox\@tempboxa\hbox{{#1}}%
  \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

To get a normal paragraphed title you can do:
`\renewcommand{\@hangfrom}[1]{#1}`
or as a block paragraph:
`\renewcommand{\@hangfrom}[1]{\noindent #1}}`

`\@seccntformat` The `\@startsection` macro also uses `\@seccntformat{NUM}` to format the section number, including the space after it. Its definition, from `ltsect.dtx`, is

```
\def\@seccntformat#1{\csname the#1\endcsname\quad}
```

This is the command to change if you need different number formatting. For example the combination

```
\renewcommand{\@seccntformat}[1]{\llap{\csname the#1\endcsname\quad}}
\renewcommand{\@hangfrom}[1]{\noindent #1}
```

will hang the section numbers in the margin.
The values used for the document division levels are the same as defined by the standard LaTeX classes, with the exception of the book division which is a memoir addition, and are given in Table 4.

12.2 Mark commands

`\bookpagemark` `\partmark` `\chaptermark` `\sectionmark` `\subsectionmark` `\subsubsectionmark` `\paragraphmark` `\subparagraphmark`

The default initialisations of the `\...mark` commands for use in the pagestyles. Most are already defined in the kernel but they are all noted here.

```
3336 \newcommand*{\bookpagemark}[1]{}
3337 \newcommand*{\partmark}[1]{}
3338 \newcommand*{\chaptermark}[1]{}
```

Table 4: Document division levels

Division	Level
book	-2
part	-1
chapter	0
section	1
subsection	2
subsubsection	3
paragraph	4
subparagraph	5

```
% \newcommand*{\sectionmark}[1]{}
% \newcommand*{\subsectionmark}[1]{}
% \newcommand*{\subsubsectionmark}[1]{}
% \newcommand*{\paragraphmark}[1]{}
% \newcommand*{\subparagraphmark}[1]{}
```

`\bibmark` Marks for the bibliography, which may be filled with `\bibname`.

```
\newcommand*{\bibmark}{}
```

`\indexmark` Marks for the index, which may be filled with `\indexname`.

```
\newcommand*{\indexmark}{}
```

`\glossarymark` Marks for the glossary, which may be filled with `\glossaryname`.

```
\newcommand*{\glossarymark}{}
```

12.3 Define Counters

`\c@secnumdepth` The value of the counter *secnumdepth* gives the depth of the highest-level sectioning command that is to produce section numbers.

```
\setcounter{secnumdepth}{2}
```

`\c@book` `\c@part` `\c@chapter` `\c@section` `\c@subsection` `\c@subsubsection` `\c@paragraph` `\c@subparagraph` These counters are used for the sectioning numbers. The macro `\newcounter{`⟨*newctr*⟩`}[`⟨*oldctr*⟩`]` defines⟨*newctr*⟩ to be a counter, which is reset to zero when counter ⟨*oldctr*⟩ is stepped. Counter ⟨*oldctr*⟩ must already be defined.
Book, part and chapter are the top level document divisions.

```
\newcounter{book} \setcounter{book}{0}
\newcounter{part} \setcounter{part}{0}
\newcounter{chapter} \setcounter{chapter}{0}
```

The lower level divisions get reset by higher level divisions.

```
\newcounter{section}[chapter]
\newcounter{subsection}[section]
```

```
\newcounter{subsubsection}[subsection]
\newcounter{paragraph}[subsubsection]
\newcounter{subparagraph}[paragraph]
```

\thebook
\thepart
\thechapter
\thesection
\thesubsection
\thesubsubsection
\theparagraph
\thesubparagraph

For any counter *CTR*, \theCTR is a macro that defines the printed version of counter *CTR*. It is defined in terms of the following macros:
\arabic{⟨*COUNTER*⟩} prints the value of ⟨*COUNTER*⟩ as an arabic numeral.
\roman{⟨*COUNTER*⟩} prints the value of ⟨*COUNTER*⟩ as a lowercase roman numeral.
\Roman{⟨*COUNTER*⟩} prints the value of ⟨*COUNTER*⟩ as an uppercase roman numeral.
\alph{⟨*COUNTER*⟩} prints the value of ⟨*COUNTER*⟩ as a lowercase letter: 1 = a, 2 = b, etc.
\Alph{⟨*COUNTER*⟩} prints the value of ⟨*COUNTER*⟩ as an uppercase letter: 1 = A, 2 = B, etc.

```
\renewcommand*{\thebook}{\@Roman\c@book}
\renewcommand*{\thepart}{\@Roman\c@part}
\renewcommand*{\thechapter}{\@arabic\c@chapter}
\renewcommand*{\thesection}{\thechapter.\@arabic\c@section}
\renewcommand*{\thesubsection}{%
              \thesection.\@arabic\c@subsection}
\renewcommand*{\thesubsubsection}{%
              \thesubsection.\@arabic\c@subsubsection}
\renewcommand*{\theparagraph}{%
              \thesubsubsection.\@arabic\c@paragraph}
\renewcommand*{\thesubparagraph}{%
              \theparagraph.\@arabic\c@subparagraph}
```

\@chapapp

\@chapapp is initially defined as \chaptername. The \appendix command redfines it as \appendixname.

```
\newcommand{\@chapapp}{\chaptername}
```

12.4 Front, main and back matter

These are the three main logical divisions in a book. As noted earlier, the boolean \if@mainmatter is TRUE iff the main matter is being processed. Chapters will be unnumbered when \if@mainmatter is FALSE.

\frontmatter
\frontmatter*

The \frontmatter command starts roman numbering and turns off chapter numbering. It ensures that lower level divisions will not have chapter numbers, nor will figures or tables. It also ensures that the next page will be recto. The starred version makes no changes to the page numbering[14].

```
\newcommand{\frontmatter}{%
  \@ifstar{\@smemfront}{\@memfront}}
```

[14] The starred versions were added to meet a request by Daniel Richard G. (skunk@mit.edu) in September 2001.

\@memfront@floats actoring out the float counter domination, in case someone wants something added or changed

```
\newcommand\@memfront@floats{%
  \counterwithout{figure}{chapter}
  \counterwithout{table}{chapter}
}
```

\@smemfront

```
\newcommand{\@smemfront}{%
  \cleardoublepage
  \@mainmatterfalse
  \setcounter{secnumdepth}{-10}
  \@memfront@floats
}
```

\@memfront

```
\newcommand{\@memfront}{%
  \@smemfront\pagenumbering{roman}}

```

\mainmatter
\mainmatter* The \mainmatter command starts arabic numbering and turns on chapter numbering. It sets numbering to the normal state. It also ensures that the next page will be recto. The starred version does not change the page numbering. Romano Giannetti suggested that \mainmatter (and \backmatter) should be almost no-ops with the article option.

```
\newcommand{\mainmatter}{%
  \@ifstar{\@smemmain}{\@memmain}}
```

\@memmain@floats actoring out the float counter domination, in case someone wants something added or changed

```
\newcommand\@memmain@floats{%
   \counterwithin{figure}{chapter}
   \counterwithin{table}{chapter}
}
```

\@smemmain If \frontmatter and \mainmatter are used with the article option, then a page numbering / margin problem can occur if the frontmatter ends on an odd page — the first mainmatter next page is odd as well which throws puts two odd-margined pages together.
One fix from Morten is to always \cleardoublepage for twosided articles.
One fix from me is for the ...matter macros to just switch on/off section numbering.
Morten's is simpler to implement.

```
\newcommand*{\@smemmain}{%
  \@mainmattertrue
  \setcounter{secnumdepth}{\value{maxsecnumdepth}}
  \ifartopt
```

```
    \if@twoside
      \cleardoublepage
    \else
      \clearpage
    \fi
  \else
    \cleardoublepage
    \@memmain@floats
  \fi}
```

`\@memmain`

```
\newcommand{\@memmain}{%
  \@smemmain\pagenumbering{arabic}}
```

`\@memback@floats` actoring out the float counter domination, in case someone wants something added or changed

```
\newcommand\@memback@floats{%
    \counterwithout{figure}{chapter}
    \counterwithout{table}{chapter}
    \setcounter{figure}{0}
    \setcounter{table}{0}
}
```

`\backmatter` This command turns off chapter numbering but leaves the page numbering alone. It twiddles the numbering caption numbering. The back matter may start on any new page.

```
\newcommand{\backmatter}{%
  \ifartopt
    \clearpage
  \else
    \if@openright
      \cleardoublepage
    \else
      \clearpage
    \fi
  \fi
  \@mainmatterfalse
  \setcounter{secnumdepth}{-10}
  \ifartopt\else
    \@memback@floats
  \fi}
```

12.5 Book

This was added at the request of Frederic Connes who said that in Frency typography there was often a division above the `\part` level.

\theHbook
\toclevel@book

These are needed if the hyperref is used.

```
\newcommand*{\theHbook}{\arabic{book}}
\newcommand*{\toclevel@book}{-2}
```

\book
\book*

\book{⟨*title*⟩} starts a new book division called ⟨*title*⟩. The actual typesetting of the title is done by \@book or \@sbook.

```
\newcommand*{\book}{%
  \@setupbook
  \secdef\@book\@sbook}
```

\beforebookskip
\midbookskip
\afterbookskip

These three macros are the skips before, in the middle, and after the Book heading.

```
\newcommand*{\beforebookskip}{\null\vfil}
\newcommand*{\midbookskip}{\par\vskip 2\onelineskip}
\newcommand*{\afterbookskip}{\vfil\newpage}
```

\@setupbook

This macro does the work of setting up for the \book command. A single column page, normally recto, with a *book* pagestyle is started.

```
\newcommand{\@setupbook}{%
  \if@openright
    \cleardoublepage
  \else
    \clearpage
  \fi
  \thispagestyle{book}%
  \if@twocolumn
    \onecolumn
    \@tempswatrue
  \else
    \@tempswafalse
  \fi
  \beforebookskip}
```

\booknamefont
\booknumfont
\booktitlefont

These three macros specify the fonts for the book name and number, and for the book title, respectively.

```
\newcommand*{\booknamefont}{\normalfont\huge\bfseries}
\newcommand*{\booknumfont}{\normalfont\huge\bfseries}
\newcommand*{\booktitlefont}{\normalfont\Huge\bfseries}
```

\printbookname
\booknamenum
\printbooknum
\printbooktitle

Macros to print the various books of a Book heading.

```
\newcommand*{\printbookname}{\booknamefont \bookname}
\newcommand*{\booknamenum}{\space}
\newcommand*{\printbooknum}{\booknumfont \thebook}
\newcommand*{\printbooktitle}[1]{\booktitlefont #1}
```

\membookinfo
\membookstarinfo
```
\newcommand{\membookinfo}[3]{}
\newcommand{\membookstarinfo}[1]{}
```

\@book `\@book[⟨short⟩]{⟨long⟩}` typesets the title of a `\book` book. There is a number if `\secnumdepth` is greater than -3.

```
\long\def\@book[#1]#2{%
  \M@gettitle{#1}%
  \phantomsection
  \ifnum \c@secnumdepth >-3\relax
    \refstepcounter{book}%
    \addcontentsline{toc}{book}%
      {\protect\booknumberline{\thebook}#1}%
    \membookinfo{\thebook}{#1}{#2}%
  \else
    \addcontentsline{toc}{book}{#1}%
    \membookinfo{}{#1}{#2}%
  \fi
```

Empty the marks, center the title on the page, and set the normal font.

```
  \bookpagemark{#1}%
  {\centering
   \interlinepenalty \@M
   \normalfont
```

Print the number, if there is one, then the title below. Macro `\@endbook` tidies everything up at the end.

```
   \ifnum \c@secnumdepth >-3\relax
     \printbookname \booknamenum \printbooknum
     \midbookskip
   \fi
   \printbooktitle{#2}\par}%
  \@endbook}
```

\@sbook `\@sbook{⟨long⟩}` formats the title of a `\book*` book. It is simpler than `\@book` because there is no number to print.

```
\long\def\@sbook#1{%
  \M@gettitle{#1}%
  \phantomsection
  \membookstarinfo{#1}%
  {\centering
   \interlinepenalty \@M
   \normalfont
   \printbooktitle{#1}\par}%
  \@endbook}
```

\ifm@mnobooknewpage
\m@mnobooknewpagetrue
\m@mnobooknewpagefalse
\bookblankpage
\nobookblankpage

The declaration \nobookblankpage prevents \book from outputting a further (blank) page when it finishes. The default is set by \bookblankpage.

```
\newif\ifm@mnobooknewpage
  \m@mnobooknewpagefalse
\newcommand*{\bookblankpage}{\m@mnobooknewpagefalse}
\newcommand*{\nobookblankpage}{\m@mnobooknewpagetrue}
```

\bookpageend
\@endbook

This finishes off both \@book and \@sbook. Following a \nobookblankpage nothing is done, otherwise (after \bookblankpage) the current page is flushed, and an extra blank page is created if both twoside and openright are in effect. At the end, if necessary, two column mode is switched back on.

```
\newcommand*{\bookpageend}{\afterbookskip
  \ifm@mnobooknewpage
  \else
    \if@twoside
      \if@openright
        \null
        \thispagestyle{afterbook}%
        \newpage
      \fi
    \fi
  \fi
  \if@tempswa
    \twocolumn
  \fi}
\def\@endbook{\bookpageend}
```

\ps@book
\ps@afterbook

Pagestyles for the book page and the (blank) page after a book page.

```
\aliaspagestyle{book}{empty}
\aliaspagestyle{afterbook}{empty}
```

12.6 Part

\part

\part{⟨*title*⟩} starts a new Part called ⟨*title*⟩. The actual typesetting of the title is done by \@part or \@spart.

```
\newcommand{\part}{%
  \@setuppart
  \secdef\@part\@spart}
```

\beforepartskip
\midpartskip
\afterpartskip

These three macros are the skips before, in the middle, and after the Part heading.

```
\newcommand{\beforepartskip}{\null\vfil}
\newcommand{\midpartskip}{\par\vskip 2\onelineskip}
\newcommand{\afterpartskip}{\vfil\newpage}
```

\@setuppart This macro does the work of setting up for the \part command. A single column page, normally recto, with a *part* pagestyle is started.

```
\newcommand{\@setuppart}{%
  \if@openright
    \cleardoublepage
  \else
    \clearpage
  \fi
  \thispagestyle{part}%
  \if@twocolumn
    \onecolumn
    \@tempswatrue
  \else
    \@tempswafalse
  \fi
  \beforepartskip}

```

\partnamefont \partnumfont \parttitlefont These three macros specify the fonts for the part name and number, and for the part title, respectively.

```
\newcommand{\partnamefont}{\normalfont\huge\bfseries}
\newcommand{\partnumfont}{\normalfont\huge\bfseries}
\newcommand{\parttitlefont}{\normalfont\Huge\bfseries}

```

\printpartname \partnamenum \printpartnum \printparttitle Macros to print the various parts of a Part heading.

```
\newcommand{\printpartname}{\partnamefont \partname}
\newcommand{\partnamenum}{\space}
\newcommand{\printpartnum}{\partnumfont \thepart}
\newcommand{\printparttitle}[1]{\parttitlefont #1}

```

\mempartinfo \mempartstarinfo `\mempartinfo{\thepart}{short}{long}` `\mempartstarinfo{long}`

```
\newcommand{\mempartinfo}[3]{}
\newcommand{\mempartstarinfo}[1]{}

```

\@part `\@part[`⟨*short*⟩`]{`⟨*long*⟩`}` typesets the title of a \part part. There is a number if \secnumdepth is greater than -2.

```
\long\def\@part[#1]#2{%
  \M@gettitle{#1}%
  \phantomsection
  \ifnum \c@secnumdepth >-2\relax
    \refstepcounter{part}%
    \addcontentsline{toc}{part}%
      {\protect\partnumberline{\thepart}#1}%
    \mempartinfo{\thepart}{#1}{#2}%
```

```
  \else
    \addcontentsline{toc}{part}{#1}%
    \mempartinfo{}{#1}{#2}%
  \fi
```

Empty the marks, center the title on the page, and set the normal font.

```
  \partmark{#1}%
  {\centering
   \interlinepenalty \@M
   \normalfont
```

Print the number, if there is one, then the title below. Macro `\@endpart` tidies everything up at the end.

```
   \ifnum \c@secnumdepth >-2\relax
     \printpartname \partnamenum \printpartnum
     \midpartskip
   \fi
   \printparttitle{#2}\par}%
  \@endpart}

```

`\@spart` `\@spart{`⟨*long*⟩`}` formats the title of a `\part*` part. It is simpler than `\@part` because there is no number to print.

```
\long\def\@spart#1{%
  \M@gettitle{#1}%
  \phantomsection
  \mempartstarinfo{#1}%
  {\centering
   \interlinepenalty \@M
   \normalfont
   \printparttitle{#1}\par}%
  \@endpart}

```

`\ifm@mnopartnewpage` `\m@mnopartnewpagetrue` `\m@mnopartnewpagefalse` `\partblankpage` `\nopartblankpage` The declaration `\nopartblankpage` prevents `\part` from outputting a blank page when it finishes. The default is set by `\partblankpage`. This code was originally supplied by Frederic Connes.

```
\newif\ifm@mnopartnewpage
  \m@mnopartnewpagefalse
\newcommand*{\partblankpage}{\m@mnopartnewpagefalse}
\newcommand*{\nopartblankpage}{\m@mnopartnewpagetrue}

```

`\partpageend` `\@endpart` This finishes off both `\@part` and `\@spart`. Following a `\nopartblankpage` nothing is done, otherwise (after `\partblankpage`) the current page is flushed. In the standard book class if two-sided mode is on a blank page is then produced. I think that this looks odd when the openany option is in force, so here it only produces an extra blank page if both twoside and openright are in effect. At the end, if necessary, two column mode is switched back on.

```
\newcommand*{\partpageend}{\afterpartskip
  \ifm@mnopartnewpage
  \else
    \if@twoside
      \if@openright
        \null
        \thispagestyle{afterpart}%
        \newpage
      \fi
    \fi
  \fi
  \if@tempswa
    \twocolumn
  \fi}
\def\@endpart{\partpageend}

```

`\ps@afterpart` Pagestyle for the (blank) page after a part page.

```
\aliaspagestyle{afterpart}{empty}

```

12.7 Chapter

`\chapter` The command to start a new chapter. Chapters always start on a new page with a *chapter* pagestyle. Floats are not allowed at the top of the page. The typesetting is done by either `\@chapter` or `\@schapter`.

```
\newcommand\chapter{%
  \ifartopt\par\else
    \clearforchapter
    \thispagestyle{chapter}
    \global\@topnum\z@
  \fi
  \@afterindentfalse
  \@ifstar{\@m@mschapter}{\@m@mchapter}}

```

`\@m@mchapter` Intermediate and support macros for the extra optional argument to `\chapter`.
`\ch@pt@c` Have to do this long windedly otherwise dear old hyperref barfs.
`\m@m@empty` The code for two optional arguments is based on a posting to CTT by Robin Fairbairns (1997/04/12 *Re: Several optional arguments for macro?*).

```
\newcommand{\@m@mchapter}[1][]{%
  \def\ch@pt@c{#1}% capture first optional arg
  \@ifnextchar[{\@chapter}{\@chapter[]}%
}
\def\m@m@empty{\@empty}

```

`\memchapinfo` `\memchapinfo{num}{toc}{head}{full}`
`\memchapstarinfo`
`\memappchapinfo`
`\memappchapstarinfo`

\memchapstarinfo{short}{full}

```
\newcommand{\memchapinfo}[4]{}
\newcommand{\memchapstarinfo}[2]{}
\newcommand{\memappchapinfo}[4]{}
\newcommand{\memappchapstarinfo}[2]{}
```

\ifm@mpn@new@chap
\m@mpn@new@chapfalse
\m@mpn@new@chaptrue
\ifm@mpn@new@schap
\m@mpn@new@schapfalse
\m@mpn@new@schaptrue

These are for supporting the [age/end notes, flagging that a \chapter(*) has been called.

```
\newif\ifm@mpn@new@chap
  \m@mpn@new@chapfalse
\newif\ifm@mpn@new@schap
  \m@mpn@new@schapfalse
```

\@chapter
\f@rtoc
\f@rhdr
\f@rbody

\@chapter[⟨*tocmark*⟩]{⟨*title*⟩} typesets the title of a \chapter. There is a number if \secnumdepth is greater than -1 and \@mainmatter is TRUE.

```
\def\@chapter[#1]#2{%
  \m@mpn@new@chaptrue%
  \m@mpn@new@schapfalse%
  \def\f@rbdy{#2}%
  \ifx\ch@pt@c\@empty % no optional args
    \def\f@rtoc{#2}%
    \def\f@rhdr{#2}%
  \else                  % at least one opt arg
    \let\f@rtoc\ch@pt@c
    \ifx\@empty#1\@empty
      \let\f@rhdr\ch@pt@c
    \else
      \def\f@rhdr{#1}%
    \fi
  \fi
  \m@m@Andfalse
  \ifnum \c@secnumdepth >\m@ne
    \if@mainmatter
      \m@m@Andtrue
    \fi
  \fi
  \ifm@m@And
    \refstepcounter{chapter}%
  \fi
```

Deal with the article option.

```
  \ifartopt
    \@makechapterhead{#2}%
    \@afterheading
    \chaptermark{\f@rhdr}%
  \else
```

Store the (short) title via \chaptermark, and add some whitespace to the LoF and LoT. Then fiddle when we are using two columns, calling \@makechapterhead to do the typesetting.

```
    \chaptermark{\f@rhdr}
    \insertchapterspace
    \if@twocolumn
      \@topnewpage[\@makechapterhead{#2}]%
    \else
      \@makechapterhead{#2}%
    \fi
    \@afterheading
  \fi
```

Vittorio De Martino (demartino.vittoria@grtn.it) on 31 March 2003 reported that for articled chapters, which do not do any \clearpage, \addcontents had to come after the title.

```
  \ifm@m@And
    \ifanappendix
      \addcontentsline{toc}{appendix}{%
        \protect\chapternumberline{\thechapter}\f@rtoc}%
      \memappchapinfo{\thechapter}{\f@rtoc}{\f@rhdr}{#2}%
    \else
      \addcontentsline{toc}{chapter}{%
        \protect\chapternumberline{\thechapter}\f@rtoc}%
      \memchapinfo{\thechapter}{\f@rtoc}{\f@rhdr}{#2}%
    \fi
  \else
    \addcontentsline{toc}{chapter}{\f@rtoc}%
    \ifanappendix
      \memappchapinfo{}{\f@rtoc}{\f@rhdr}{#2}%
    \else
      \memchapinfo{}{\f@rtoc}{\f@rhdr}{#2}%
    \fi
  \fi
```

Add hook for title referencing.

```
  \ifheadnameref\M@gettitle{\f@rhdr}\else\M@gettitle{\f@rtoc}\fi}

```

\@makechapterhead This *really* typesets a \chapter. Leave some whitespace and prepare to set \raggedright.

```
\def\@makechapterhead#1{%
  \chapterheadstart%  \vspace*{50\p@}%
  {%\parindent \z@ \raggedright \normalfont
   \parindent \z@ \memRTLraggedright \normalfont
```

If there is a number, typeset it, otherwise call \printchapternonum.

```
    \ifm@m@And
      \printchaptername \chapternamenum \printchapternum
      \afterchapternum % \par\nobreak \vskip 20\p@
```

```
3705    \else
3706      \printchapternonum
3707    \fi
```

Typeset the title.

```
3708    \interlinepenalty\@M
3709    \printchaptertitle{#1} % \Huge \bfseries #1
3710    \afterchaptertitle % \par\nobreak \vskip 40\p@
3711   }}
3712
```

\insertchapterspace By default, a \chapter inserts some vertical space into the LoF and LoT. The macro \insertchapterspace performs the insertion.

```
3713 \newcommand*{\insertchapterspace}{%
3714   \addtocontents{lof}{\protect\addvspace{10pt}}%
3715   \addtocontents{lot}{\protect\addvspace{10pt}}}
3716
```

\beforechapskip \midchapskip \afterchapskip Lengths separating the various parts of a chapter heading.

```
3717 \newlength{\beforechapskip}\setlength{\beforechapskip}{50pt}
3718 \newlength{\midchapskip}\setlength{\midchapskip}{20pt}
3719 \newlength{\afterchapskip}\setlength{\afterchapskip}{40pt}
3720
```

\@chs@def@ult This sets up all the definitions and defaults used in \@makechapterhead and \@makeschapterhead.

\chapterheadstart \printchaptername \chapternamenum \printchapternum \afterchapternum \printchapternum \printchapternonum \printchaptertitle \afterchaptertitle

```
3721 \newcommand{\@chs@def@ult}{%
3722   \def\chapterheadstart{\vspace*{\beforechapskip}}%
3723   \def\printchaptername{\chapnamefont \@chapapp}%
3724   \def\chapternamenum{\space}%
3725   \def\printchapternum{\chapnumfont \thechapter}%
3726   \def\afterchapternum{\par\nobreak\vskip \midchapskip}%
3727   \def\printchapternonum{}%
3728   \def\printchaptertitle##1{\chaptitlefont ##1}%
3729   \def\afterchaptertitle{\par\nobreak\vskip \afterchapskip}%
```

\chapnamefont \chapnumfont \chaptitlefont Fonts for setting chapter name, number, and title.

```
3730   \def\chapnamefont{\normalfont\huge\bfseries}%
3731   \def\chapnumfont{\normalfont\huge\bfseries}%
3732   \def\chaptitlefont{\normalfont\Huge\bfseries}%
3733   \setlength{\beforechapskip}{50pt}%
3734   \setlength{\midchapskip}{20pt}%
3735   \setlength{\afterchapskip}{40pt}}
3736
```

\@m@mschapter This deals with the new optional argument for starred chapters.

```
3737 \newcommand{\@m@mschapter}[2][\@empty]{%
3738   \@schapter{#2}%
```

```
  \ifx \@empty#1
    \def\f@rhdr{#2}%
  \else%                    opt arg
    \def\f@rhdr{#1}%
    \setcounter{secnumdepth}{-10}%
    \chaptermark{#1}%
    \setcounter{secnumdepth}{\value{maxsecnumdepth}}%
  \fi
  \ifanappendix
    \memappchapstarinfo{\f@rhdr}{#2}%
  \else
    \memchapstarinfo{\f@rhdr}{#2}%
  \fi
  \ifheadnameref\M@gettitle{\f@rhdr}\else\M@gettitle{#2}\fi}

```

`\@schapter` `\f@rbdy`

`\@schapter{`⟨*long*⟩`}` typesets the title of a `\chapter*`. It is easier than the `\@chapter` as there is no number to worry about.

```
\newcommand{\@schapter}[1]{%
  \m@mpn@new@schaptrue%
  \m@mpn@new@chapfalse%
  \def\f@rbdy{#1}%
  \ifartopt
    \@makeschapterhead{#1}%
  \else
    \if@twocolumn
      \@topnewpage[\@makeschapterhead{#1}]%
    \else
      \@makeschapterhead{#1}%
    \fi
  \fi
  \@afterheading}

```

`\@makeschapterhead`

This *really* typesets a `\chapter*`, and is similar to `\@makechapterhead`.

```
\def\@makeschapterhead#1{%
  \chapterheadstart
  {%\parindent \z@ \raggedright \normalfont
   \parindent \z@ \memRTLraggedright \normalfont
   \printchapternonum
   \interlinepenalty\@M
   \printchaptertitle{#1}%
   \afterchaptertitle}%
}

```

12.7.1 Chapter styling

`\makechapterstyle` `\chapterstyle`

`\chapterstyle{`⟨*style*⟩`}` is like `\pagestyle`, except it's for chapters.

`\makechapterstyle{`⟨*style*⟩`}{`⟨*text*⟩`}` creates or overrides the ⟨*style*⟩ chapter style defining it as ⟨*text*⟩ So, `\makechapterstyle{fred}{code}` specifies that the chapter style *fred* is defined as the macro `\chs@fred{code}`, and `\chapterstyle{fred}` calls the macro `\chs@fred`.

```
\newcommand{\makechapterstyle}[2]{\@namedef{chs@#1}{\@chs@def@ult #2}}
\newcommand{\chapterstyle}[1]{\@nameuse{chs@#1}}
```

Set the *default* chapter style.

```
\makechapterstyle{default}{}%
%%  \setlength{\beforechapskip}{50pt}
%%  \def\chapterheadstart{\vspace*{\beforechapskip}}
%%  \def\chapnamefont{\normalfont\huge\bfseries}
%%  \def\printchaptername{\chapnamefont \@chapapp}
%%  \def\chapternamenum{\space}
%%  \def\chapnumfont{\normalfont\huge\bfseries}
%%  \def\printchapternum{\chapnumfont \thechapter}
%%  \setlength{\midchapskip}{20pt}
%%  \def\afterchapternum{\par\nobreak\vskip \midchapskip}
%%  \def\printchapternonum{}
%%  \def\chaptitlefont{\normalfont\Huge\bfseries}
%%  \def\printchaptertitle##1{\chaptitlefont ##1}
%%  \setlength{\afterchapskip}{40pt}
%%  \def\afterchaptertitle{\par\nobreak\vskip \afterchapskip}}
\chapterstyle{default}
```

`\chs@section` The *section* chapter style. It prints the heading as though it were a section.

```
\makechapterstyle{section}{%
  \chapterstyle{default}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\chapnumfont}{\normalfont\Huge\bfseries}
  \renewcommand{\printchapternum}{\chapnumfont \thechapter\space}
  \renewcommand{\afterchapternum}{}}
```

`\chs@article` The *article* chapter style. It prints the heading as though it were a section in an article class document.

```
\makechapterstyle{article}{%
  \chapterstyle{default}
  \setlength{\beforechapskip}{3.5ex \@plus 1ex \@minus .2ex}
  \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}
  \setlength{\afterchapskip}{2.3ex \@plus .2ex}
  \renewcommand{\printchaptername}{}
  \renewcommand{\chapternamenum}{}
  \renewcommand{\chaptitlefont}{\normalfont\Large\bfseries}
  \renewcommand{\chapnumfont}{\chaptitlefont}
  \renewcommand{\printchapternum}{\chapnumfont \thechapter\quad}
```

```
  \renewcommand{\afterchapternum}{}}
```

\chs@reparticle The *reparticle* chapter style. It replicates the appearance of a \section in an article class document.

```
\makechapterstyle{reparticle}{%
  \chapterstyle{default}
  \setlength{\beforechapskip}{3.5ex \@plus 1ex \@minus .2ex}
  \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}
  \setlength{\afterchapskip}{2.3ex \@plus .2ex}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\chaptitlefont}{\normalfont\Large\bfseries}
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  \renewcommand*{\printchapternum}{\@hangfrom{\chapnumfont \thechapter\quad}}%
  \renewcommand*{\afterchapternum}{}}
```

\reparticle Call this to get division heads to replicate the article class. For example: \ifartopt \reparticle \fi
The \chapter commands must still be used, though.

```
\newcommand*{\reparticle}{%
  \chapterstyle{reparticle}
%%  \setsecheadstyle{\normalfont\large\bfseries\raggedright}
  \setsecheadstyle{\normalfont\large\bfseries\memRTLraggedright}%
%%  \setsubsecheadstyle{\normalfont\bfseries\raggedright}
  \setsubsecheadstyle{\normalfont\bfseries\memRTLraggedright}%
}
```

\chs@hangnum The *hangnum* style puts the chapter number in the margin.

```
\makechapterstyle{hangnum}{%
  \chapterstyle{default}
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  \settowidth{\chapindent}{\chapnumfont 999}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{%
    \noindent\llap{\makebox[\chapindent][l]{\chapnumfont \thechapter}}}
  \renewcommand*{\afterchapternum}{}}
```

\chapindent \chs@companion The *companion* style is like that in the *LaTeX Companion* series. It requires the \chapindent length. The user needs to be careful to have a wide enough spine margin on verso pages if the title may appear on a verso page.

```
\newlength{\chapindent}

\makechapterstyle{companion}{%
  \chapterstyle{default}
```

```
3853  \renewcommand*{\chapnamefont}{\normalfont\LARGE\scshape}
3854  \renewcommand*{\printchaptername}{\raggedleft\chapnamefont \@chapapp}
3855  \renewcommand*{\chapnumfont}{\normalfont\Huge}
3856  \setlength{\chapindent}{\marginparsep}
3857  \addtolength{\chapindent}{\marginparwidth}
3858  \renewcommand*{\printchaptertitle}[1]{%
3859    \begin{adjustwidth}{}{-\chapindent}
3860      \raggedleft \chaptitlefont ##1\par\nobreak
3861    \end{adjustwidth}}}
3862
```

`\chs@demo` An exotic chapter style for demonstration purposes.

```
3863 \makechapterstyle{demo}{%
3864  \chapterstyle{default}
3865  \renewcommand*{\printchaptername}{\centering}
3866  \renewcommand*{\printchapternum}{\chapnumfont \numtoName{\c@chapter}}
3867  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
3868  \renewcommand*{\printchaptertitle}[1]{%
3869    \hrule\vskip\onelineskip \raggedleft \chaptitlefont ##1}
3870  \renewcommand*{\afterchaptertitle}%
3871                {\vskip\onelineskip \hrule\vskip \afterchapskip}}
3872
```

`\chs@bianchi` Posted to CTT on 2003/12/09, *New chapter style: chapter vs chapter** by Stefano Bianchi.

```
3873 \makechapterstyle{bianchi}{%
3874  \chapterstyle{default}
3875  \renewcommand*{\chapnamefont}{\normalfont\Large\sffamily\itshape}
3876  \renewcommand*{\chapnumfont}{\normalfont\huge}
3877  \renewcommand*{\printchaptername}{%
3878    \chapnamefont\centering\@chapapp}
3879  \renewcommand*{\printchapternum}{\chapnumfont \textit{\thechapter}}
3880  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
3881  \renewcommand*{\printchaptertitle}[1]{%
3882    \hrule\vskip\onelineskip \centering \chaptitlefont\textbf{##1}\par}
3883  \renewcommand*{\afterchaptertitle}{\vskip\onelineskip \hrule\vskip
3884    \afterchapskip}
3885  \renewcommand*{\printchapternonum}{%
3886    \vphantom{\chapnumfont \textit{9}}\afterchapternum}}
3887
```

`\chs@bringhurst` My bringhurst style.

```
3888 \makechapterstyle{bringhurst}{%
3889  \chapterstyle{default}
3890  \renewcommand*{\chapterheadstart}{}
3891  \renewcommand*{\printchaptername}{}
3892  \renewcommand*{\chapternamenum}{}
3893  \renewcommand*{\printchapternum}{}
3894  \renewcommand*{\afterchapternum}{}
```

```
  \renewcommand*{\printchaptertitle}[1]{%
%%%     \raggedright\Large\scshape\MakeLowercase{##1}}
    \memRTLraggedright\Large\scshape\MakeLowercase{##1}}
  \renewcommand*{\afterchaptertitle}{%
  \vskip\onelineskip \hrule\vskip\onelineskip}}

```

`\chs@brotherton` An extremely simple chapterstyle created by William Adams for Mike Brotherton's science fiction novel *Star Dragon*, posted to CTT 2006/12/09, *An example of a novel?*.

```
\makechapterstyle{brotherton}{%
  \chapterstyle{default}
  \renewcommand*{\printchapternum}{\chapnumfont
    \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}}

```

`\chs@chappell` My Chappell style

```
\makechapterstyle{chappell}{%
  \chapterstyle{default}
  \setlength{\beforechapskip}{0pt}
  \renewcommand*{\chapnamefont}{\large\centering}
  \renewcommand*{\chapnumfont}{\large}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\printchaptername \chapnumfont 1}
    \afterchapternum
    \vskip \onelineskip \vskip -\topskip}
  \renewcommand*{\chaptitlefont}{\Large\itshape}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \centering\chaptitlefont ##1}}

```

`\chs@culver` By me in an answer to Christopher Culver on CTT, *"Biblical" formatting, how?* on 2004/03/29,where I called it the 'biblical' style.

```
\makechapterstyle{culver}{%
  \chapterstyle{default}
  \chapterstyle{article}%
  \renewcommand*{\thechapter}{\Roman{chapter}}
  \renewcommand*{\printchapternum}{% center number/title
    \centering\chapnumfont \thechapter\space\space}%
  \renewcommand*{\printchapternonum}{\centering}
  \renewcommand*{\clearforchapter}{}% no new page
  \aliaspagestyle{chapter}{headings}}% no special pagestyle

```

`\chs@dash` A very simple style of mine but I couldn't think of a good name for it.

```
\makechapterstyle{dash}{%
  \chapterstyle{default}
  \setlength{\beforechapskip}{5\onelineskip}
  \renewcommand*{\printchaptername}{}
```

```
\renewcommand*{\chapternamenum}{}
\renewcommand*{\chapnumfont}{\normalfont\large}
\settoheight{\midchapskip}{\chapnumfont 1}
\renewcommand*{\printchapternum}{\centering \chapnumfont
  \rule[0.5\midchapskip]{1em}{0.4pt} \thechapter\
  \rule[0.5\midchapskip]{1em}{0.4pt}}
\renewcommand*{\afterchapternum}{\par\nobreak\vskip 0.5\onelineskip}
\renewcommand*{\printchapternonum}{\centering
               \vphantom{\chapnumfont 1}\afterchapternum}
\renewcommand*{\chaptitlefont}{\normalfont\Large}
\renewcommand*{\printchaptertitle}[1]{\centering \chaptitlefont ##1}
\setlength{\afterchapskip}{2.5\onelineskip}}
```

`\chs@demo2` My second version of the demo chapterstyle.

```
\makechapterstyle{demo2}{%
  \chapterstyle{default}
  \renewcommand*{\printchaptername}{\centering}
  \renewcommand*{\printchapternum}{\chapnumfont
     \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}
  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \raggedleft \chaptitlefont ##1}
  \renewcommand*{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip \afterchapskip}
  \setlength{\beforechapskip}{3\baselineskip}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\chapnumfont One}
    \afterchapternum%
    \vskip\topskip}
  \setlength{\beforechapskip}{2\onelineskip}}
```

`\chs@demo3` My third version of the demo chapterstyle.

```
\makechapterstyle{demo3}{%
  \chapterstyle{default}
  \renewcommand*{\printchaptername}{\centering}
  \renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape}
  \renewcommand*{\printchapternum}{\chapnumfont
     \ifanappendix \thechapter \else \numtoName{\c@chapter}\fi}
  \renewcommand*{\chaptitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\printchaptertitle}[1]{%
    \hrule\vskip\onelineskip \raggedleft \chaptitlefont ##1}
  \renewcommand*{\afterchaptertitle}{%
    \vskip\onelineskip \hrule\vskip \afterchapskip}
  \setlength{\beforechapskip}{0pt}
  \setlength{\midchapskip}{2\onelineskip}
  \setlength{\afterchapskip}{2\onelineskip}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\chapnumfont One}
```

```
    \afterchapternum%
    \vskip\topskip}}
```

`\chs@ell` Another of my styles which I'll probably use for my next book.

```
\makechapterstyle{ell}{%
  \chapterstyle{default}
  \renewcommand*{\chapnumfont}{\normalfont\HUGE\sffamily}
  \renewcommand*{\chaptitlefont}{\normalfont\huge\sffamily}
  \settowidth{\chapindent}{\chapnumfont 111}
  \renewcommand*{\chapterheadstart}{\begingroup
    \vspace*{\beforechapskip}%
    \begin{adjustwidth}{}{-\chapindent}%
    \hrulefill
    \smash{\rule{0.4pt}{15mm}}
    \end{adjustwidth}\endgroup}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{%
    \begin{adjustwidth}{}{-\chapindent}
    \hfill
    \raisebox{10mm}[0pt][0pt]{\chapnumfont \thechapter}%
                              \hspace*{1em}
    \end{adjustwidth}\vspace*{-3.0\onelineskip}}
  \renewcommand*{\printchaptertitle}[1]{%
    \vskip\onelineskip
    \raggedleft {\chaptitlefont ##1}\par\nobreak}}
```

`\chs@ger` Posted to CTT on 2002/04/12 *Fancy Headings, Chapter Headings* by Gerardo Garcia.

```
\makechapterstyle{ger}{%
  \chapterstyle{default}
  \renewcommand*{\chapterheadstart}{\vspace*{\beforechapskip}
  \mbox{}\\\mbox{}\rule[0pt]{\textwidth}{0.4pt}\par}
  \setlength{\midchapskip}{20pt}
  \renewcommand*{\printchaptertitle}[1]{\chaptitlefont ##1
    \\\mbox{}\rule[5pt]{\textwidth}{0.4pt}}}
```

`\chs@lyhne` Posted to CTT 2006/02/09 *Glossary*, by Anders Lyhne. It requires the graphicx package. I have modified it to remove the new length and adjusted the unnumbered appearance.

```
\makechapterstyle{lyhne}{%  needs graphicx package
  \chapterstyle{default}
  \setlength{\beforechapskip}{1.5cm}
  \setlength{\afterchapskip}{1cm}
  \setlength{\midchapskip}{2cm}
  \renewcommand*{\chapnamefont}{\normalfont\normalsize\scshape\raggedleft}
```

```
  \renewcommand*{\chaptitlefont}{\normalfont\normalsize\bfseries\sffamily\raggedleft}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{\makebox[0pt][l]{\hspace{0.2em}%
    \resizebox{!}{2ex}{\chapnamefont\bfseries\sffamily\thechapter}}}
  \renewcommand*{\afterchapternum}{\par\hspace{1.5cm}\hrule\vspace{0.2cm}}
\renewcommand*{\printchapternonum}{\vphantom{\chapnamefont 1}\afterchapternum}
  \renewcommand*{\afterchaptertitle}{\vskip 0.2cm
    \hrule\vskip\afterchapskip}}

```

`\chs@madsen` Posted to CTT on 2003/12/09, *New chapter style: chapter vs chapter** by Lars Madsen. This requires the graphicx package,

```
\makechapterstyle{madsen}{% requires graphicx package
  \chapterstyle{default}
  \renewcommand*{\chapnamefont}{%
    \normalfont\Large\scshape\raggedleft}
  \renewcommand*{\chaptitlefont}{%
    \normalfont\Huge\bfseries\sffamily\raggedleft}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{%
    \makebox[0pt][l]{\hspace{0.4em}
      \resizebox{!}{4ex}{%
        \chapnamefont\bfseries\sffamily\thechapter}
    }%
  }%
  \renewcommand*{\printchapternonum}{%
    \chapnamefont \phantom{\printchaptername \chapternamenum \printchapternum}
    \afterchapternum %
  }%
  \renewcommand*{\afterchapternum}{%
    \par\hspace{1.5cm}\hrule\vskip\midchapskip}}

```

`\colorchapnum` `\colorchaptitle` `\chs@pedersen` Posted to CTT 2006/01/31, *Chapter style*, by Troels Pedersen. It requires the graphicx package and possibly the color package. I have used a `\sidebar` instead of the original `\marginpar`, and eliminated a new length. The original color specification, the same for the title and nunmber, can now be set via `\colorchapnum` and `\colorchaptitle`.

```
\newcommand*{\colorchapnum}{}
\newcommand*{\colorchaptitle}{}
\makechapterstyle{pedersen}{%
  \chapterstyle{default}
  \setlength{\beforechapskip}{-20pt}
  \setlength{\afterchapskip}{10pt}
  \renewcommand*{\chapnamefont}{\normalfont\LARGE\itshape}
  \renewcommand*{\chapnumfont}{\normalfont\HUGE\itshape\colorchapnum}
  \renewcommand*{\chaptitlefont}{\normalfont\huge\itshape\colorchaptitle}
  \renewcommand*{\afterchapternum}{}
  \renewcommand*{\printchaptername}{}
```

```
  \setlength{\midchapskip}{20mm}% was \numberheight
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\printchapternum}{%
    \sidebar{\raisebox{0pt}[0pt][0pt]{\makebox[0pt][l]{%
      \resizebox{!}{\midchapskip}{\chapnumfont\thechapter}}}}}
  \renewcommand*{\printchaptertitle}[1]{\chaptitlefont ##1}}

```

`\chs@southall` Style provided by Thomas Dye. I have modified the original to eliminate the use of new lengths.

```
%% Thomas Dye's southall chapter style
\makechapterstyle{southall}{%
  \chapterstyle{default}
  \setlength{\afterchapskip}{5\baselineskip}
  \setlength{\beforechapskip}{36pt}%    \headindent
  \setlength{\midchapskip}{\textwidth}% \rightblock
  \addtolength{\midchapskip}{-\beforechapskip}
  \renewcommand*{\chapterheadstart}{\vspace*{2\baselineskip}}
%%%  \renewcommand*{\chaptitlefont}{\huge\rmfamily\raggedright}
  \renewcommand*{\chaptitlefont}{\huge\rmfamily\memRTLraggedright}
  \renewcommand*{\chapnumfont}{\chaptitlefont}
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\afterchapternum}{}
  \renewcommand*{\printchapternum}{%
    \begin{minipage}[t][\baselineskip][b]{\beforechapskip}
      {\vspace{0pt}\chapnumfont%%%\figureversion{lining}
                   \thechapter}
    \end{minipage}}
  \renewcommand*{\printchaptertitle}[1]{%
    \hfill\begin{minipage}[t]{\midchapskip}
      {\vspace{0pt}\chaptitlefont ##1\par}\end{minipage}}
  \renewcommand*{\afterchaptertitle}{%
    \par\vspace{\baselineskip}%
    \hrulefill \par\nobreak\noindent \vskip \afterchapskip}}

```

`\chs@thatcher` Original posted to CTT on 2006/01/18 by Scott Thatcher, *memoir: chapter headings capitalize math symbols.* I have modified it to cater for multiline titles, appendices, and unnumbered chapters.

```
\makechapterstyle{thatcher}{%
  \chapterstyle{default}
  \renewcommand*{\chapterheadstart}{}
  \renewcommand*{\printchaptername}{%
    \centerline{\chapnumfont{\@chapapp\ \thechapter}}}
  \renewcommand*{\chapternamenum}{}
  \renewcommand*{\chapnumfont}{\normalfont\scshape\MakeLowercase}
  \renewcommand*{\printchapternum}{}
  \renewcommand*{\afterchapternum}{%
```

```
    \par\centerline{\parbox{0.5in}{\hrulefill}}\par}
  \renewcommand*{\printchapternonum}{%
    \vphantom{\chapnumfont \@chapapp 1}\par
    \parbox{0.5in}{}\par}
  \renewcommand*{\chaptitlefont}{\normalfont\large}
  \renewcommand*{\printchaptertitle}[1]{%
    \centering \chaptitlefont\MakeUppercase{##1}}}

```

\chs@veelo This, from Baastian Veelo, has been noted in the documentation for quite a time. I have modified the original to eliminate the need for extra lengths. It well suits trimmed documents. We are scaling the chapter number, which most DVI viewers will not display accurately. It requires the graphicx package.

```
\makechapterstyle{veelo}{%
%  \chapterstyle{default}
   \setlength{\afterchapskip}{40pt}
  \renewcommand*{\chapterheadstart}{\vspace*{40pt}}
  \renewcommand*{\afterchapternum}{\par\nobreak\vskip 25pt}
   \renewcommand*{\chapnamefont}{\normalfont\LARGE\flushright}
   \renewcommand*{\chapnumfont}{\normalfont\HUGE}
   \renewcommand*{\chaptitlefont}{\normalfont\HUGE\bfseries\flushright}
   \renewcommand*{\printchaptername}{%
     \chapnamefont\MakeUppercase{\@chapapp}}
   \renewcommand*{\chapternamenum}{}
%   \setlength{\numberheight}{18mm}
%   \setlength{\barlength}{\paperwidth}
%   \addtolength{\barlength}{-\textwidth}
%   \addtolength{\barlength}{-\spinemargin}
  \setlength{\beforechapskip}{18mm}%  \numberheight
  \setlength{\midchapskip}{\paperwidth}% \barlength
  \addtolength{\midchapskip}{-\textwidth}
  \addtolength{\midchapskip}{-\spinemargin}
   \renewcommand*{\printchapternum}{%
     \makebox[0pt][l]{%
       \hspace{.8em}%
       \resizebox{!}{\beforechapskip}{\chapnumfont \thechapter}%
       \hspace{.8em}%
       \rule{\midchapskip}{\beforechapskip}%
     }%
   }%
   \makeoddfoot{plain}{}{}{\thepage}}

```

\chs@verville I posted the original to CTT on 2005/01/18, *Headers and special formatting of sections*, in answer to a question by Guy Verville. This version caters for unnumbered chapters.

```
\makechapterstyle{verville}{%
 % \chapterstyle{default}
  \setlength{\beforechapskip}{0pt}
```

```
  \renewcommand*{\printchaptername}{}
  \renewcommand*{\printchapternum}{%
    \hrule \vskip 0.5\onelineskip
    \Huge \centering \thechapter.\ }
  \renewcommand*{\printchapternonum}{%
    \hrule \vskip 0.5\onelineskip
    \Huge \centering}
  \renewcommand*{\afterchapternum}{}
  \setlength{\midchapskip}{0pt}
  \renewcommand*{\printchaptertitle}[1]{%
    ##1 \par
    \vskip 0.5\onelineskip
   \hrule}}

```

`\chs@crosshead` A centered chapterstyle (from Bringhurst)

```
\makechapterstyle{crosshead}{%
  \setlength{\beforechapskip}{2\onelineskip}%
  \renewcommand*{\chapterheadstart}{\vspace{\beforechapskip}}%
  \setlength{\afterchapskip}{2\onelineskip \@plus .2\onelineskip
                             \@minus 0.2\onelineskip}%
  \renewcommand*{\printchaptername}{}%
  \renewcommand*{\chapternamenum}{}%
  \renewcommand*{\chapnumfont}{\normalfont\LARGE\bfseries}%
  \renewcommand*{\chaptitlefont}{\chapnumfont}%
  \renewcommand*{\printchapternum}{%
    \centering\chapnumfont \thechapter\quad}%
  \renewcommand{\afterchapternum}{}%
  \renewcommand*{\printchapternonum}{\centering}}

```

`\chs@dowding` A centered chapterstyle (from Dowding's *Finer Points*)

```
\makechapterstyle{dowding}{%
  \setlength{\beforechapskip}{2\onelineskip}%
  \setlength{\afterchapskip}{1.5\onelineskip \@plus .1\onelineskip
                             \@minus 0.167\onelineskip}%
  \renewcommand*{\chapnamefont}{\normalfont}%
  \renewcommand*{\chapnumfont}{\chapnamefont}%
  \renewcommand*{\printchapternum}{\centering\chapnumfont \ifanappendix \thechapter
                \else \numtoName{\c@chapter}\fi}%
  \renewcommand*{\chaptitlefont}{\normalfont\itshape\huge\centering}%
  \renewcommand*{\printchapternonum}{%
    \vphantom{\printchaptername}\vskip\midchapskip}}

```

`\chs@komalike` A chapterstyle approximating the KOMA script style (`scrbook.cls`)

```
\makechapterstyle{komalike}{%
  \setlength{\beforechapskip}{2\onelineskip}%
  \setlength{\afterchapskip}{1.5\onelineskip \@plus .1\onelineskip
```

```
                              \@minus 0.167\onelineskip}%
  \renewcommand*{\printchaptername}{}%
  \renewcommand*{\chapternamenum}{}%
  \renewcommand*{\chapnumfont}{\normalfont\LARGE\sffamily\bfseries}%
  \renewcommand*{\printchapternum}{\chapnumfont \thechapter\space}%
  \renewcommand*{\afterchapternum}{}%
  \renewcommand*{\chaptitlefont}{\normalfont\LARGE\sffamily\bfseries}}

```

`\chs@ntglike` A chapterstyle approximating the NTG style (`boek.cls`)

```
\makechapterstyle{ntglike}{%
  \setlength{\beforechapskip}{50pt \@plus 20pt}%
  \renewcommand*{\chapnamefont}{\normalfont\Large\bfseries}%
  \renewcommand*{\chapnumfont}{\normalfont\Large\bfseries}%
  \renewcommand*{\chaptitlefont}{\normalfont\Large\bfseries}}

```

`\chs@tandh` A chapterstyle based on Thames & Hudson *Typography*.

```
\makechapterstyle{tandh}{%
  \setlength{\beforechapskip}{1\onelineskip}%
  \setlength{\afterchapskip}{2\onelineskip \@plus .1\onelineskip
                             \@minus 0.167\onelineskip}%
  \renewcommand*{\printchaptername}{}%
  \renewcommand*{\chapternamenum}{}%
  \renewcommand*{\chapnumfont}{\normalfont\huge\bfseries}%
  \renewcommand*{\printchapternum}{\chapnumfont \thechapter\quad}%
  \renewcommand*{\afterchapternum}{}%
%%%  \renewcommand*{\chaptitlefont}{\chapnumfont\raggedright}}
  \renewcommand*{\chaptitlefont}{\chapnumfont\memRTLraggedright}}

```

`\chs@wilsondob` A chapterstyle based on Adrian Wilson's *Design of books*.

```
\makechapterstyle{wilsondob}{%
  \setlength{\beforechapskip}{2\onelineskip}%
  \setlength{\afterchapskip}{4\onelineskip \@plus .1\onelineskip
                             \@minus 0.167\onelineskip}%
  \renewcommand*{\printchaptername}{}%
  \renewcommand*{\chapternamenum}{}%
  \renewcommand*{\chapnumfont}{\normalfont\Huge\itshape}%
  \renewcommand*{\printchapternum}{\raggedleft\chapnumfont \thechapter\quad}%
  \renewcommand*{\afterchapternum}{}%
  \renewcommand*{\chaptitlefont}{\chapnumfont}%
  \renewcommand*{\printchapternonum}{\raggedleft}}

```

12.8 Lower level headings

These commands all make use of `\@startsection`. However, for the purposes of the class the kernel version needs modification to support:

- make short pages (where a section heading is moved to the top of the next page) ragged bottom;
- provide headings with two optional arguments.

We will tweak `\@startsection` so that a short page (where a section heading is moved from the bottom of the page to the top of the next) can be set `\raggedbottom`.

`\ifraggedbottomsection` `\raggedbottomsectiontrue` for ragged short pages.

```
\ifraggedbottomsectiontrue
\raggedbottomsectionfalse
    \raggedbottomsection
    \normalbottomsection
4222 \newif\ifraggedbottomsection
4223   \raggedbottomsectionfalse
4224 \newcommand*{\raggedbottomsection}{\raggedbottomsectiontrue}
4225 \newcommand*{\normalbottomsection}{\raggedbottomsectionfalse}
4226
```

`\bottomsectionskip` Decreasing this length increases short page bottom flushness.

```
4227 \newlength{\bottomsectionskip}
4228   \setlength{\bottomsectionskip}{10mm}
4229
```

We will add a second optional argument for `\section`, etc. See the CTT thread *'Long headers'* 15 Jan 2003.
The kernel `\@dblarg` is used in situations where the default value for an optional argument is the same as the required argument. Schematically,
`\@dblarg -> Main, Main`
The macros `\@trplargomm` and `\@trplargoom` are extensions to this idea for two optional arguments.

`\@trplargomm` `\@trplargomm -> Opt, Main, Main`

```
\@xtrplargomm
  \@xxtrplarg
4230 \newcommand{\@trplargomm}[1]{%
4231   \@ifnextchar[{\@xtrplargomm{#1}}%
4232                {\@xxtrplarg{#1}}}
4233 \long\def\@xtrplargomm#1[#2]{\@dblarg{#1[#2]}}
4234 \newcommand{\@xxtrplarg}[2]{#1[{#2}][{#2}]{#2}}
```

`\@trplargoom` `\@trplargoom -> Opt, Opt, Main`

```
\@xtrplargoom
4235 \newcommand{\@trplargoom}[1]{%
4236   \@ifnextchar[{\@xtrplargoom{#1}}%
4237                {\@xxtrplarg{#1}}}
4238 \long\def\@xtrplargoom#1[#2]{%
4239   \@ifnextchar[{#1[{#2}]}%
4240                {#1[{#2}][{#2}]}}
4241
```

`\memsecinfo` `\memsecinfo{secname}{\thenum}{toc}{head}{title}`
`\memsecstarinfo` `\memsecstarinfo{secname}{title}`

```
4242 \newcommand{\memsecinfo}[5]{}
4243 \newcommand{\memsecstarinfo}[2]{}
4244
```

\@startsection
\m@msecn@me

Change kernel \@startsection to:

- Make short pages raggedbottom. This is based on the thread *Can \flushbottom and \section be made to live together?* on CTT in September 2002.
- add an extra optional argument.

The original is in ltsect.dtx.
\@startsection{⟨*name*⟩}{⟨*level*⟩}{⟨*indent*⟩}
{⟨*beforeskip*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}

```
4245 \renewcommand{\@startsection}[6]{%
```

Do raggedbottom stuff.

```
4246   \ifraggedbottomsection\if@nobreak\else
4247     \vskip\z@\@plus\bottomsectionskip
4248     \penalty\z@
4249     \vskip\z@\@plus -\bottomsectionskip
4250   \fi\fi
```

Save the section name.

```
4251   \def\m@msecn@me{#1}%
```

The original code.

```
4252   \if@noskipsec \leavevmode \fi
4253   \par
4254   \@tempskipa #4\relax
4255   \@afterindenttrue
4256   \ifdim \@tempskipa <\z@
4257     \@tempskipa -\@tempskipa \@afterindentfalse
4258   \fi
4259   \if@nobreak
4260     \everypar{}%
4261   \else
4262     \addpenalty\@secpenalty\addvspace\@tempskipa
4263   \fi
```

For the extra optional argument, change the original \@dblarg{\@sect{... below to \@trplargoom{\M@sect{....

```
4264   \@ifstar
4265     {\@ssect{#3}{#4}{#5}{#6}}%
4266     {\@trplargoom{\M@sect{#1}{#2}{#3}{#4}{#5}{#6}}}}
4267
```

\M@sect

At least the hyperref and nameref packages, and possibly other packages, modify the kernel \@sect assuming 8 args. That's why I've called my version, which has 9 args, something else (i.e., \M@sect). This version also has the support for title referencing.
\M@sect{⟨*name*⟩}{⟨*level*⟩}{⟨*indent*⟩}{⟨*beforeskip*⟩}{⟨*afterskip*⟩}{⟨*style*⟩}
[⟨*toc*⟩][⟨*head*⟩]{⟨*title*⟩}

```
\def\M@sect#1#2#3#4#5#6[#7][#8]#9{%
  \ifheadnameref\M@gettitle{#8}\else\M@gettitle{#7}\fi
  \ifnum #2>\c@secnumdepth
    \let\@svsec\@empty
    \memsecinfo{#1}{}{#7}{#8}{#9}%
  \else
    \refstepcounter{#1}%
    \protected@edef\@svsec{\@seccntformat{#1}\relax}%
    \memsecinfo{#1}{\@nameuse{the#1}}{#7}{#8}{#9}%
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
      #6{%
      \@hangfrom{\hskip #3\relax\@svsec}%
        \interlinepenalty \@M #9\@@par}%
    \endgroup
    \csname #1mark\endcsname{#8}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
  \else
    \def\@svsechd{%
      #6{\hskip #3\relax
     \@svsec #9}%
     \csname #1mark\endcsname{#8}%
     \addcontentsline{toc}{#1}{%
       \ifnum #2>\c@secnumdepth \else
        \protect\numberline{\csname the#1\endcsname}%
       \fi
       #7}}%
  \fi
  \@xsect{#5}}

```

`\@ssect` Add hook for name reference to `\section*` etc.
`\@ssect{`⟨*indent*⟩`}{`⟨*beforeskip*⟩`}{`⟨*afterskip*⟩`}{`⟨*style*⟩`}{`⟨*title*⟩`}`

```
\let\@mem@old@ssect\@ssect
\def\@ssect#1#2#3#4#5{%
  \M@gettitle{#5}%
  \memsecstarinfo{\m@msecn@me}{#5}%
  \@mem@old@ssect{#1}{#2}{#3}{#4}{#5}}

```

`\section` A normal heading with white space above and below and no indentation of the first paragraph. By default the heading is set in a `\Large\bfseries` font.

```
\newcommand{\section}{%
```

```
  \sechook%
  \@startsection{section}{1}%  level 1
      {\secindent}%            heading indent
      {\beforesecskip}%        skip before the heading
      {\aftersecskip}%         skip after the heading
      {\normalfont\secheadstyle}} % font
```

\sechook
\setsechook

\sechook is called at the start of a \section and \setsechook{⟨code⟩} redefines it.

```
\newcommand{\sechook}{}
\newcommand{\setsechook}[1]{\renewcommand{\sechook}{#1}}
```

\secindent
\beforesecskip
\aftersecskip
\secheadstyle
\setsecindent
\setbeforesecskip
\setaftersecskip
\setsecheadstyle

\secindent is the indentation of the section heading from the left margin, \beforesecskip and \aftersecskip specify the white space before and after the heading, and \secheadstyle specifies the heading style. These are set to the default values for the book class, except that the heading will be \raggedright, thus preventing hyphenation. The \set... commands are for the user to change the values.

```
\newlength{\secindent}
\newcommand{\setsecindent}[1]{\setlength{\secindent}{#1}}
  \setsecindent{\z@}
\newskip\beforesecskip
\newcommand{\setbeforesecskip}[1]{\setlength{\beforesecskip}{#1}}
  \setbeforesecskip{-3.5ex \@plus -1ex \@minus -.2ex}
\newskip\aftersecskip
\newcommand{\setaftersecskip}[1]{\setlength{\aftersecskip}{#1}}
  \setaftersecskip{2.3ex \@plus .2ex}
\newcommand{\secheadstyle}{}
\newcommand{\setsecheadstyle}[1]{\renewcommand{\secheadstyle}{#1}}
%%%  \setsecheadstyle{\Large\bfseries\raggedright}
  \setsecheadstyle{\Large\bfseries\memRTLraggedright}

```

\subsection

A normal heading with white space above and below and no indentation of the first paragraph. By default the heading is set in a \large\bfseries font.

```
\newcommand{\subsection}{%
  \subsechook%
  \@startsection{subsection}{2}%  level 2
      {\subsecindent}%            heading indent
      {\beforesubsecskip}%        skip before the heading
      {\aftersubsecskip}%         skip after the heading
      {\normalfont\subsecheadstyle}} % font
```

\subsechook
\setsubsechook

\subsechook is called at the start of a \subsection and \setsubsechook{⟨code⟩} redefines it.

```
\newcommand{\subsechook}{}
\newcommand{\setsubsechook}[1]{\renewcommand{\subsechook}{#1}}
```

\subsecindent
\beforesubsecskip
\aftersubsecskip
\subsecheadstyle
\setsubsecindent
\setbeforesubsecskip
\setaftersubsecskip
\setsubsecheadstyle

The macros for controlling \subsection headings.

```
\newlength{\subsecindent}
\newcommand{\setsubsecindent}[1]{\setlength{\subsecindent}{#1}}
  \setsubsecindent{\z@}
\newskip\beforesubsecskip
\newcommand{\setbeforesubsecskip}[1]{\setlength{\beforesubsecskip}{#1}}
  \setbeforesubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}
\newskip\aftersubsecskip
\newcommand{\setaftersubsecskip}[1]{\setlength{\aftersubsecskip}{#1}}
  \setaftersubsecskip{1.5ex \@plus .2ex}
\newcommand{\subsecheadstyle}{}
\newcommand{\setsubsecheadstyle}[1]{\renewcommand{\subsecheadstyle}{#1}}
%%  \setsubsecheadstyle{\large\bfseries\raggedright}
  \setsubsecheadstyle{\large\bfseries\memRTLraggedright}

```

\subsubsection

A normal heading with white space above and below and no indentation of the first paragraph. By default the heading is set in a \normalsize\bfseries font.

```
\newcommand{\subsubsection}{%
  \subsubsechook%
  \@startsection{subsubsection}{3}%  level 3
      {\subsubsecindent}%            heading indent
      {\beforesubsubsecskip}%        skip before the heading
      {\aftersubsubsecskip}%         skip after the heading
      {\normalfont\subsubsecheadstyle}} % font
```

\subsubsechook
\setsubsubsechook

\subsubsechook is called at the start of a \subsubsection and \setsubsubsechook{⟨*code*⟩} redefines it.

```
\newcommand{\subsubsechook}{}
\newcommand{\setsubsubsechook}[1]{\renewcommand{\subsubsechook}{#1}}
```

\subsubsecindent
\beforesubsubsecskip
\aftersubsubsecskip
\subsubsecheadstyle
\setsubsubsecindent
\setbeforesubsubsecskip
\setaftersubsubsecskip
\setsubsubsecheadstyle

The macros for controlling \subsubsection headings.

```
\newlength{\subsubsecindent}
\newcommand{\setsubsubsecindent}[1]{%
            \setlength{\subsubsecindent}{#1}}
  \setsubsubsecindent{\z@}
\newskip\beforesubsubsecskip
\newcommand{\setbeforesubsubsecskip}[1]{%
            \setlength{\beforesubsubsecskip}{#1}}
  \setbeforesubsubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}
\newskip\aftersubsubsecskip
\newcommand{\setaftersubsubsecskip}[1]{%
            \setlength{\aftersubsubsecskip}{#1}}
  \setaftersubsubsecskip{1.5ex \@plus .2ex}
\newcommand{\subsubsecheadstyle}{}
\newcommand{\setsubsubsecheadstyle}[1]{%
            \renewcommand{\subsubsecheadstyle}{#1}}
%%  \setsubsubsecheadstyle{\normalsize\bfseries\raggedright}
  \setsubsubsecheadstyle{\normalsize\bfseries\memRTLraggedright}

```

\paragraph A runin heading with white space above and to the right of the heading. By default the heading is set in a \normalsize\bfseries font.

```
\newcommand{\paragraph}{%
  \parahook%
  \@startsection{paragraph}{4}%  level 4
      {\paraindent}%            heading indent
      {\beforeparaskip}%        skip before the heading
      {\afterparaskip}%         skip after the heading
      {\normalfont\paraheadstyle}} % font
```

\parahook \setparahook

\parahook is called at the start of a \paragraph and \setparahook{⟨*code*⟩} redefines it.

```
\newcommand{\parahook}{}
\newcommand{\setparahook}[1]{\renewcommand{\parahook}{#1}}
```

\paraindent \beforeparaskip \afterparaskip \paraheadstyle \setparaindent \setbeforeparaskip \setafterparaskip \setparaheadstyle

The macros for controlling \paragraph headings.

```
\newlength{\paraindent}
\newcommand{\setparaindent}[1]{\setlength{\paraindent}{#1}}
  \setparaindent{\z@}
\newskip\beforeparaskip
\newcommand{\setbeforeparaskip}[1]{\setlength{\beforeparaskip}{#1}}
  \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}
\newskip\afterparaskip
\newcommand{\setafterparaskip}[1]{\setlength{\afterparaskip}{#1}}
  \setafterparaskip{-1em}
\newcommand{\paraheadstyle}{}
\newcommand{\setparaheadstyle}[1]{\renewcommand{\paraheadstyle}{#1}}
  \setparaheadstyle{\normalsize\bfseries}

```

\subparagraph A runin heading with white space above and to the right of the heading. By default the heading is set in a \normalsize\bfseries font.

```
\newcommand{\subparagraph}{%
  \subparahook%
  \@startsection{subparagraph}{5}%  level 5
      {\subparaindent}%            heading indent
      {\beforesubparaskip}%        skip before the heading
      {\aftersubparaskip}%         skip after the heading
      {\normalfont\subparaheadstyle}} % font
```

\subparahook \setsubparahook

\subparahook is called at the start of a \subparagraph and \setsubparahook{⟨*code*⟩} redefines it.

```
\newcommand{\subparahook}{}
\newcommand{\setsubparahook}[1]{\renewcommand{\subparahook}{#1}}
```

\subparaindent \beforesubparaskip \aftersubparaskip \subparaheadstyle \setsubparaindent \setbeforesubparaskip \setaftersubparaskip \setsubparaheadstyle

The macros for controlling \subparagraph headings.

```
\newlength{\subparaindent}
\newcommand{\setsubparaindent}[1]{%
            \setlength{\subparaindent}{#1}}
```

```
  \setsubparaindent{\parindent}
\newskip\beforesubparaskip
\newcommand{\setbeforesubparaskip}[1]{%
            \setlength{\beforesubparaskip}{#1}}
  \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}
\newskip\aftersubparaskip
\newcommand{\setaftersubparaskip}[1]{%
            \setlength{\aftersubparaskip}{#1}}
  \setaftersubparaskip{-1em}
\newcommand{\subparaheadstyle}{}
\newcommand{\setsubparaheadstyle}[1]{%
            \renewcommand{\subparaheadstyle}{#1}}
  \setsubparaheadstyle{\normalsize\bfseries}

```

`\sethangfrom` The macro `\sethangfrom{`⟨*code*⟩`}` is a user-level command for changing the definition of `\@hangfrom`. Use `#1` in ⟨*code*⟩ for the argument to `\@hangfrom` (or `##1` if used inside another macro).

```
\newcommand{\sethangfrom}[1]{\renewcommand{\@hangfrom}[1]{#1}}
```

`\setsecnumformat` The macro `\setsecnumformat{`⟨*code*⟩`}` is a user-level command for changing the definition of `\@seccntformat`. Use `#1` in ⟨*code*⟩ for the argument to `\@seccntformat` (or `##1` if used inside another macro).

```
\newcommand{\setsecnumformat}[1]{\renewcommand{\@seccntformat}[1]{#1}}
```

`\hangsecnum` `\defaultsecnum` These are declarations for putting sectional numbers in the margin, or the default sectional number formatting.

```
\newcommand{\hangsecnum}{%
  \def\@seccntformat##1{\llap{\csname the##1\endcsname\quad}}}
\newcommand{\defaultsecnum}{%
  \def\@seccntformat##1{\csname the##1\endcsname\quad}}

```

12.8.1 Anonymous headings

`\plainbreak` `\plainbreak{`⟨*num*⟩`}` generates ⟨*num*⟩ blanks lines and suppresses the indentation of a following paragraph. The starred version, `\plainbreak*`, does not suppress paragraph indentation.

```
\newcommand{\plainbreak}{\@ifstar{\@spbreak}{\@pbreak}}
```

`\@pbreak` `\@spbreak` These are the internal forms for the two versions of `\plainbreak`. The code for `\@pbreak` is almost a straight copy of code posted to CTT by Donald Arseneau on 2001/03/26.

```
\newcommand*{\@pbreak}[1]{\par
  \penalty -100
  \vskip #1\onelineskip \@plus 2\onelineskip
  \penalty -20
  \vskip \z@ \@plus -2\onelineskip
```

```
  \@afterindentfalse
  \@afterheading}
\newcommand*{\@spbreak}[1]{\par
  \penalty -100
  \vskip #1\onelineskip \@plus 2\onelineskip
  \penalty -20
  \vskip \z@ \@plus -2\onelineskip
  \@afterindenttrue
  \@afterheading}

```

`\fancybreak` `\fancybreak{`⟨*text*⟩`}` typesets ⟨*text*⟩ centered. For example, `\fancybreak{{*}\\{* * *}\\{*}}`. It suppresses indentation of the following paragraph. The starred version leaves indentation as is.

```
\newcommand{\fancybreak}{\@ifstar{\@sfbreak}{\@fbreak}}
```

`\@fbreak` `\@sfbreak` These are the internal forms for the two versions of `\fancybreak`.

```
\newcommand{\@fbreak}[1]{\par
  \penalty -100
  \noindent\parbox{\linewidth}{\centering #1}%%\null
  \par
%%  \penalty -20
%%  \vskip -\onelineskip
  \@afterindentfalse
  \@afterheading}
\newcommand{\@sfbreak}[1]{\par
  \penalty -100
  \noindent\parbox{\linewidth}{\centering #1}%%\null
  \par
%%  \penalty -20
%%  \vskip -\onelineskip
  \@afterindenttrue
  \@afterheading}

```

`\plainfancybreak` The `\plainfancybreak{`⟨*space*⟩`}{`⟨*num*⟩`}{`⟨*text*⟩`}` acts like `\plainbreak{`⟨*num*⟩`}` in the middle of a page and like `\fancybreak{`⟨*text*⟩`}` at the bottom of a page. The ⟨*space*⟩ argument is the vertical space required for the ⟨*num*⟩ blank lines and some additional lines of text. From experiments, it seems that ⟨*space*⟩ should be at least (3+⟨*num*⟩) lines (`\onelineskip`). There is also a starred version, `\plainfancybreak*`, to match the other starred break commands.

```
\newcommand{\plainfancybreak}{\@ifstar{\@spfbreak}{\@pfbreak}}
```

`\@pfbreak` `\@spfbreak` These are the internal macros for the unstarred and starred versions of `\plainfancybreak`. They essentially do the same thing, except they call, respectively, the unstarred and starred internal versions of `\plainbreak` and `\fancybreak`. The code for checking the amount of space left on the page is from an early, and abandoned, version of the code for `\needspace`.

```
\newcommand{\@pfbreak}[3]{\par
  \@tempdimc\pagegoal \advance\@tempdimc-\pagetotal
  \ifdim #1>\@tempdimc \@fbreak{#3}\else \@pbreak{#2}\fi}
\newcommand{\@spfbreak}[3]{\par
  \@tempdimc\pagegoal \advance\@tempdimc-\pagetotal
  \ifdim #1>\@tempdimc \@sfbreak{#3}\else \@spbreak{#2}\fi}
```

From the thread *Customizing section hooks in memoir.cls* on CTT in October 2002 it appeared that `\plainfancybreak` may be fragile. Donald Arseneau said that `\plainfancybreak` guessed at the amount of space available and gave code based on a modified output routine to make it more robust. The following code is based on that thread.

`\pen@ltyabovefpbreak` `\pen@ltybelowfpbreak` Penalities for communication with the output routine.

```
\newcommand*{\pen@ltyabovepfbreak}{2}
\newcommand*{\pen@ltybelowpfbreak}{-4}
```

`\pfbreakskip` The vertical space taken by the plain and fancy breaks.

```
\newlength{\pfbreakskip}
  \setlength{\pfbreakskip}{2\baselineskip}
```

`\pfbreakdisplay` This is the fancybreak display, which must fit into `\pfbreakskip` vertical space.

```
\newcommand{\pfbreakdisplay}{*\quad*\quad*}
```

`\pfbre@kdispl@y` Typesets `\pfbreakdisplay` vertically and horizontally centered.

```
\def\pfbre@kdispl@y{\vbox to 1\pfbreakskip{\vss
  \hb@xt@ \columnwidth{\hss \pfbreakdisplay \hss}%
  \vss}}
```

`\nopfbreakOutput` Save the current output routine.

```
\edef\nopfbreakOutput{\the\output}
```

`\pfbreakOutput` Special output to handle the `\pfbreak`.

```
\def\pfbreakOutput{%
  \ifnum\outputpenalty=\pen@ltyabovepfbreak
    \nopfbreakOutput
    \pfbre@kdispl@y
    \nobreak
    \vskip-\pfbreakskip
  \else\ifnum\outputpenalty=\pen@ltybelowpfbreak
    \unvbox 255\relax
    \nobreak
    \vskip-\pfbreakskip
    \pfbre@kdispl@y
    \break
```

```
  \else
    \nopfbreakOutput
  \fi
  \fi}
```

`\output` Use the new `\pfbreak` output routine.

```
\output={\pfbreakOutput}

```

`\pfbreak` Typesets a plain break in the middle of the page, otherwise a fancybreak at either the bottom or top of the page.

```
\newcommand{\pfbreak}{\@ifstar{\@spfbreakgap}{\@pfbreakgap}}
```

`\@pfbreakgap` Unstarred version of `\pfbreak`.

```
\newcommand{\@pfbreakgap}{%
  \par {%
  \skip@\lastskip
  \nobreak
  \vskip -\ifdim\prevdepth>\maxdepth \maxdepth
           \else\ifdim\prevdepth>-1000pt\prevdepth
             \else\ifinner 0pt
               \else \pagedepth
           \fi \fi \fi
  \vskip -\skip@
  \ifdim\skip@<\pfbreakskip
    \advance\skip@ -1\skip@ \advance\skip@ 1\pfbreakskip
  \fi
  \penalty\pen@ltyabovepfbreak
  \vskip\skip@
  \penalty\pen@ltybelowpfbreak
  }
  \@afterindentfalse
  \@afterheading
}
```

`\@spfbreakgap` Starred version of `\pfbreak`.

```
\newcommand{\@spfbreakgap}{%
  \par {%
  \skip@\lastskip
  \nobreak
  \vskip -\ifdim\prevdepth>\maxdepth \maxdepth
           \else\ifdim\prevdepth>-1000pt\prevdepth
             \else\ifinner 0pt
               \else \pagedepth
           \fi \fi \fi
  \vskip -\skip@
  \ifdim\skip@<\pfbreakskip
    \advance\skip@ -1\skip@ \advance\skip@ 1\pfbreakskip
  \fi
```

```
  \penalty\pen@ltyabovepfbreak
  \vskip\skip@
  \penalty\pen@ltybelowpfbreak
  }
  \@afterindenttrue
  \@afterheading
}
```

While on the subject of breaks...

`\noprelistbreak` Putting this immediately before a list (e.g., `itemize`) should prevent a page break at that point.

```
\newcommand*{\noprelistbreak}{\@nobreaktrue\nopagebreak}
```

12.9 Division head styles

The styles of the division heads should go together. As an aid the class supplies some ready made collections.

`\makeheadstyles` `\makeheadstyles{`⟨*name*⟩`}{`⟨*code*⟩`}` creates the ⟨*name*⟩ collection of division
`\headstyles` head styles, defined by ⟨*code*⟩. `\headstyles{`⟨*name*⟩`}` makes the ⟨*name*⟩ division
`\@hds@def@ult` head styles the current styles.
`\hds@default`

```
\newcommand{\makeheadstyles}[2]{%
  \@namedef{hds@#1}{\@hds@def@ult #2}}
\newcommand*{\headstyles}[1]{\@nameuse{hds@#1}}
```

`\@hds@def@ult` is the default set of division head styles.

```
\newcommand*{\@hds@def@ult}{%
% Default \cs{book} style
%    \begin{macrocode}
% book
  \renewcommand*{\beforebookskip}{\null\vfil}%
  \renewcommand*{\midbookskip}{\par\vskip 20pt}%
  \renewcommand*{\afterbookskip}{\vfil\newpage}%
  \renewcommand*{\booknamefont}{\normalfont\huge\bfseries}%
  \renewcommand*{\booknumfont}{\normalfont\huge\bfseries}%
  \renewcommand*{\booktitlefont}{\normalfont\Huge\bfseries}%
  \renewcommand*{\printbookname}{\booknamefont \bookname}%
  \renewcommand*{\booknamenum}{\space}%
  \renewcommand*{\printbooknum}{\booknumfont \thebook}%
  \renewcommand*{\printbooktitle}[1]{\booktitlefont{##1}}%
```

Default `\part` style

```
% part
  \renewcommand*{\beforepartskip}{\null\vfil}%
  \renewcommand*{\midpartskip}{\par\vskip 20pt}%
  \renewcommand*{\afterpartskip}{\vfil\newpage}%
  \renewcommand*{\partnamefont}{\normalfont\huge\bfseries}%
```

```
  \renewcommand*{\partnumfont}{\normalfont\huge\bfseries}%
  \renewcommand*{\parttitlefont}{\normalfont\Huge\bfseries}%
  \renewcommand*{\printpartname}{\partnamefont \partname}%
  \renewcommand*{\partnamenum}{\space}%
  \renewcommand*{\printpartnum}{\partnumfont \thepart}%
  \renewcommand*{\printparttitle}[1]{\parttitlefont{##1}}%
```

Default `\chapter` style.

```
  \@chs@def@ult%               default chapterstyle
```

Default `\section` style

```
% section
  \setsechook{}
  \setsecindent{\z@}%
  \setbeforesecskip{-3.5ex \@plus -1ex \@minus -.2ex}%
  \setaftersecskip{2.3ex \@plus .2ex}%
%%%  \setsecheadstyle{\Large\bfseries\raggedright}%
  \setsecheadstyle{\Large\bfseries\memRTLraggedright}%
```

Default `\subsection` style

```
% subsection
  \setsubsechook{}%
  \setsubsecindent{\z@}%
  \setbeforesubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
  \setaftersubsecskip{1.5ex \@plus .2ex}%
%%%  \setsubsecheadstyle{\large\bfseries\raggedright}%
  \setsubsecheadstyle{\large\bfseries\memRTLraggedright}%
```

Default `\subsubsection` style

```
% subsubsection
  \setsubsubsechook{}%
  \setsubsubsecindent{\z@}%
  \setbeforesubsubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
  \setaftersubsubsecskip{1.5ex \@plus .2ex}%
%%%  \setsubsubsecheadstyle{\normalsize\bfseries\raggedright}%
  \setsubsubsecheadstyle{\normalsize\bfseries\memRTLraggedright}%
```

Default `\paragraph` style

```
% paragraph
  \setparahook{}%
  \setparaindent{\z@}%
  \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalsize\bfseries}%
```

Default `\paragraph` style

```
% subparagraph
  \setsubparahook{}%
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalsize\bfseries}}
```

Set up and use the default head styles

```
\makeheadstyles{default}{}
\headstyles{default}
```

`\addperiod` Puts a period at the end of its argument.

```
\newcommand*{\addperiod}[1]{#1.}
```

`\hds@memman` Head styles used in the *The Memoir Class* user guide. In this, as the later ones, only changes from the defaults need specifying.

```
\makeheadstyles{memman}{%
% book changes
  \renewcommand*{\booknamefont}{\normalfont\huge\sffamily}
  \renewcommand*{\booknumfont}{\normalfont\huge\sffamily}
  \renewcommand*{\booktitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\midbookskip}{\par\vskip 2\onelineskip}%
% part changes
  \renewcommand*{\partnamefont}{\normalfont\huge\sffamily}
  \renewcommand*{\partnumfont}{\normalfont\huge\sffamily}
  \renewcommand*{\parttitlefont}{\normalfont\Huge\sffamily}
  \renewcommand*{\midpartskip}{\par\vskip 2\onelineskip}%
% chapter
  \chapterstyle{demo3}
% section
  \setbeforesecskip{-1.333\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{0.667\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsecheadstyle{\normalfont\scshape\raggedright}%
  \setsecheadstyle{\normalfont\scshape\memRTLraggedright}%
% subsection
  \setbeforesubsecskip{-0.667\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsubsecheadstyle{\normalfont\bfseries\raggedright}%
  \setsubsecheadstyle{\normalfont\bfseries\memRTLraggedright}%
% subsubsection
  \setbeforesubsubsecskip{-0.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsubsubsecheadstyle{\normalfont\normalsize\itshape\raggedright}%
  \setsubsubsecheadstyle{\normalfont\normalsize\itshape\memRTLraggedright}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
```

```
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}

```

\hds@bringhurst Head styles based on Bringhurst's book

```
\makeheadstyles{bringhurst}{%
% chapter
\chapterstyle{bringhurst}
% section
  \setbeforesecskip{-1\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{1\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsecheadstyle{\normalfont\raggedright\scshape\MakeLowercase}%
  \setsecheadstyle{\normalfont\memRTLraggedright\scshape\MakeLowercase}%
% subsection
  \setbeforesubsecskip{-1.0\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{1.0\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsubsecheadstyle{\sethangfrom{\noindent ####1}\normalfont\itshape\raggedright}%
  \setsubsecheadstyle{\sethangfrom{\noindent ####1}\normalfont\itshape\memRTLraggedright}%
% subsubsection
  \setbeforesubsubsecskip{1.0\onelineskip
                          \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubsubsecskip{-1em}%
  \setsubsubsecheadstyle{\normalfont\normalsize\scshape\MakeLowercase}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}

```

\hds@crosshead Head styles based on one of Bringhurst's suggestions.

```
\makeheadstyles{crosshead}{%
  \chapterstyle{crosshead}
% section
  \setbeforesecskip{-1.25\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{0.75\onelineskip \@plus 0.1\onelineskip}%
  \setsecheadstyle{\normalfont\centering\MakeUppercase}%
% subsection
  \setbeforesubsecskip{-1.25\onelineskip
```

```
                        \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.75\onelineskip \@plus 0.1\onelineskip}%
  \setsubsecheadstyle{\normalfont\centering\bfseries}%
% subsubsection
  \setbeforesubsubsecskip{-.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{.333\onelineskip \@plus 0.1\onelineskip}%
  \setsubsubsecheadstyle{\normalfont\normalsize\centering\scshape\MakeLowercase}%
% paragraph
  \setbeforeparaskip{-.667\onelineskip
                     \@plus -02.5\onelineskip \@minus -0.25\onelineskip}%
  \setafterparaskip{.333\onelineskip \@plus 0.1\onelineskip}%
  \setparaheadstyle{\normalfont\normalsize\centering\itshape}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\scshape\MakeLowercase}}

```

`\hds@dowding` Head styles based on Dowding's *Finer Points*.

```
\makeheadstyles{dowding}{%
% chapter
  \chapterstyle{dowding}
% section
  \setbeforesecskip{-2\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{1\onelineskip \@plus 0.1\onelineskip}%
  \setsecheadstyle{\normalfont\centering\MakeUppercase}%
% subsection
  \setbeforesubsecskip{-1.2\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.8\onelineskip \@plus 0.1\onelineskip}%
  \setsubsecheadstyle{\normalfont\scshape\centering\MakeLowercase}%
% subsubsection
  \setbeforesubsubsecskip{-0.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
  \setsubsubsecheadstyle{\normalfont\normalsize\centering\itshape}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
```

```
4754   \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}
4755
```

`\hds@komalike` Head styles based on KOMA script classes (`scrbook.cls`).

```
4756 \makeheadstyles{komalike}{%
4757 % part
4758   \renewcommand*{\partnamefont}{\huge\sffamily\bfseries}%
4759   \renewcommand*{\partnumfont}{\huge\sffamily\bfseries}%
4760   \renewcommand*{\parttitlefont}{\huge\sffamily\bfseries}%
4761 % chapter
4762   \chapterstyle{komalike}
4763 % section
4764   \setbeforesecskip{-3.5ex \@plus -1ex \@minus -.2ex}%
4765   \setaftersecskip{2.3ex \@plus .2ex}%
4766 %%%  \setsecheadstyle{\normalfont\Large\sffamily\bfseries\raggedright}%
4767   \setsecheadstyle{\normalfont\Large\sffamily\bfseries\memRTLraggedright}%
4768 % subsection
4769   \setbeforesubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4770   \setaftersubsecskip{1.5ex \@plus .2ex}%
4771 %%%  \setsubsecheadstyle{\normalfont\large\sffamily\bfseries\raggedright}%
4772   \setsubsecheadstyle{\normalfont\large\sffamily\bfseries\memRTLraggedright}%
4773 % subsubsection
4774   \setbeforesubsubsecskip{-3.25ex \@plus -1ex \@minus -.2ex}%
4775   \setaftersubsubsecskip{1.5ex \@plus .2ex}%
4776 %%%  \setsubsubsecheadstyle{\normalfont\normalsize\sffamily\bfseries\raggedright}%
4777   \setsubsubsecheadstyle{\normalfont\normalsize\sffamily\bfseries\memRTLraggedright}%
4778 % paragraph
4779   \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4780   \setafterparaskip{-1em}%
4781   \setparaheadstyle{\normalfont\normalsize\sffamily\bfseries}%
4782 % subparagraph
4783   \setsubparaindent{\parindent}%
4784   \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
4785   \setaftersubparaskip{-1em}%
4786   \setsubparaheadstyle{\normalfont\normalsize\sffamily\bfseries}}
4787
```

`\hds@ntglike` Head styles based on the NTG classes (`boek.cls`).

```
4788 \makeheadstyles{ntglike}{%
4789 % part
4790   \renewcommand*{\partnamefont}{\Large\bfseries\MakeUppercase}%
4791   \renewcommand*{\partnumfont}{\Large\bfseries}%
4792   \renewcommand*{\parttitlefont}{\Large\MakeUppercase}%
4793 % chapter
4794   \chapterstyle{ntglike}
4795 % section
4796   \setbeforesecskip{-2\onelineskip
4797                     \@plus -1\onelineskip \@minus -.5\onelineskip}%
4798   \setaftersecskip{0.5\onelineskip}%
4799   \setsecheadstyle{\normalfont\large\bfseries}%
```

```
% subsection
  \setbeforesubsecskip{-1\onelineskip
                       \@plus -.5\onelineskip \@minus -.25\onelineskip}%
  \setaftersubsecskip{0.01\onelineskip}%
  \setsubsecheadstyle{\normalfont\normalsize\bfseries}%
% subsubsection
  \setbeforesubsubsecskip{-1\onelineskip
                          \@plus -.5\onelineskip \@minus -.25\onelineskip}%
  \setaftersubsubsecskip{0.01\onelineskip}%
  \setsubsubsecheadstyle{\normalfont\normalsize\slshape}%
% paragraph
  \setbeforeparaskip{3.25ex \@plus 1ex \@minus .2ex}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\slshape}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{3.25ex \@plus 1ex \@minus .2ex}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\slshape}}

```

\hds@tandh Head styles based on Thames & Hudson *Typography*.

```
\makeheadstyles{tandh}{%
% chapter
  \chapterstyle{tandh}
% section
  \setbeforesecskip{-2\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{1\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsecheadstyle{\normalfont\raggedright\MakeUppercase}%
  \setsecheadstyle{\normalfont\memRTLraggedright\MakeUppercase}%
% subsection
  \setbeforesubsecskip{-1.2\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.8\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsubsecheadstyle{\normalfont\Large\itshape\raggedright}%
  \setsubsecheadstyle{\normalfont\Large\itshape\memRTLraggedright}%
% subsubsection
  \setbeforesubsubsecskip{-0.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
%%%  \setsubsubsecheadstyle{\normalfont\normalsize\bfseries\raggedright}%
  \setsubsubsecheadstyle{\normalfont\normalsize\bfseries\memRTLraggedright}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
```

```
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}

```

\hds@wilsondob Head styles based on Adrian Wilson's *The Design of Books*.

```
\makeheadstyles{wilsondob}{%
% chapter
  \chapterstyle{wilsondob}
% section
  \setbeforesecskip{-1.333\onelineskip
                    \@plus -0.5\onelineskip \@minus -.5\onelineskip}%
  \setaftersecskip{0.667\onelineskip \@plus 0.1\onelineskip}%
%%  \setsecheadstyle{\normalfont\raggedright\MakeUppercase}%
  \setsecheadstyle{\normalfont\memRTLraggedright\MakeUppercase}%
% subsection
  \setbeforesubsecskip{-0.667\onelineskip
                       \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
%%  \setsubsecheadstyle{\normalfont\Large\itshape\raggedright}%
  \setsubsecheadstyle{\normalfont\Large\itshape\memRTLraggedright}%
% subsubsection
  \setbeforesubsubsecskip{-0.667\onelineskip
                          \@plus -0.25\onelineskip \@minus -0.25\onelineskip}%
  \setaftersubsubsecskip{0.333\onelineskip \@plus 0.1\onelineskip}%
%%  \setsubsubsecheadstyle{\normalfont\normalsize\raggedright\scshape\MakeLowercase}%
  \setsubsubsecheadstyle{\normalfont\normalsize\memRTLraggedright\scshape\MakeLowercase}%
% paragraph
  \setbeforeparaskip{1.0\onelineskip
                     \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setafterparaskip{-1em}%
  \setparaheadstyle{\normalfont\normalsize\itshape\addperiod}%
% subparagraph
  \setsubparaindent{\parindent}%
  \setbeforesubparaskip{1.0\onelineskip
                        \@plus 0.5\onelineskip \@minus 0.2\onelineskip}%
  \setaftersubparaskip{-1em}%
  \setsubparaheadstyle{\normalfont\normalsize\itshape\addperiod}}

```

12.10 Appendices

\ifanappendix \anappendixtrue \anappendixfalse \appendix

In the standard book class the \appendix command does the following:

- Resets the chapter and section counters to zero
- Sets \@chapapp to \appendixname.
- Redefines \thechapter to produce alphabetic appendix numbers.

```
\newif\ifanappendix
  \anappendixfalse
\newcommand{\appendix}{\par
  \setcounter{chapter}{0}%
  \setcounter{section}{0}%
  \gdef\@chapapp{\appendixname}%
  \gdef\thechapter{\@Alph\c@chapter}%
  \anappendixtrue}
```

This class provides additional appendixing capabilities.

`\appendixpage` `\appendixpage*` The command to typeset a page announcing the start of the appendices. It is based on the `\part` definition. The normal version makes an entry in the ToC but the starred version does not.

```
\newcommand{\appendixpage}{%
  \@ifstar{\@sapppage}{\@apppage}}
```

`\memapppageinfo` `\memapppagestarinfo` `\memapppageinfo{\appendixpagename}` `\memapppagestarinfo{\appendixpagename}`

```
\newcommand{\memapppageinfo}[1]{}
\newcommand{\memapppagestarinfo}[1]{}
```

`\@apppage` `\@apppage` typesets an appendix page and makes an entry in the ToC.

```
\def\@apppage{%
  \@setuppart
  \addappheadtotoc
  \partmark{\appendixpagename}%
  \memapppageinfo{\appendixpagename}%
  {\centering
   \interlinepenalty \@M
   \normalfont
   \printparttitle{\appendixpagename}\par}%
  \@endpart}
```

`\@sapppage` `\@sapppage` typesets an appendix page and does not make an entry in the ToC.

```
\def\@sapppage{%
  \@setuppart
  \partmark{\appendixpagename}%
  \memapppagestarinfo{\appendixpagename}%
  {\centering
   \interlinepenalty \@M
   \normalfont
   \printparttitle{\appendixpagename}\par}%
  \@endpart}
```

\addappheadtotoc This command adds an 'appendices' line to the ToC. The style is the same as used in tocbibind for the 'List of figures' line. That is, as a Chapter heading.

```
\def\addappheadtotoc{%
  \phantomsection\addcontentsline{toc}{chapter}{\appendixtocname}}
```

\@ppsavesec
\@pprestoresec
\@ppsaveapp
\restoreapp For the appendices environment we need to save and restore the main document division number and the appendix number. The \restoreapp command is the one for the user.

```
\newcounter{@ppsavesec}
\newcounter{@ppsaveapp}
\setcounter{@ppsaveapp}{0}
\newcommand{\@ppsavesec}{%
  \setcounter{@ppsavesec}{\value{chapter}}}
\newcommand{\@pprestoresec}{%
  \setcounter{chapter}{\value{@ppsavesec}}}
\newcommand{\@ppsaveapp}{%
  \setcounter{@ppsaveapp}{\value{chapter}}}
\newcommand{\restoreapp}{%
  \setcounter{chapter}{\value{@ppsaveapp}}}

```

\@resets@pp Resets the appropriate sectioning counters and names. This does almost exactly what the default \appendix command does, except that it saves and restores sectional numbering. It saves the sectional number at the start and restores the appendix number at the end.

```
\newcommand{\@resets@pp}{%
  \par
  \@ppsavesec
  \setcounter{section}{0}%
  \setcounter{chapter}{0}%
  \renewcommand\@chapapp{\appendixname}%
  \renewcommand\thechapter{\@Alph\c@chapter}%
  \restoreapp
}

```

appendices This is the heart of the package. Start it off by doing the resetting done by the \appendix command but also save the main counters. At the end of the environment save the appendix number and restore the main counters.

```
\newenvironment{appendices}%
  {\@resets@pp\anappendixtrue}%
  {\@ppsaveapp\@pprestoresec\anappendixfalse}

```

\setthesection The user commands for specifying the numbering style for subappendices.

```
\newcommand{\setthesection}{\thechapter.\Alph{section}}

```

\@resets@ppsub Similar to \@resets@pp except that it is for use within the subappendices environment; as such, it is a bit simpler.

```
\newcommand{\@resets@ppsub}{
  \par
  \setcounter{section}{0}
  \renewcommand{\thesection}{\setthesection}
}
```

\ifnamesubappendix Switch for adding an 'appendix' name before subappendix numbers.

\namesubappendixtrue
\namesubappendixfalse
\namedsubappendices
\unnamedsubappendices

```
\newif\ifnamesubappendix
  \namesubappendixfalse
\newcommand*{\namedsubappendices}{\namesubappendixtrue}
\newcommand*{\unnamedsubappendices}{\namesubappendixfalse}
```

subappendices The environment for subappendices. Start it off by doing the resetting of the \section command.

```
\newenvironment{subappendices}{%
  \@resets@ppsub
```

Change the definition of \addappheadtotoc to give a section entry.

```
  \def\addappheadtotoc{\phantomsection
    \addcontentsline{toc}{section}{\appendixtocname}}
```

To implement the naming we do cunning things with the \@seccntformat command.

```
  \ifnamesubappendix
    \def\sectionname{\appendixname}
    \def\@seccntformat##1{\@ifundefined{##1name}%
                          {}{\csname ##1name\endcsname\ }%
        \csname the##1\endcsname\quad}
  \fi
```

That's it.

```
  }{}
```

\@formatsecmark@pp Formats the page header for a redefined \sectionmark.

```
\newcommand{\@formatsecmark@pp}[1]{%
  \MakeUppercase{\appendixname\space
    \ifnum \c@secnumdepth >\z@
      \thesection\quad
    \fi
    #1}}
```

12.11 Appendixpage-like pages

This capability was suggested to me by Lars Madsen on 2004/11/28.

\leadpagetoclevel
\newleadpage
\renewleadpage

\newleadpage[<page-style>]{<cmdname>}{<title>} creates new macros called \cmdname and \cmdname* that when called typeset a page like an Appendixpage, with a title <title> using page style <page-style> (default empty). The plain version adds an entry to the ToC but the starred \cmdname* does not. \renewleadpage changes the definitions.
The ToC entry style is set by \leadpagetoclevel (default chapter). To have a part-type entry:
\renewcommand*{\leadpagetoclevel}{part}.
The \partmark command is used if you need to mark the title.

```
\newcommand*{\leadpagetoclevel}{chapter}
\newcommand*{\newleadpage}[3][empty]{%
  \@namedef{#2}{\@ifstar{\dlfm@msapppage{#1}{#2}{#3}}%
                        {\dlfm@mapppage{#1}{#2}{#3}}}}
\newcommand*{\renewleadpage}[3][empty]{%
  \@namedef{#2}{\@ifstar{\dlfm@msapppage{#1}{#2}{#3}}%
                        {\dlfm@mapppage{#1}{#2}{#3}}}}

```

\memleadpageinfo
\memleadpagestarinfo

\memleadpageinfo{pstyle}{name}{title}
\memleadpagestarinfo{pstyle}{name}{title}

```
\newcommand{\memleadpageinfo}[3]{}
\newcommand{\memleadpagestarinfo}[3]{}

```

\dlfm@msapppage
\dlfm@mapppage

Implement the starred and regular versions of \(re)newleadpage

```
\newcommand*{\dlfm@msapppage}[3]{%
  \@setuppart
  \partmark{#3}%
  \memleadpagestarinfo{#1}{#2}{#3}%
  {\centering
  \interlinepenalty \@M
  \normalfont
  \printparttitle{#3}\par
  \thispagestyle{#1}}%
  \dlfm@m@endpart{#1}}
\newcommand*{\dlfm@mapppage}[3]{%
  \@setuppart
  \phantomsection
  \addcontentsline{toc}{\leadpagetoclevel}{#3}%
  \partmark{#3}%
  \memleadpageinfo{#1}{#2}{#3}%
  {\centering
  \interlinepenalty \@M
  \normalfont
  \printparttitle{#3}\par
  \thispagestyle{#1}}%
  \dlfm@m@endpart{#1}}

```

\dlfm@m@endpart Finishes off a part-like page.

```
\newcommand*{\dlfm@m@endpart}[1]{%
  \if@twoside
    \if@openright
      \null
      \thispagestyle{#1}%
      \newpage
    \fi
  \fi
  \if@tempswa
    \twocolumn
  \fi}
```

12.12 Paragraphs

Much of the code in this section is taken from my *Glisterings* columns [Wil07, Wil08].

\memorigdbs \memorigpar \atcentercr \memorigdbs saves the original definition of \\ and \memorigpar saves the original \par. The macro \atcentercr provides a user call to \@centercr. These could come in handy for odd paragraph shapes.

```
\let\memorigdbs\\
\let\memorigpar\par
\let\atcentercr\@centercr
```

12.12.1 Normal (block) paragraphs

\flushleftright Sets the paragraphing to LaTeX's normal form.

```
\newcommand*{\flushleftright}{%
  \let\\\memorigdbs
  \leftskip\z@skip
  \rightskip\leftskip
  \parfillskip\@flushglue
  \everypar{}}
```

\linenottooshort This declaration \linenottooshort[⟨*length*⟩] specifies paragraphs such that the last line is at least about ⟨*length*⟩ long (the default is 4em).

```
\newcommand*{\linenottooshort}[1][4em]{%
  \@tempdima=\hsize
  \advance\@tempdima -#1
  \leftskip\z@skip
  \rightskip\leftskip
  \parfillskip=\@tempdima \@minus \@tempdima}
```

\russianpar Using \russianpar instead of \par to end a paragraph causes it to be set according to Russian typography, where the last line of a multiline paragraph must be either at least as long as \parindent and have at least \parindent space at the right, or it must be flushleft and flushright.

```
\newcommand*{\russianpar}{\ifhmode\unskip
  \strut\vadjust{}\nobreak
  \discretionary{}%
  {\hbox{\hskip2\parindent
         \vrule depth 273sp width 0sp height \ht\strutbox}}%
  {\hbox{\hskip\parindent}}%
  \hskip-2\parindent \@minus 2\parindent
  \hskip\hsize \@minus \hsize
  \kern\z@ \parfillskip\z@
  \memorigpar
  \ifdim\prevdepth=273sp
    \nobreak
    \vskip-2\baselineskip
    \hbox{\strut}%
  \fi\fi}

```

\lastlineparrule \lastlinerulefill Using \lastlinerulefill instead of \par to end a paragraph causes all short lines to be filled at the right by a rule (\lastlineparrule) extending to the righthand margin.

```
\newcommand*{\lastlineparrule}{%
  \hrule height 0.5ex depth \@tempdimb\relax}
\newcommand*{\lastlinerulefill}{%
  \let\\\@centercr
  \@tempdimb=-0.5ex \advance\@tempdimb 0.4pt
  \unskip\nobreak\space
  \leaders\lastlineparrule\hskip\@flushglue
  \vadjust{}{\parfillskip\z@\memorigpar}}

```

12.12.2 Centered lines

\centerlastline This declaration specifies normal paragraphs except that the last line of each is centered.

```
\newcommand*{\centerlastline}{%
%%%  \leftskip\@flushglue
  \memRTLleftskip\@flushglue
%%%  \rightskip=\z@ plus -1fil
  \memRTLrightskip=\z@ plus -1fil
  \parfillskip=\z@ plus 2fil}

```

\leftcenterright This declaration specifies paragraphs where the first line is flushleft (raggedright), the last is flushright (raggedleft) and all inbetween are centered. Set \everypar{} afterwards.

```
\newcommand*{\leftcenterright}{%;
  \let\\\break
  \parindent\z@
  \leftskip\@flushglue
  \rightskip\leftskip
  \parfillskip \z@ \@plus -1fil
  \everypar={\hskip \z@ \@plus -1fil}}

```

\centerfloat This is a version of \centering that can be used to center a wide float with respect to the text block (normally the left of a wide float is aligned with the left of the textblock). This can only be used for centering something where LaTeX knows the width (e.g., a figure or a table / tabular).
This is a modified version of code by Robin Fairbairns (CTT, *Re: Centering a table: problem with rotating.sty, maybe a strange document class?*, 3 Jan 2009).

```
\newcommand*{\centerfloat}{%
  \parindent \z@
  \leftskip \z@ \@plus 1fil \@minus \textwidth
  \rightskip\leftskip
  \parfillskip \z@skip}

```

12.12.3 Ragged

The kernel code for raggedright (in `ltmiscen.dtx`):

```
\def\raggedright{%
  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
  \leftskip\z@skip
  \parindent\z@}
%% \@flushglue = 0pt plus 1fil      %% from ltalloc.dtx
%% \z@skip = 0pt plus 0pt minus 0pt %%
```

produces very ragged text with no paragraph indent.

\ragrparindent \raggedyright[⟨*plus*⟩] provides controllable ragged right paragraphs.
\raggedyright

```
\newdimen\ragrparindent
  \setlength{\ragrparindent}{\parindent}
\newcommand{\raggedyright}[1][2em]{%
  \let\\\@centercr\@rightskip \z@ \@plus #1\relax
%%%  \rightskip\@rightskip
  \memRTLrightskip\@rightskip
%%%  \leftskip\z@skip
  \memRTLleftskip\z@skip
  \parindent\ragrparindent}

```

\justlastraggedleft This declaration specifies paragraphs where the lines are justified, except for the last which is raggedleft (flushright)..

```
\newcommand*{\justlastraggedleft}{%
%%%  \leftskip\@flushglue
  \memRTLleftskip\@flushglue
%%%  \rightskip-\leftskip
  \memRTLrightskip-\memRTLleftskip
  \parfillskip\leftskip
  \parindent \z@}

```

\raggedrightthenleft This declaration specifies paragraphs where the first line is raggedright (flushleft) and all the rest are raggedleft (flushright). Note that this alters \everpar, which may need to be reset afterwards to \everypar{}.

```
\newcommand*{\raggedrightthenleft}{%
  \parindent \z@
%%%  \leftskip \z@ \@plus 1fill
  \memRTLleftskip \z@ \@plus 1fill
%%%  \rightskip\@flushglue
  \memRTLrightskip\@flushglue
  \parfillskip \z@
  \everypar{\hskip \z@ \@plus -1fill}}

```

12.12.4 Hanging

\hangfrom This is a user-level version of the kernel \@hangfrom macro (only the name is changed) as defined in ltsec.dtx.
\hangfrom{⟨*text*⟩} puts ⟨*text*⟩ in a box and makes a hanging paragraph of the following material (a bit like a description item).

```
\newcommand{\hangfrom}[1]{%
  \setbox\@tempboxa\hbox{{#1}}%
  \hangindent \wd\@tempboxa\noindent\box\@tempboxa}

```

\hangpara \hangpara{⟨*indent*⟩}{⟨*afternum*⟩} at the start of a paragraph will make it hung. If ⟨*indent*⟩ is positive the left will be indented, otherwise the right. If ⟨*afternum*⟩, say N, is positive the N+1 th lines onwards will be indented. For N negative, the first N lines will be indented.

```
\newcommand{\hangpara}[2]{\hangindent#1\hangafter#2\noindent}

```

hangparas \begin{hangparas}{⟨*indent*⟩}{⟨*afternum*⟩} hangs a series of paragraphs.

```
\newenvironment{hangparas}[2]{\setlength{\parindent}{\z@}
  \everypar={\hangpara{#1}{#2}}}{\par}

```

12.12.5 Miscellaneous

\leftspringright \leftspringright{⟨*leftfrac*⟩}{⟨*rightfrac*⟩}{⟨*lefttext*⟩}{⟨*righttext*⟩} sets the ⟨*lefttext*⟩ flushleft (raggedright) in a column ⟨*leftfrac*⟩ of the current textwidth

and the ⟨*righttext*⟩ flushright (raggedleft) in a column ⟨*rightfrac*⟩ of the textwidth, with space (1.0 - ⟨*leftfrac*⟩ - ⟨*rightfrac*⟩) of the textwidth between them. Both ⟨*leftfrac*⟩ and ⟨*rightfrac*⟩ must be given as decimal numbers (e.g., `0.25` not `1/4`).

```
\newcommand{\leftspringright}[4]{%
  \@tempdimb=\hsize
  \par\noindent\hbox to\@tempdimb{%
    \vtop{\hsize=#1\@tempdimb \flushleft#3\par}\hss
    \vtop{\hsize=#2\@tempdimb \flushright#4\par}}}

```

`\sourceatright` Putting `\sourceatright[`⟨*length*⟩`]{`⟨*text*⟩`}` at the end of a paragraph will set ⟨*text*⟩ flushright on the same line provided the line is short enough to allow ⟨*length*⟩ (default 2em) between the end of the line and ⟨*text*⟩. If there is not enough space then ⟨*text*⟩ is set flushright on the following line.

```
\newcommand*{\sourceatright}[2][2em]{{%
  \unskip\nobreak\hfil\penalty50
  \hskip#1\hbox{}\nobreak\hfil{#2}
  \parfillskip\z@\finalhyphendemerits=0\par}}

```

13 Lists

13.1 General List Parameters

The following commands are used to set the default values for the list environment's parameters. See the LaTeX manual for an explanation of the meanings of the parameters. Defaults for the list environment are set as follows. First, `\rightmargin`, `\listparindent` and `\itemindent` are set to 0pt. Then, for a Kth level list, the command `\@listK` is called, where 'K' denotes 'i', ''i', ... , 'vi'. (I.e., `\@listiii` is called for a third-level list.) By convention, `\@listK` should set `\leftmargin` to `\leftmarginK`.

`\leftmargin` `\leftmargini` `\leftmarginii` `\leftmarginiii` `\leftmarginiv` `\leftmarginv` `\leftmarginvi` For efficiency, level-one list's values are defined at top level, and `\@listi` is defined to set only `\leftmargin`. In two column mode the margins should be smaller than in one column typesetting.

```
\if@twocolumn
  \setlength{\leftmargini}{2em}
\else
  \setlength{\leftmargini}{2.5em}
\fi
```

The value of `\leftmargin` has to be set at this outer level.

```
\leftmargin \leftmargini
```

Lower level list margins are calculated so that they are indented more than the label in an outer list.

```
\setlength{\leftmarginiii}{2.2em}
\setlength{\leftmarginiiii}{1.87em}
\setlength{\leftmarginiv}{1.7em}
\if@twocolumn
  \setlength{\leftmarginv}{.5em}
  \setlength{\leftmarginvi}{.5em}
\else
  \setlength{\leftmarginv}{1em}
  \setlength{\leftmarginvi}{1em}
\fi
```

`\itemindent` Here we set the `\itemindent` which is the extra indentation before a label.

```
\setlength{\itemindent}{\z@}
```

`\labelsep` `\labelsep` is the distance between the label and the text of an item;
`\labelwidth` `\labelwidth` is the width of the label.

```
\setlength{\labelsep}{0.5em}
\setlength{\labelwidth}{\leftmargini}
  \addtolength{\labelwidth}{-\labelsep}
```

`\@beginparpenalty` These penalties are inserted before and after a list or paragraph environment.
`\@endparpenalty` They are set to a bonus value to encourage page breaking at these points.

`\@itempenalty` This penalty is inserted between list items.

```
\@beginparpenalty -\@lowpenalty
\@endparpenalty   -\@lowpenalty
\@itempenalty     -\@lowpenalty

```

`\everylistparindent` The kernel sets `\listparindent` to zero within a `\list`, where it can be
`\list` overridden in `\list`'s second argument. Here it is set to `\everyparlistindent` by default, which in turn is 0pt.

```
\newdimen\everylistparindent
  \everylistparindent \z@
\renewcommand*{\list}[2]{%
  \ifnum \@listdepth >5\relax
    \@toodeep
  \else
    \global\advance\@listdepth\@ne
  \fi
  \rightmargin\z@
  \listparindent\everylistparindent
  \itemindent\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \def\@itemlabel{#1}%
  \let\makelabel\@mklab
  \@nmbrlistfalse
  #2\relax
  \@trivlist
```

```
  \parskip\parsep
  \parindent\listparindent
  \advance\linewidth -\rightmargin
  \advance\linewidth -\leftmargin
  \advance\@totalleftmargin \leftmargin
  \parshape \@ne \@totalleftmargin \linewidth
  \ignorespaces}
```

\parsepi
\topsepi
\itemsepi
\parsepii
\topsepii
\topsepiii

Lists may be nested and the exact layout depends on the level of nesting. These lengths are used to control the nesting-level aspects.

```
\newlength{\parsepi}
\newlength{\topsepi}
\newlength{\itemsepi}
\newlength{\parsepii}
\newlength{\topsepii}
\newlength{\topsepiii}
```

\itemsepii
\itemsepiii
\partopsepii
\partopsepii

We need some new lengths for lists to cater for non-zero \parskip.

```
\newlength{\itemsepii}
\newlength{\itemsepiii}
\newlength{\partopsepii}
\newlength{\partopsepiii}
```

\setnzplist

Common code for non-zero \parskip in lists.

```
\newcommand*{\setnzplist}{%
  \partopsep \p@ \@plus\z@ \@minus\p@
  \topsepi\z@
  \parsepi\parskip
  \itemsepi\z@
  \topsepii\z@
  \parsepii\parskip
  \itemsepii\z@
  \topsepiii\z@
%%    \parsepiii\parskip
  \itemsepiii\z@}
```

\defaultlists

The standard LaTeX classes have list parameters that give some separation between lists and \items in lists. This macro sets those values. This is a simplification of memoir's original, and will apply to any font size.

```
\newcommand*{\defaultlists}{%
  \setlength{\partopsep}{0.2\onelineskip \@plus 0.1\onelineskip
                                         \@minus 0.1\onelineskip}%
  \parsepi = 0.3333\onelineskip \@plus 0.1667\onelineskip \@minus \p@
  \itemsepi = \parsepi
  \topsepi = 0.6667\onelineskip \@plus 0.3333\onelineskip
                                \@minus 0.2\onelineskip
```

```
  \parsepii = 0.1667\onelineskip \@plus \p@ \@minus \p@
  \topsepii = \parsepi
  \topsepiii = \parsepii
  \everylistparindent \listparindent
```

Additional code to cater for non-zero \parskips.

```
  \itemsepii\parsepii
  \itemsepiii\topsepiii
  \partopsepiii \p@ \@plus\z@ \@minus\p@
  \ifm@mnzpskip
    \setnzplist
  \fi}
\defaultlists
```

\firmlists
\firmlists*
\m@msfirmlists
\m@mfirmlists

These give approximately half the vertical spacing of the default lists, with all spaces equal. The starred version allows slightly less space before and after the list when it is preceded by a blank line.

```
\newcommand*{\firmlists}{%
  \@ifstar{\m@msfirmlists}{\m@mfirmlists}}

\newcommand*{\m@msfirmlists}{
  \setlength{\partopsep}{\z@ \@plus \p@ \@minus \p@}%
  \parsepi = 0.1667\onelineskip \@plus 0.0833\onelineskip \@minus \p@
  \itemsepi = \parsepi
  \topsepi = \parsepi
  \parsepii = 0.0833\onelineskip \@plus \p@ \@minus \p@
  \topsepii = \parsepi
  \topsepiii = \parsepii
  \everylistparindent\listparindent}

\newcommand*{\m@mfirmlists}{
  \setlength{\partopsep}{0.1\onelineskip \@plus 0.05\onelineskip
                                         \@minus 0.05\onelineskip}%
  \parsepi = 0.1667\onelineskip \@plus 0.0833\onelineskip \@minus \p@
  \itemsepi = \parsepi
  \topsepi = \parsepi
  \parsepii = 0.0833\onelineskip \@plus \p@ \@minus \p@
  \topsepii = \parsepi
  \topsepiii = \parsepii
  \everylistparindent\listparindent}
```

\tightlists
\tightlists*
\m@mstightlists
\m@mtightlists

This macro sets the parameters for lists that have less open vertical space in them. I think that these look neater than the defaults. Effectively, no additional vertical space is added. The starred version allows slightly no extra space before and after the list when it is preceded by a blank line, whereas the unstarred version puts half a \onelineskip before *and* after.

```
\newcommand*{\tightlists}{%
```

```
  \@ifstar{\m@mstightlists}{\m@mtightlists}

\newcommand*{\m@mstightlists}{%
  \setlength{\partopsep}{\z@ \@plus \p@ \@minus \p@}%
  \parsepi = \z@ \@plus \p@ \@minus \p@
  \itemsepi = \parsepi
  \topsepi = \z@ \@plus \p@ \@minus \p@
  \parsepii = \z@ \@plus \p@ \@minus \p@
  \topsepii = \parsepi
  \topsepiii = \parsepii
  \everylistparindent\parindent
  \ifm@mnzpskip
    \setnzplist
    \partopsepiii\partopsep
  \fi}

\newcommand*{\m@mtightlists}{%
  \setlength{\partopsep}{0.5\onelineskip \@plus \p@ \@minus \p@}%
  \parsepi = \z@ \@plus \p@ \@minus \p@
  \itemsepi = \parsepi
  \topsepi = \z@ \@plus \p@ \@minus \p@
  \parsepii = \z@ \@plus \p@ \@minus \p@
  \topsepii = \parsepi
  \topsepiii = \parsepii
  \everylistparindent\parindent
  \ifm@mnzpskip
    \setnzplist
    \partopsepiii\partopsep
  \fi}

```

`\firmlist` `\tightlist`

These two macros can be used at the start of a list environment to reduce the vertical gaps. `\tightlist` removes all interior spaces while `\firmlist` only removes some.

```
\newcommand{\firmlist}{%
  \setlength{\itemsep}{0.5\itemsep}\setlength{\parskip}{0.5\parskip}}
\newcommand{\tightlist}{%
  \setlength{\itemsep}{0pt}\setlength{\parskip}{0pt}}

```

The space before and after a `trivlist` environment is controlled by the `\topsep` and `\partopsep` skips. There are several environments, such as `center`, that are defined as a `trivlist`.

`\m@msavetopsep` `\m@msavepartopsep` `\savetrivseps` `\restoretrivseps`

Two skips to store the `\topsep` and `\partopsep` values and a means of setting them and restoring them.

```
\newskip\m@msavetopsep
\newskip\m@msavepartopsep
\newcommand*{\savetrivseps}{%
```

```
  \m@msavetopsep\topsep
  \m@msavepartopsep\partopsep}
\newcommand*{\restoretrivseps}{%
  \topsep\m@msavetopsep
  \partopsep\m@msavepartopsep}
```

Save the initial \topsep and \partopsep values.

```
\savetrivseps
```

\zerotrivseps A macro to zero \topsep and \partopsep.

```
\newcommand*{\zerotrivseps}{%
  \topsep\z@
  \partopsep\z@}
```

\@listI \@listi \@listI defines top level and \@listi values of \leftmargin, \parsep, \topsep, and \itemsep

```
\def\@listi{\leftmargin\leftmargini
  \parsep\parsepi
  \topsep\topsepi
  \itemsep\itemsepi}
\let\@listI\@listi
```

We should initialise these parameters to the standard defaults

```
\defaultlists
\@listi
```

\@listii \@listiii \@listiv \@listv \@listvi Here are the same macros for the lower level lists.

```
\def\@listii{\leftmargin\leftmarginii
             \labelwidth\leftmarginii
             \advance\labelwidth-\labelsep
             \topsep\topsepii
             \parsep\parsepii
             \itemsep\parsepii}

\def\@listiii{\leftmargin\leftmarginiii
              \labelwidth\leftmarginiii
              \advance\labelwidth-\labelsep
              \topsep\topsepiii
              \parsep\z@
%%%              \itemsep\topsep
%%%              \partopsep \p@ \@plus\z@ \@minus\p@
              \itemsep\itemsepiii
              \partopsep\partopsepiii}

\def\@listiv{\leftmargin\leftmarginiv
             \labelwidth\leftmarginiv
             \advance\labelwidth-\labelsep}
```

```
5328
5329 \def\@listv{\leftmargin\leftmarginv
5330             \labelwidth\leftmarginv
5331             \advance\labelwidth-\labelsep}
5332
5333 \def\@listvi{\leftmargin\leftmarginvi
5334              \labelwidth\leftmarginvi
5335              \advance\labelwidth-\labelsep}
5336
```

13.2 Enumerate

The enumerate environment uses four counters: *enumi*, *enumii*, *enumiii* and *enumiv*, where *enumN* controls the numbering of the Nth level enumeration.

\theenumi
\theenumii
\theenumiii
\theenumiv

The counters are already defined in `latex.dtx`, but their representation is changed here.

```
5337 \renewcommand{\theenumi}{\@arabic\c@enumi}
5338 \renewcommand{\theenumii}{\@alph\c@enumii}
5339 \renewcommand{\theenumiii}{\@roman\c@enumiii}
5340 \renewcommand{\theenumiv}{\@Alph\c@enumiv}
```

\labelenumi
\labelenumii
\labelenumiii
\labelenumiv

The label for each item is generated by the commands `\labelenumi` ... `\labelenumiv`.

```
5341 \newcommand{\labelenumi}{\theenumi.}
5342 \newcommand{\labelenumii}{\theenumii)}
5343 \newcommand{\labelenumiii}{\theenumiii.}
5344 \newcommand{\labelenumiv}{\theenumiv.}
```

\p@enumii
\p@enumiii
\p@enumiv

The expansion of `\p@enumN\theenumN` defines the output of a `\ref` command when referencing an item of the Nth level of an enumerated list.

```
5345 \renewcommand{\p@enumii}{\theenumi}
5346 \renewcommand{\p@enumiii}{\theenumi(\theenumii)}
5347 \renewcommand{\p@enumiv}{\p@enumiii\theenumiii}
```

The following is taken directly from David Carlisle's enumerate package.
START OF DAVID CARLISLE'S CODE AND COMMENTARY
This package gives the enumerate environment an optional argument which determines the style in which the counter is printed.
An occurence of one of the tokens `A a I i` or `1` produces the value of the counter printed with (respectively) `\Alph \alph \Roman \roman` or `\arabic`. These letters may be surrounded by any strings involving any other TEX expressions, however the tokens `A a I i 1` must be inside a `{ }` group if they are not to be taken as special.

\@enlab

Internal token register used to build up the label command from the optional argument.

```
5348 \newtoks\@enLab
```

\@enQmark This just expands to a '?'. \ref will produce this, if no counter is printed.

```
\def\@enQmark{?}
```

The next four macros build up the command that will print the item label. They each gobble one token or group from the optional argument, and add corresponding tokens to the register \@enLab. They each end with a call to \@enloop, which starts the processing of the next token.

\@enLabel Add the counter to the label. #2 will be one of the 'special' tokens A a I i 1, and is thrown away. #1 will be a command like \Roman.

```
\def\@enLabel#1#2{%
  \edef\@enThe{\noexpand#1{\@enumctr}}%
  \@enLab\expandafter{\the\@enLab\csname the\@enumctr\endcsname}%
  \@enloop}
```

\@enSpace Add a space to the label. The tricky bit is to gobble the space token, as you can
\@enSp@ce not do this with a macro argument.

```
\def\@enSpace{\afterassignment\@enSp@ce\let\@memtempa= }
\def\@enSp@ce{\@enLab\expandafter{\the\@enLab\space}\@enloop}
```

\@enGroup Add a { } group to the label.

```
\def\@enGroup#1{\@enLab\expandafter{\the\@enLab{#1}}\@enloop}
```

\@enOther Add anything else to the label

```
\def\@enOther#1{\@enLab\expandafter{\the\@enLab#1}\@enloop}
```

\@enloop The body of the main loop. Eating tokens this way instead of using \@tfor lets
\@enloop@ you see spaces and **all** braces. \@tfor would treat a and {a} as special, but not {{a}}.

```
\def\@enloop{\futurelet\@entemp\@enloop@}
```

```
\def\@enloop@{%
  \ifx A\@entemp         \def\@memtempa{\@enLabel\Alph  }\else
  \ifx a\@entemp         \def\@memtempa{\@enLabel\alph  }\else
  \ifx i\@entemp         \def\@memtempa{\@enLabel\roman }\else
  \ifx I\@entemp         \def\@memtempa{\@enLabel\Roman }\else
  \ifx 1\@entemp         \def\@memtempa{\@enLabel\arabic}\else
  \ifx \@sptoken\@entemp \let\@memtempa\@enSpace         \else
  \ifx \bgroup\@entemp   \let\@memtempa\@enGroup         \else
  \ifx \@enum@\@entemp   \let\@memtempa\@gobble          \else
                         \let\@memtempa\@enOther
```

Hook for possible extensions

```
                         \@enhook
```

```
              \fi\fi\fi\fi\fi\fi\fi\fi
```

Process the current token, then look at the next.

```
  \@memtempa}
```

`\@enhook` Hook for possible extensions. Some packages may want to extend the number of special characters that are associated with counter representations. This feature was requested to enable Russian alphabetic counting, but here I give an example of a footnote symbol counter, triggered by `*`.
To enable a new counter type based on a letter, you just need to add a new `\ifx` clause by analogy with the code above. So for example to make `*` trigger footnote symbol counting. a package should do the following.
Initialise the hook, in case the package is loaded before enumerate.

```
\providecommand\@enhook{}
```

Add to the hook a new `\ifx` clause that associates `*` with the `\fnsymbol` counter command.

```
\g@addto@macro\@enhook{%
  \ifx *\@entemp
    \def\@memtempa{\@enLabel\fnsymbol}%
  \fi}
```

This code sequence should work whether it is loaded before or after this enumerate package. Any number of new counter types may be added in this way. At this point we just need initialise the hook, taking care not to over write any definitions another package may already have added. (PRW: as this is now in a class, it can be defined instead of `\provided`).

```
%% \providecommand\@enhook{}
  \newcommand\@enhook{}
```

`\enumerate` The new `enumerate` environment. This is the first half of the original enumerate environment. If there is an optional argument, call `\@@enum@` to define the label commands, otherwise call `\@enum@` which is the second half of the original definition.

```
\def\enumerate{%
  \ifnum \@enumdepth >3 \@toodeep\else
      \advance\@enumdepth \@ne
      \edef\@enumctr{enum\romannumeral\the\@enumdepth}\fi
  \@ifnextchar[{\@@enum@}{\@enum@}}
```

`\@@enum@` Handle the optional argument..

```
\def\@@enum@[#1]{%
```

Initialise the loop which will break apart the optional argument. The command to print the label is built up in `\@enlab`. `\@enThe` will be used to define `\theenum`n.

```
  \@enLab{}\let\@enThe\@enQmark
```

The `\@enum@` below is never expanded, it is used to detect the end of the token list.

```
  \@enloop#1\@enum@
```

Issue a warning if we did not find one of the 'special' tokens.

```
\ifx\@enThe\@enQmark\@warning{The counter will not be printed.%
 ^^J\space\@spaces\@spaces\@spaces The label is: \the\@enLab}\fi
```

Define \labelenum*n* and \theenum*n*.

```
\expandafter\edef\csname label\@enumctr\endcsname{\the\@enLab}%
\expandafter\let\csname the\@enumctr\endcsname\@enThe
```

Set the counter to 7 so that we get the width of 'vii' if roman numbering is in force then set \leftmargin*n*. to the width of the label plus \labelsep.

```
\csname c@\@enumctr\endcsname7
\expandafter\settowidth
           \csname leftmargin\romannumeral\@enumdepth\endcsname
           {\the\@enLab\hspace{\labelsep}}%
```

Finally call \@enum@ which is the second half of the original definition.

```
\@enum@}
```

\@enum@ All the list parameters have now been defined, so call \list. This is taken straight from the original definition of \enumerate.

```
\def\@enum@{\list{\csname label\@enumctr\endcsname}%
          {\usecounter{\@enumctr}\def\makelabel##1{\hss\llap{##1}}}}

```

END OF DAVID CARLISLE'S CODE AND COMMENTARY

13.3 Itemize

\labelitemi \labelitemii \labelitemiii \labelitemiii Itemization is controlled by the commands: \labelitemi, \labelitemii, etc., which define the labels of the various itemization levels: the symbols used are: bullet (•), bold en-dash (–), centered asterisk (∗), and centered dot (·).

```
\newcommand{\labelitemi}{\textbullet}
\newcommand{\labelitemii}{\normalfont\bfseries \textendash}
\newcommand{\labelitemiii}{\textasteriskcentered}
\newcommand{\labelitemiv}{\textperiodcentered}
```

It seems like a reasonable idea to give the itemize environment an optional argument to match enumerate. Fortunately this seems to be much simpler and I might even be able to work it out for myself.

itemize This is a hack at the kernel code for itemize.

```
\renewcommand{\itemize}[1][\@empty]{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth\@ne
    \ifx \@empty #1\else % optional argument
      \@namedef{labelitem\romannumeral\the\@itemdepth}{#1}%
    \fi
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \expandafter
```

```
    \list
      \csname\@itemitem\endcsname
       {\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\enditemize =\endlist
```

13.4 Description

The class defines two description environment, the standard one and a 'block' one, and also two semi-configurable versions.

`description` The description environment is defined here – while the default itemize and enumerate environments are defined in `latex.dtx`.

```
\newenvironment{description}%
               {\list{}{\labelwidth\z@ \itemindent-\leftmargin
                        \let\makelabel\descriptionlabel}}%
               {\endlist}
```

`\descriptionlabel` To change the formatting of the label, you must redefine `\descriptionlabel`.

```
\newcommand*{\descriptionlabel}[1]{\hspace\labelsep
                                   \normalfont\bfseries #1}
```

`blockdescription` The 'block' description environment.

```
\newenvironment{blockdescription}%
               {\list{}{\labelwidth\z@ \itemindent 0.5em \labelsep 0.5em
                        \let\makelabel\blockdescriptionlabel}}%
               {\endlist}
```

`\blockdescriptionlabel` In order to change the formatting of the label, you must redefine the macro `\blockdescriptionlabel`.

```
\newcommand*{\blockdescriptionlabel}[1]{%% \hspace\labelsep
                                   \normalfont\bfseries #1}
```

`labelled` This is a version of the description environment which takes the name, without the backslash, of some `\labelcode` as its argument. For example:

```
\newcommand*{\sflabel}[1]{\hspace\labelsep \normalfont\sffamily #1}
\begin{labelled}{sflabel}
  \item[label] using a sans font for the labels
```

```
\newenvironment{labelled}[1]%
  {\list{}{\labelwidth\z@ \itemindent-\leftmargin
           \def\m@malabel{\@nameuse{#1}} \let\makelabel\m@malabel}}%
  {\endlist}
```

flexlabelled

\begin{flexlabelled}{labelcode}{labelwidth}{labelsep}{itemindent}{leftmargin}{rightmargin}
lets you specify some of the parameters for a description-like list. The first argument is as for the labelled environment. The others are all lengths for the various parameters; a * instead of a length means that that argument is to be ignored.

```
\newenvironment{flexlabelled}[6]%
  {\list{}{\nametest{#2}{*}%
           \ifsamename\else \labelwidth #2  \fi
           \nametest{#3}{*}%
           \ifsamename\else \labelsep #3 \fi
           \nametest{#4}{*}%
           \ifsamename\else \itemindent #4 \fi
           \nametest{#5}{*}%
           \ifsamename\else \leftmargin #5 \fi
           \nametest{#6}{*}%
           \ifsamename\else  \rightmargin #6 \fi
           \def\m@malabel{\@nameuse{#1}} \let\makelabel\m@malabel}}%
  {\endlist}

```

13.5 Quotation

quotation The quotation environment is defined by making clever use of the list environment's parameters. The lines in the environment are set smaller than \textwidth. The first line of a paragraph inside this environment is indented.

```
\newenvironment{quotation}%
               {\list{}{\listparindent 1.5em%
                        \itemindent    \listparindent
                        \rightmargin   \leftmargin
                        \parsep        \z@ \@plus\p@}%
                \item[]}%
               {\endlist}
```

13.6 Quote

quote The quote environment is like the quotation environment except that paragraphs are not indented.

```
\newenvironment{quote}%
               {\list{}{\rightmargin\leftmargin}%
                \item[]}%
               {\endlist}
```

13.7 Theorem

This document class does not define it's own theorem environments, the defaults, supplied by latex.dtx are available.

13.8 Listing of symbols and abbreviations

Another element is the listing of symbols and abbreviations.

symbols
\symbollabel
```
\newcommand{\symbollabel}[1]{{#1 \hfill}}
\newenvironment{symbols}{\list{}%
    {\itemindent 0em \leftmargin 8em
     \labelsep 1em \labelwidth 5em
     \let\makelabel\symbollabel}}%
    {\endlist}
```

\symboldef
Within a symbols environment the command \symboldef{⟨*symbol*⟩}{⟨*meaning*⟩} is used to specify and explain each symbol or abbreviation.

```
\newcommand{\symboldef}[2]{\item[#1] #2}
```

14 Abstracts

Books usually do not have abstracts, but I decided to add in the code from the abstract package [Wil01a].
We just have the report or article style for the abstract with no titlepage option. The original code, from classes.dtx for this is:

```
\newenvironment{abstract}{%
  \if@twocolumn
    \section*{\abstractname}
  \else
    \small
    \begin{center}%
      {\bfseries \abstractname\vspace{-.5em}\vspace{\z@}}%%
    \end{center}%
    \quotation
  \fi}%
  {\if@twocolumn\else\endquotation\fi}
```

The following \if... commands are for implementing various options.

\if@bsonecol
\ifadd@bstotoc
\ifnumber@bs
\if@bsrunin
```
\newif\if@bsonecol
  \@bsonecoltrue
\newif\ifadd@bstotoc
  \add@bstotocfalse
\newif\ifnumber@bs
  \number@bsfalse
\newif\if@bsrunin
  \@bsruninfalse
```

\abstractcol
\abstractintoc
\abstractnum
\abstractrunin

These are the use-level commands for setting the options. If the abstract is runin, then it must not be numbered otherwise adding it to the ToC might result in something peculiar.

```
\newcommand{\abstractcol}{\@bsonecolfalse}
\newcommand{\abstractintoc}{\add@bstotoctrue}
\newcommand{\abstractnum}{\number@bstrue\@bsruninfalse}
\newcommand{\abstractrunin}{\@bsrunintrue\number@bsfalse}
```

The next set of macros comprise the implementation of the `abstract` environment.

\abstractnamefont
\abstracttextfont

These two macros are for specifying the fonts for typesetting the abstract's title and text. They are initialised for the default case (i.e., no class options).

```
\newcommand{\abstractnamefont}{\normalfont\small\bfseries}
\newcommand{\abstracttextfont}{\normalfont\small}
```

\abscolnamefont
\abscoltextfont

These two macros are for specifying the fonts for typesetting the abstract's title and text in a two column document where the abstarct is part of a column.

```
\newcommand{\abscolnamefont}{\normalfont\Large\bfseries}
\newcommand{\abscoltextfont}{\normalfont}
```

\absnamepos
\abstitleskip

`\absnamepos` specifies the environment in which the abstract's title name will be typeset, and the length `\abstitleskip` is an adjustement to the vertical space between the title and the abstract's text. These are initialised for the default case.

```
\newcommand{\absnamepos}{center}
\newlength{\abstitleskip} \setlength{\abstitleskip}{-0.5em}
```

\absleftindent
\abs@leftindent
\absrightindent
\absparindent
\absparsep

The abstract's text is typeset as a single item list, called `@bstr@ctlist`. These lengths set the left and right margin indents, the paragraph indentation, and the inter-paragraph vertical space. Their initial values are all class option-dependent.

```
\newlength{\absleftindent}
  \absleftindent=\leftmargin
\newdimen\abs@leftindent
  \abs@leftindent=\leftmargin
\newlength{\absrightindent}
  \absrightindent=\leftmargin
\newlength{\absparindent}
\newlength{\absparsep}
```

\abslabeldelim
\@bslabeldelim

The contents of `\abslabeldelim` are typeset after a run-in heading.

```
\newcommand{\abslabeldelim}[1]{\def\@bslabeldelim{#1}}
\abslabeldelim{}
```

`\@bsrunintitle` This macro typeset the run-in heading.

```
\newcommand{\@bsrunintitle}{%
  \hspace*{\abstitleskip}{\abstractnamefont\abstractname\@bslabeldelim}}

```

`\setup@bstract` Now arrange to set all the class option-dependent values.

```
\newcommand{\setup@bstract}{%
  \abs@leftindent=\absleftindent
  \if@twocolumn
```

Values for the twocolumn class option.

```
    \if@bsonecol
    \else
      \abs@leftindent=\z@
      \absrightindent=\z@
      \renewcommand*{\abstractnamefont}{\abscolnamefont}
      \renewcommand*{\abstracttextfont}{\abscoltextfont}
      \renewcommand*{\absnamepos}{flushleft}
      \setlength{\abstitleskip}{-2ex}
    \fi
  \fi}

\AtBeginDocument{\setlength{\absparindent}{\parindent}
                 \setlength{\absparsep}{\parskip}}

```

`@bstr@ctlist` The abstract's text is typeset within the `@bstr@ctlist` list environment.

```
\newenvironment{@bstr@ctlist}{%
  \list{}{%
          %%\topsep        \z@
          \partopsep     \z@
          \listparindent \absparindent
          \itemindent    \listparindent
          \leftmargin    \abs@leftindent
          \rightmargin   \absrightindent
          \parsep        \absparsep}%
  \item\relax}
  {\endlist}

```

`\put@bsintoc` This macro adds the abstract's title to the ToC. It does nothing if the abstract is being numbered.

```
\newcommand{\put@bsintoc}{%
  \ifadd@bstotoc
    \ifnumber@bs\else
      \phantomsection
      \addcontentsline{toc}{chapter}{\abstractname}
    \fi
  \fi}

```

`\num@bs` This macro generates a numbered abstract heading.

```
\newcommand{\num@bs}{\chapter{\abstractname}}
```

`abstract` At last we are in position to define the `abstract` environment. This follows very much along the lines of the standard class definitions, but with macros inserted at strategic points.
The twocolumn option and the default case. These use the same code as any style differences are embedded in the new macros.

```
\newenvironment{abstract}{%
  \setup@bstract
  \if@bsrunin\else
    \ifnumber@bs \num@bs \else
      \begin{\absnamepos}\abstractnamefont\abstractname\end\absnamepos%
      \vspace{\abstitleskip}%
    \fi
  \fi
  \put@bsintoc%
  \begin{@bstr@ctlist}\if@bsrunin\@bsrunintitle\fi\abstracttextfont}%
  {\par\end{@bstr@ctlist}}
```

`onecolabstract` An environment for typesetting a single column abstract, particularly as the optional argument to the `\twocolumn` command.

```
\newenvironment{onecolabstract}{%
  \begin{@twocolumnfalse}\begin{abstract}}{%
  \end{abstract}\end{@twocolumnfalse}}
```

`\thanks` `\@bs@thanks` We have to keep the contents of the `\thanks` commands as normally these are emptied by the `\maketitle` command. I do this by extending the definition of the `\thanks` (from `ltsect.dtx`) command, so that `\@bs@thanks` has a copy of the contents of `\@thanks`.

```
\addtoiargdef{\thanks}{}{%
  \protected@xdef\@bs@thanks{\@bs@thanks
    \protect\footnotetext[\the\c@footnote]{#1}}%
}
\let\@bs@thanks\@empty
```

`\saythanks` This macro typesets any `\thanks` commands *after* using `onecolabstract`.

```
\newcommand{\saythanks}{\begingroup
  \renewcommand{\thefootnote}{\fnsymbol{footnote}}\@bs@thanks
  \endgroup\global\let\@bs@thanks\@empty}
```

15 Verse

The class provides a more flexible `verse` environment than the standard classes. This is based on the verse package [Wil01j].

15.1 Environments

Before proceeding with the main, here are some macros for aspects of line numbering.

`\c@vslineno` `\c@poemline` `\c@modulo@vs` `\c@memfvsline`

We need counters for stanza and poem lines. The `memfvsline` counter is for adjusting the starting line for verse line numbers.

```
\newcounter{vslineno}
\newcounter{poemline}
\newcounter{modulo@vs}
\newcounter{memfvsline}
```

`\poemlines`

`\poemlines{`⟨*nth*⟩`}` specifies that every ⟨*nth*⟩ line of a poem is to be numbered.

```
%%%\newcommand{\poemlines}[1]{\linenumberfrequency{#1}%
%%%  \@memwarn{Use \string\linenumberfrequency\space
%%%                        instead of \string\poemlines}}
```

`\linenumberfont` `\vlvnumfont`

Set line numbering font(s).

```
\newcommand{\linenumberfont}[1]{\def\vlvnumfont{#1}}
%%% \linenumberfont{\small\rmfamily}
```

`\ifbvcountlines` `\bvcountlinestrue` `\bvcountlinesfalse`

Looking ahead, TRUE for (boxed) verbatim line numbers to be printed. Default is not to print them. Not that `\linenumberfrequency` twiddles with this.

```
\newif\ifbvcountlines%  TRUE to print line numbers of (boxed) verbatim lines
  \bvcountlinesfalse
```

`\linenumberfrequency` `\linemodnum`

Set numbering intervals (number modulo). Default is modulus 0.

```
\newcommand{\linenumberfrequency}[1]{%
  \ifnum #1< \@ne
    \def\linemodnum{0\relax}
    \bvcountlinesfalse
  \else
    \def\linemodnum{#1\relax}
    \bvcountlinestrue
  \fi}

%%%%\linenumberfrequency{0}
```

\setverselinenums `\setverselinenums{`⟨*firstline*⟩`}{`⟨*startnumsat*⟩`}` sets the first line number to ⟨*firstline*⟩ and the first line number to be printed is ⟨*startnumsat*⟩. Use within the `verse` environment before the first verse. Note that `firstline <= startnumsat < firstline + linenumberfrequency`

```
\newcommand*{\setverselinenums}[2]{%
  \c@poemline #1\relax \advance\c@poemline \m@ne
  \refstepcounter{poemline}%
  \ifnum\z@<\linemodnum%   we are printing line numbers
    \@tempcnta #2\relax
    \divide\@tempcnta\linemodnum
    \multiply\@tempcnta\linemodnum
    \c@memfvsline #2\relax
    \advance\c@memfvsline-\@tempcnta
  \fi}
```

\getthelinenumber `\getthelinenumber{`⟨*counter*⟩`}{`⟨*start*⟩`}` returns `\thecounter` if it is exactly divisible by `\linenumberfrequency`, provided this is not zero. ⟨*start*⟩ is the first number.

```
\newcommand{\getthelinenumber}[2]{%
  \ifnum\@ne>\linemodnum%  no line numbers
  \else
    \ifnum\@ne=\linemodnum% every line numbered
      \@nameuse{the#1}%
    \else
      \@tempcnta=\@nameuse{c@#1}%
      \advance\@tempcnta -\@nameuse{c@#2}%
      \divide\@tempcnta \linemodnum
      \multiply\@tempcnta \linemodnum
      \advance\@tempcnta \@nameuse{c@#2}%
      \ifnum\@tempcnta=\@nameuse{c@#1}\@nameuse{the#1}\fi
    \fi
  \fi}
```

\ifaltindent This should be set TRUE for indenting alternate lines.

```
\newif\ifaltindent
  \altindentfalse
```

\ifpattern This should be set TRUE for indenting lines according to a pattern.

```
\newif\ifpattern
  \patternfalse
```

\ifstarpattern This should be set TRUE for indenting lines according in a `patverse*` environment.

```
\newif\ifstarpattern
  \starpatternfalse
```

\vleftskip \vrightskip Skips to the left and right of a line of verse.

```
\newlength{\vleftskip}
  \setlength{\vleftskip}{3em}
\newlength{\vrightskip}
  \setlength{\vrightskip}{1em}
```

\stanzaskip Skip between stanzas.

```
\newlength{\stanzaskip}
  \setlength{\stanzaskip}{\onelineskip}
```

\flagverse \flagverse{⟨*flag*⟩} inserts ⟨*flag*⟩ at the left (of a line).

```
\newcommand{\flagverse}[1]{%
%%%  \hskip-\vleftskip\llap{#1}\hskip\vleftskip\ignorespaces}
  \hskip-\memRTLvleftskip\llap{#1}\hskip\memRTLvleftskip\ignorespaces}
```

\versewidth The length \versewidth is a convenience length for the user.

```
\newlength{\versewidth}
```

\vgap \vin \vindent The length \vgap is used as the basis for spacing. \vin makes a horizontal space of \vgap and \vindent is the indentation of wrapped lines.

```
\newlength{\vgap} \setlength{\vgap}{1.5em}
\newcommand{\vin}{\hspace*{\vgap}}
\newlength{\vindent} \setlength{\vindent}{2\vgap}
```

\vinphantom Macro to leave blank space coresponding to a string.

```
\newcommand{\vinphantom}[1]{\leavevmode\phantom{#1}}
```

\vleftofline Macro to insert something immediately to the left of the start of the line.

```
\newcommand*{\vleftofline}[1]{\leavevmode\llap{#1}}
```

\vleftmargin Length to adjust the default left margin within a verse environment.

```
\newdimen\vleftmargin
  \vleftmargin=\leftmargini
```

\verselinebreak Break a verse line by inserting \newline.

```
\newcommand{\verselinebreak}[1][\z@]{\newline\hspace*{#1}\ignorespaces}
```

\incr@vsline Increment the line counters.

```
\newcommand{\incr@vsline}{%
  \refstepcounter{poemline}%
  \stepcounter{vslineno}}
```

\@vsifbang Like the kernel \@ifstar except it looks for an exclamation mark!

```
\newcommand{\@vsifbang}[1]{\@ifnextchar !{\@firstoftwo{#1}}}
```

\@vsifgt Like the kernel \@ifstar except it looks for a > character.

```
\newcommand{\@vsifgt}[1]{\@ifnextchar >{\@firstoftwo{#1}}}

```

\verselinenumbersright Declarations for setting line numbers at the right (the default) or the left..
\verselinenumbersleft
\@vstypelinenum

```
\newcommand*{\verselinenumbersright}{\def\@vstypelinenum{\@vslnumright}}
\newcommand*{\verselinenumbersleft}{\def\@vstypelinenum{\@vslnumleft}}
\verselinenumbersright

```

\@vslnumright Internal code for right/left line numbers.
\@vslnumleft

```
\newcommand*{\@vslnumright}{%
  \hfill\rlap{%\kern\vrightskip\kern\rightmargin%
              \kern\memRTLvrightskip\kern\rightmargin%
              \vlvnumfont\getthelinenumber{poemline}{memfvsline}}}
\newcommand*{\@vslnumleft}{%
  \hfill\rlap{%\kern-\textwidth\kern-\vrightskip%
              \kern-\textwidth\kern-\memRTLvrightskip%
              \vlvnumfont\getthelinenumber{poemline}{memfvsline}}}
```

\@vscentercr This puts the poem line number in the margin, increments the line numbers, and then deals with the options. It is based on the kernel \@centercr. This has to handle various forms of the \\ command: \\, *, \\!, and \\>, together with an optional length argument.

```
\newcommand{\@vscentercr}{%
  \ifhmode \unskip\else \@nolnerr\fi
  \@vstypelinenum%
```

For > call \verselinebreak to process it.

```
  \@vsifgt{\verselinebreak}{%
    \incr@vsline
```

If the call is *... call \@vsxcentercr to handle the *.... If the call is \\!, do nothing. If the call is \\![...], call \@vsicentercr to handle the [...]. Otherwise, call \@vsxcentercr.

```
    \par\@ifstar{\nobreak\@vsxcentercr}{%
      \@vsifbang{\@ifnextchar[ {\@vsicentercr}{}}{\@vsxcentercr}}}}
```

\@vsxcentercr Processes *, and either calls cs@vsicentercr to handle a [length], or \start@vsline.

```
\newcommand{\@vsxcentercr}{\addvspace{-\parskip}%
  \@ifnextchar[ {\@vsicentercr}{\start@vsline}}
```

\@vsicentercr Processes (\\...)[length] and then calls \start@vsline.

```
\def\@vsicentercr[#1]{\vskip #1\ignorespaces \start@vsline}
```

`\start@vsline` This is called at the start of every verse line except the first.

```
\newcommand{\start@vsline}{%
  \ifaltindent\ifodd\c@vslineno\else\vin\fi\fi%
  \ifpattern\get@vsindent\fi%
  \ifstarpattern\getstar@vsindent\fi}
```

`\theHpoemline` For the hyperref package need a way of distinguishing lines of a poem. See the thread *PDFTEX/Hyperef hates memoir verse environment?* on CTT October 2002.

```
\newcounter{verse}
\setcounter{verse}{0}
\newcommand{\theHpoemline}{\theverse.\thepoemline}
```

`verse` The extended `verse` environment. It sets the verse line counter, then defines the particular list environment adjusting the margins to center according to the length parameter. If the length parameter is at least the `\linewidth` then the 'centering' defaults to the original `verse` layout.

```
\newenvironment{verse}[1][\linewidth]{%
  \refstepcounter{verse}%
  \setcounter{poemline}{0}\refstepcounter{poemline}%
  \setcounter{vslineno}{1}%
  \let\\=\@vscentercr
  \list{}{\itemsep     \z@
          \itemindent  -\vindent
          \listparindent\itemindent
          \leftmargin   \vleftmargin
          \parsep       \stanzaskip
          \ifdim #1<\linewidth%   %% short line
            \rightmargin        \z@
            \leftmargin         \linewidth
            \advance\leftmargin -#1\relax
            \advance\leftmargin -0.5\leftmargin
            \advance\leftmargin \vindent
          \else
            \ifdim #1>\linewidth%  %% long line
              \rightmargin \z@
              \leftmargin  \vindent
            \else%                   %% default
              \rightmargin \leftmargin
              \advance\leftmargin \vindent
            \fi
          \fi}
  \item[]}{\endlist}
```

`altverse` This sets `\altindenttrue` (afterwards false) and initialises the line counter.

```
\newenvironment{altverse}%
```

```
{\starpatternfalse\patternfalse\altindenttrue
 \setcounter{vslineno}{1}}%
{\altindentfalse}
```

15.2 Patterns

The pattern code is based on the idea of converting a string of digits to an array of digits, and then being able to access the digit at a particular position in the array.

`\ifbounderror` A flag set TRUE if an attempt is made to access an array element outside the array limits.

```
\newif\ifbounderror
  \bounderrorfalse
```

`\ifinteger` A flag to indicate if a 'number' is an integer (TRUE) or not (FALSE).

```
\newif\ifinteger
```

`\c@chrsinstr` A counter for the number of characters.

```
\newcounter{chrsinstr}  % CHARactersINSTRing
```

`\newarray` `\newarray{`⟨*arrayname*⟩`}{`⟨*low*⟩`}{`⟨*high*⟩`}` defines an array called ⟨*arrayname*⟩ (no backslash e.g. `MyArray`), with low and high limts ⟨*low*⟩ and ⟨*high*⟩.

```
\newcommand{\newarray}[3]{%
  \@nameedef{#1-low}{#2}%
  \@nameedef{#1-high}{#3}%
  \ifnum #3<#2
    \@memerror{Limits for array #1 are in reverse order}{\@ehc}%
  \fi}
```

`\stringtoarray` `\stringtoarray{`⟨*arrayname*⟩`}{`⟨*string*⟩`}` puts each character from ⟨*string*⟩ sequentially into the ⟨*arrayname*⟩ array, starting with ⟨*low*⟩ = 1. It checks for an empty ⟨*string*⟩ and handles that specially.

```
\newcommand{\stringtoarray}[2]{%
  \def\@vsarrayname{#1}%
  \protected@edef\the@vsstring{#2}%
  \newarray{\@vsarrayname}{1}{1}%
  \@ifmtarg{#2}{%
    \c@chrsinstr \z@
    \@namedef{\@vsarrayname-1}{}
  }{%
    \c@chrsinstr \@ne
    \expandafter\@vsstringtoarray \the@vsstring\@vsend
  }}
```

`\@vsstringtoarray` Recursively adds characters to the array `\@vsarrayname`, incrementing the array's high limit.

```
\def\@vsstringtoarray #1#2\@vsend{%
  \@namedef{\@vsarrayname-\the\c@chrsinstr}{#1}
  \@nameedef{\@vsarrayname-high}{\the\c@chrsinstr}
  \@ifmtarg{#2}{%
    \def\@vsinext{}%
  }{%
    \advance\c@chrsinstr \@ne
    \def\@vsinext{%
      \@vsstringtoarray #2\@vsend%
    }%
  }
  \@vsinext}

```

`\setarrayelement` `\setarrayelement{`⟨*arrayname*⟩`}{`⟨*index*⟩`}{`⟨*value*⟩`}` sets the ⟨*arrayname*⟩ array's element at ⟨*index*⟩ to ⟨*value*⟩.

```
\newcommand{\setarrayelement}[3]{%
  \checkarrayindex{#1}{#2}%
  \@nameedef{#1-#2}{#3}}
```

`\getarrayelement` `\getarrayelement{`⟨*arrayname*⟩`}{`⟨*index*⟩`}{`⟨*value*⟩`}` defines the parameterless macro ⟨*value*⟩ (e.g., `\result`) to be the value at ⟨*index*⟩ in the ⟨*arrayname*⟩ array.

```
\newcommand{\getarrayelement}[3]{%
  \checkarrayindex{#1}{#2}%
  \protected@edef#3{\@nameuse{#1-#2}}}

```

`\checkarrayindex` `\checkarrayindex{`⟨*arrayname*⟩`}{`⟨*index*⟩`}` checks that the ⟨*index*⟩ of the ⟨*arrayname*⟩ array is valid. `\ifbounderror` is set FALSE if everything is OK, otherwise it is set TRUE.

```
\newcommand{\checkarrayindex}[2]{%
  \bounderrorfalse
  \expandafter\ifx\csname #1-low\endcsname\relax%
    \ifpattern\else
      \@memerror{No array called #1}{\@ehc}%
    \fi
    \bounderrortrue
  \fi
  \ifnum #2<\@nameuse{#1-low}\relax%
    \ifpattern\else
      \@memerror{Index #2 outside limits for array #1}{\@ehc}%
    \fi
    \bounderrortrue
  \fi
  \ifnum #2>\@nameuse{#1-high}\relax%
    \ifpattern\else
```

```
5774      \@memerror{Index #2 outside limits for array #1}{\@ehc}%
5775    \fi
5776    \bounderrortrue
5777  \fi}
5778
```

\arraytostring `\arraytostring{⟨arrayname⟩}{⟨string⟩}` converts the characters in the ⟨*arrayname*⟩ array into the parameterless macro ⟨*string*⟩ (e.g., `\MyString`).

```
5779 \newcommand{\arraytostring}[2]{%
5780   \def#2{}%
5781   \c@chrsinstr = \@nameuse{#1-low}%
5782   \@vsarraytostring{#1}{#2}}
5783
```

\@vsarraytostring `\@vsarraytostring{⟨arrayname⟩}{⟨string⟩}` recursively adds the (character) elements from ⟨*arrayname*⟩ to ⟨*string*⟩.

```
5784 \newcommand{\@vsarraytostring}[2]{%
5785   \ifnum\c@chrsinstr>\@nameuse{#1-high}\else
5786     \protected@edef#2{#2\@nameuse{#1-\thechrsinstr}}%
5787     \advance\c@chrsinstr\@ne%
5788     \@vsarraytostring{#1}{#2}%
5789   \fi}
5790
```

\checkifinteger `\checkifinteger{⟨num⟩}` checks if ⟨*num*⟩ is an integer. If it is, then `\ifinteger` is set TRUE, otherwise it is set FALSE. (Code based on Donald Arseneau's cite package).

```
5791 \newcommand{\checkifinteger}[1]{%
5792   \protected@edef\@vsa{#1}%
5793   \ifcat _\ifnum9<1\gobm{#1} _\else A\fi
5794     \integertrue%
5795   \else
5796     \integerfalse%
5797   \fi}
```

\gobm `\gobm{⟨num⟩}` is defined as ⟨*num*⟩. It could be defined as: `\newcommand{\gobm}[1]{\ifx-#1\expandafter\gobm\else#1\fi}` which would remove a leading minus sign (hyphen) from its argument (`gobm` = gobble minus sign). (Code from a posting to CTT by Donald Arseneau on 1997/07/21).

```
5798 \newcommand{\gobm}[1]{#1}
5799
```

\indentpattern `\indentpattern{⟨digits⟩}` stores ⟨*digits*⟩ for use as a verse indentation pattern.

```
5800 \newcommand{\indentpattern}[1]{%
5801   \stringtoarray{Array@vs}{#1}}
```

\get@vsindent \get@vsindent gets the indent pattern digit for the \thevslineno, then uses this to specify the line indentation as digit*\vgap.

```
\newcommand{\get@vsindent}{%
  \getarrayelement{Array@vs}{\number\value{vslineno}}{\@vspat}%
  \ifbounderror
    \arraytostring{Array@vs}{\@vsp@t}%
    \@memwarn{%
      Index `\thevslineno' for pattern `\@vsp@t' is out of bounds}%
    \def\@vspat{0}%
  \else
    \checkifinteger{\@vspat}%
    \ifinteger\else
      \arraytostring{Array@vs}{\@vsp@t}%
      \@memwarn{%
       `\@vspat' at index `\thevslineno' in pattern `\@vsp@t'
       is not a digit}%
      \def\@vspat{0}%
    \fi
  \fi
  \ifcase\@vspat\else\hspace*{\@vspat\vgap}\fi}
```

\getstar@vsindent \getstar@vsindent gets the indent pattern digit for the patverse* environment, then uses this to specify the line indentation as digit*\vgap. It lets the pattern repeat by resetting the vslineno counter.

```
\newcommand{\getstar@vsindent}{%
  \expandafter\ifx\csname Array@vs-high\endcsname\relax
    \@mememror{A pattern has not been specified}{\@ehc}
  \else
    \ifnum\c@vslineno>\@nameuse{Array@vs-high}%
      \setcounter{vslineno}{1}%
     \fi
     \get@vsindent
  \fi}
```

patverse The environment for setting verse line indents according to a pattern. It starts by setting \ifpattern TRUE, any other flags to FALSE, and initialises the line number. It ends by setting \ifpattern FALSE.

```
\newenvironment{patverse}%
  {\starpatternfalse\patterntrue\altindentfalse
   \setcounter{vslineno}{1}}%
  {\patternfalse}
```

patverse* The environment for setting verse line indents according to a repeating pattern. It starts by setting \ifstarpattern TRUE, any other flags to FALSE, and initialises the line number. It ends by setting \ifstarpattern FALSE.

```
\newenvironment{patverse*}%
  {\starpatterntrue\patternfalse\altindentfalse
```

```
5837    \setcounter{vslineno}{1}}%
5838   {\starpatternfalse}
5839
```

15.3 Titles

`\poemtitle` Typeset a poem title (like `\section` or other). The actual work is done by `\@vsptitle` (plain) or `\@vssptitle` (starred).

```
5840 \newcommand{\poemtitle}{\par%
5841   \secdef\@vsptitle\@vssptitle}
```

`\poemtoc` The kind of entry `\poemtitle` is to make in the ToC.

```
5842 \newcommand{\poemtoc}{section}
5843
```

`\mempoeminfo` `\mempoeminfo{title}`
`\mempoemstarinfo` `\mempoemstarinfo{title}`

```
5844 \newcommand{\mempoeminfo}[1]{}
5845 \newcommand{\mempoemstarinfo}[1]{}
5846
```

`\@vsptitle` Typeset a `\poemtitle`.

```
5847 \long\def\@vsptitle[#1]#2{%
5848   \phantomsection
5849   \addcontentsline{toc}{\poemtoc}{#1}%
5850   \M@gettitle{#1}%
5851   \mempoeminfo{#1}%
5852   \poemtitlemark{#1}%
5853   \@vstypeptitle{#2}%
5854   \@afterheading}
5855
```

`\@vssptitle` Typeset a `\poemtitle*`.

```
5856 \long\def\@vssptitle#1{%
5857   \M@gettitle{#1}%
5858   \mempoemstarinfo{#1}%
5859   \@vstypeptitle{#1}%
5860   \@afterheading}
5861
```

`\@vstypeptitle` This *really* typesets the title.

```
5862 \newcommand{\@vstypeptitle}[1]{%
5863   \vspace{\beforepoemtitleskip}%
5864   {\poemtitlefont #1\par}%
5865   \vspace{\afterpoemtitleskip}%
5866 }
5867
```

\poemtitlefont
\poemtitlemark
Sets the appearance to the title of a poem, and something for a header.

```
\newcommand{\poemtitlefont}{\normalfont\large\bfseries\centering}
\newcommand{\poemtitlemark}[1]{}
```

\beforepoemtitleskip
\afterpoemtitleskip
Lengths before and after a poem title, using approximately \section values.

```
\newlength{\beforepoemtitleskip}
  \setlength{\beforepoemtitleskip}{3.5ex \@plus 1ex \@minus .2ex}
\newlength{\afterpoemtitleskip}
  \setlength{\afterpoemtitleskip}{2.3ex \@plus.2ex}
```

\if@numptitle
\@numptitletrue
\@numptitlefalse
\NumberPoemTitle
\PlainPoemTitle
Following the \NumberPoemTitle declaration \PoemTitles are numbered. The \PlainPoemTitle declaration produces unnumbered titles.

```
\newif\if@numptitle
\newcommand*{\NumberPoemTitle}{\@numptitletrue}
\newcommand*{\PlainPoemTitle}{\@numptitlefalse}
\NumberPoemTitle
```

\c@poem
\thepoem
\theHpoem
Counter for poem titles.

```
\newcounter{poem}\setcounter{poem}{0}
  \renewcommand*{\thepoem}{\@arabic\c@poem}
\newcommand*{\theHpoem}{\arabic{poem}}
```

\poemtitlestarmark
\poemtitlepstyle
\poemtitlestarpstyle
\poemtitlemark{⟨*poemf@rhdr*⟩} is used for marks for a \PoemTitle and \poemtitlestarmark{⟨*poemf@hdr*⟩} is for marks for \PoemTitle*. The \poemtitlepstyle and \poemtitlestarpstyle macros are provided as hooks so that, for example, pagestyles can be set for the regular and starred versions of \PoemTitle. By default these macros do nothing.

```
\newcommand*{\poemtitlestarmark}[1]{}
\newcommand*{\poemtitlepstyle}{}
\newcommand*{\poemtitlestarpstyle}{}
```

\PoemTitle
The command to produce a article style chapter-like (numbered) poem title.

```
\newcommand\PoemTitle{%
  \par
  \@afterindentfalse
  \@ifstar{\@m@msPoemTitle}{\@m@mPoemTitle}}
```

\@m@mPoemTitle
\poemt@c
Intermediate and support macros for the extra optional argument to \PoemTitle. Have to do this long windedly otherwise dear old hyperref barfs.

```
\newcommand{\@m@mPoemTitle}[1][]{%
  \def\poemt@c{#1}% capture first optional arg
  \@ifnextchar[{\@PoemTitle}{\@PoemTitle[]}%
}
```

\memPoemTitleinfo \memPoemTitleinfo{num}{toc}{head}{full}
\memPoemTitlestarinfo \memPoemTitlestarinfo{short}{full}

```
\newcommand{\memPoemTitleinfo}[4]{}
\newcommand{\memPoemTitlestarinfo}[2]{}
```

\@PoemTitle \poemf@rtoc \poemf@rhdr

\@PoemTitle[⟨*tocmark*⟩]{⟨*title*⟩} typesets the title of a \PoemTitle. There is a number for \@numptitletrue and \@mainmattertrue.

```
\def\@PoemTitle[#1]#2{%
  \phantomsection
  \ifx\poemt@c\@empty % no optional args
    \def\poemf@rtoc{#2}%
    \def\poemf@rhdr{#2}%
  \else                  % at least one opt arg
    \let\poemf@rtoc\poemt@c
    \ifx\@empty#1\@empty
      \let\poemf@rhdr\poemt@c
    \else
      \def\poemf@rhdr{#1}%
    \fi
  \fi
  \m@m@Andfalse
  \if@numptitle
    \if@mainmatter
      \m@m@Andtrue
    \fi
  \fi
  \ifm@m@And
    \refstepcounter{poem}%
  \fi
```

Store the (short) title via \poemtitlemark and call \makePoemTitlehead to do the typesetting.

```
  \@makePoemTitlehead{#2}%
  \@afterheading
  \poemtitlemark{\poemf@rhdr}%
  \poemtitlepstyle
```

Add the title to the ToC.

```
  \ifm@m@And
    \addcontentsline{toc}{\poemtoc}{%
      \protect\numberline{\thepoem}\poemf@rtoc}%
    \memPoemTitleinfo{\thepoem}{\poemf@rtoc}{\poemf@rhdr}{#2}%
  \else
    \addcontentsline{toc}{\poemtoc}{\poemf@rtoc}%
    \memPoemTitleinfo{}{\poemf@rtoc}{\poemf@rhdr}{#2}%
  \fi
```

Add hook for title referencing.

```
  \ifheadnameref\M@gettitle{\poemf@rhdr}\else\M@gettitle{\poemf@rtoc}\fi}

```

\@makePoemTitlehead This *really* typesets a \PoemTitle. Leave some whitespace.

```
\def\@makePoemTitlehead#1{%
  \PoemTitleheadstart
  \parindent \z@ \normalfont
```

If there is a number, typeset it, otherwise call \printPoemTitlenonum.

```
  \ifm@m@And
    \printPoemTitlenum
    \afterPoemTitlenum
  \else
    \printPoemTitlenonum
  \fi
```

Typeset the title.

```
  \interlinepenalty\@M
  \printPoemTitletitle{#1}%
  \afterPoemTitle}}

```

\@PTchs@def@ult This sets up all the definitions used in \@makePoemTitlehead and \@makesPoemTitlehead.

\PoemTitleheadstart
\printPoemTitlenum
\afterPoemTitlenum
\printPoemTitlenonum
\printPoemTitletitle
\afterPoemTitle

```
\newcommand{\@PTchs@def@ult}{%
  \def\PoemTitleheadstart{\vspace{\beforePoemTitleskip}}
  \def\printPoemTitlenum{\PoemTitlenumfont \thepoem}
  \def\afterPoemTitlenum{\par\nobreak\vskip \midPoemTitleskip}
  \def\printPoemTitlenonum{}
  \def\printPoemTitletitle##1{\PoemTitlefont ##1}
  \def\afterPoemTitle{\par\nobreak\vskip \afterPoemTitleskip}}
\@PTchs@def@ult

```

\PoemTitlenumfont Fonts for setting the Poem Title number and title.
\PoemTitlefont

```
\newcommand*{\PoemTitlenumfont}{\normalfont\large\centering}
\newcommand*{\PoemTitlefont}{\normalfont\large\centering}
```

\beforePoemTitleskip Lengths separating the various parts of a Poem Title heading.
\midPoemTitleskip
\afterPoemTitleskip

```
\newlength{\beforePoemTitleskip}
  \setlength{\beforePoemTitleskip}{1\onelineskip}
\newlength{\midPoemTitleskip}
  \setlength{\midPoemTitleskip}{0pt}
\newlength{\afterPoemTitleskip}
  \setlength{\afterPoemTitleskip}{1\onelineskip}

```

\@m@msPoemTitle This deals with the optional argument for starred PoemTitles.

```
\newcommand{\@m@msPoemTitle}[2][\@empty]{%
  \@sPoemTitle{#2}%
  \ifx \@empty#1
```

```
    \def\poemf@rhdr{#2}%
  \else   % opt arg
    \def\poemf@rhdr{#1}%
  \fi
  \poemtitlestarmark{\poemf@rhdr}%
  \poemtitlestarpstyle
  \memPoemTitlestarinfo{\poemf@rhdr}{#2}}
```

`\@sPoemTitle` `\@sPoemTitle{`⟨*long*⟩`}` typesets the title of a `\PoemTitle*`. It is easier than the `\@PoemTitle` as there is no number or ToC entry to worry about.

```
\newcommand{\@sPoemTitle}[1]{%
  \@makesPoemTitlehead{#1}%
  \@afterheading
  \M@gettitle{#1}}
```

`\@makesPoemTitlehead` This *really* typesets a `\PoemTitle*`, and is similar to `\@makePoemTitlehead`.

```
\def\@makesPoemTitlehead#1{{%
  \PoemTitleheadstart
  \parindent \z@ \normalfont
  \printPoemTitlenonum
  \interlinepenalty\@M
  \printPoemTitletitle{#1}
  \afterPoemTitle}}
```

16 Setting parameters for existing environments

16.1 Array and tabular

`\arraycolsep` The columns in an array environment are separated by 2`\arraycolsep`.

```
\setlength\arraycolsep{5\p@}
```

`\tabcolsep` The columns in an tabular environment are separated by 2`\tabcolsep`.

```
\setlength\tabcolsep{6\p@}
```

`\arrayrulewidth` The width of rules in the array and tabular environments is given by `\arrayrulewidth`.

```
\setlength\arrayrulewidth{.4\p@}
```

`\doublerulesep` The space between adjacent rules in the array and tabular environments is given by `\doublerulesep`.

```
\setlength\doublerulesep{2\p@}
```

16.2 Tabbing

\tabbingsep This controls the space that the \' command puts in. (See LaTeX manual for an explanation.)

```
\setlength\tabbingsep{\labelsep}
```

16.3 Minipage

\@minipagerestore The macro \@minipagerestore is called upon entry to a minipage environment to set up things that are to be handled differently inside a minipage environment. In the standard styles it does nothing, as was the case originally for memoir. Memoir extends it to support verbatim footnotes (see later) in minipages and also for abnormal parskips (see later).

```
\newcommand{\@minipagerestore}{%
```

For \verbfootnote to work in a minipage we have to use \@verbmpfootnotetext instead of \@verbfootnotetext.

```
  \let\@verbfootnotetext\@verbmpfootnotetext
```

The next is for enabling the extended footnotes in a minipage.

```
  \m@mdoextrafeetmini
```

And this is for abnormal parskips.

```
  \ifm@mnzpskip \parskip=\m@mabparskip\fi}
```

\@mpfootins Minipages have their own footnotes; \skip\@mpfootins plays the same rôle for footnotes in a minipage as \skip\footins does for ordinary footnotes.

```
\skip\@mpfootins = \skip\footins
```

16.4 Framed boxes

\fboxsep The space left by \fbox and \framebox between the box and the text in it.

\fboxrule The width of the rules in the box made by \fbox and \framebox.

```
\setlength\fboxsep{3\p@}
\setlength\fboxrule{.4\p@}
```

16.5 Equation and eqnarray

\theequation The equation counter will be reset at beginning of a new chapter and the equation number will be prefixed by the chapter number.
This code must follow the \chapter definition, or more exactly the definition of the chapter counter.

```
\@addtoreset{equation}{chapter}
\renewcommand{\theequation}{%
  \ifnum \c@chapter>\z@ \thechapter.\fi \@arabic\c@equation}
```

`\jot` `\jot` is the extra space added between lines of an eqnarray environment. The default value is used.

```
% \setlength\jot{3pt}
```

`\@eqnnum` The macro `\@eqnnum` defines how equation numbers are to appear in equations. Again the default is used.

```
% \def\@eqnnum{(\theequation)}

```

17 Array and tabular

17.1 Array

The description and code are essentially copied from the array package [MC98].

```

%%%%%%%%%% Array package code %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% With acknowledgements to %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Frank Mittelbach & David Carlisle %%%%%%%%%%%%%%%%%%%%

```

17.1.1 The construction of the preamble

PW: The original array package has redundant code (caused by a quick mashing of an early array and the newarray packages). I have tried to remove the redundancies and overlaps.
It is obvious that those environments will consist mainly of an `\halign`, because TEX typesets tables using this primitive. That is why we will now take a look at the algorithm which determines a preamble for a `\halign` starting with a given user preamble using the options mentioned above.
The most interesting macros of this implementation are without doubt those which are responsible for the construction of the preamble for the `\halign`. The underlying algorithm was developed by LAMPORT (resp. KNUTH, see texhax V87#??), and it has been extended and improved.
The user preamble will be read token by token. A token is a single character like `c` or a block enclosed in `{...}`. For example the preamble of `\begin{tabular}{lc||c@{\hspace{1cm}}}` consists of the token `l`, `c`, `|`, `|`, `@` and `\hspace{1cm}`. The currently used token and the one, used before, are needed to decide on how the construction of the preamble has to be continued. In the example mentioned above the `l` causes the preamble to begin with `\hskip\tabcolsep`. Furthermore `# \hfil` would be appended to define a flush left column. The next token is a `c`. Because it was preceded by an `l` it generates a new column. This is done with `\hskip \tabcolsep & \hskip \tabcolsep`. The column which is to be centered will be appended with `\hfil # \hfil`. The token `|` would then add a space of `\hskip \tabcolsep` and a vertical line because the last tokens was a `c`. The following token `|` would only add a space `\hskip \doublerulesep` because

it was preceded by the token |. We will not discuss our example further but rather take a look at the general case of constructing preambles.
The example shows that the desired preamble for the \halign can be constructed as soon as the action of all combinations of the preamble tokens are specified. There are 18 such tokens so we have $19 \cdot 18 = 342$ combinations if we count the beginning of the preamble as a special token. Fortunately, there are many combinations which generate the same spaces, so we can define token classes. We will identify a token within a class with a number, so we can insert the formatting (for example of a column). Table 5 lists all token classes and their corresponding numbers.

token	\@chclass	\@chnum	token	\@chclass	\@chnum
c	0	0	Start	4	--
l	0	1	@-arg	5	--
r	0	2	!	6	--
p-arg	0	3	@	7	--
t-arg	0	4	<	8	--
b-arg	0	5	>	9	--
|	1	0	p	10	3
!-arg	1	1	t	10	4
<-arg	2	--	b	10	5
>-arg	3	--			

Table 5: Classes of preamble tokens

\@chclass
\@chnum
\@lastchclass

The class and the number of the current token are saved in the count registers \@chclass and \@chnum, while the class of the previous token is stored in the count register \lastchclass. All of the mentioned registers are already allocated in latex.tex, which is the reason why the following three lines of code are commented out. Later throughout the text I will not mention it again explicitely whenever I use a % sign. These parts are already defined in latex.tex.

```
% \newcount \@chclass
% \newcount \@chnum
% \newcount \@lastchclass
```

\@addtopreamble

We will save the already constructed preamble for the \halign in the global macro \@preamble. This will then be enlarged with the command \@addtopreamble.

```
\def\@addtopreamble#1{\xdef\@preamble{\@preamble #1}}
```

\@testpach

With the help of \@lastchclass we can now define a macro which determines the class and the number of a given preamble token and assigns them to the registers \@chclass and \@chnum.

```
\def\@testpach{\@chclass
```

First we deal with the cases in which the token (#1) is the argument of !, @, < or >. We can see this from the value of \@lastchclass:

```
\ifnum \@lastchclass=6 \@ne \@chnum \@ne \else
 \ifnum \@lastchclass=7 5 \else
  \ifnum \@lastchclass=8 \tw@ \else
   \ifnum \@lastchclass=9 \thr@@
```

Otherwise we will assume that the token belongs to the class 0 and assign the corresponding number to \@chnum if our assumption is correct.

```
  \else \z@
```

If the last token was a p, m or a b, \@chnum already has the right value. This is the reason for the somewhat curious choice of the token numbers in class 10.

```
  \ifnum \@lastchclass = 10 \else
```

Otherwise we will check if \@nextchar is either a c, l or an r. Some applications change the catcodes of certain characters like "@" in amstex.sty. As a result the tests below would fail since they assume non-active character tokens. Therefore we evaluate \@nextchar once thereby turning the first token of its replacement text into a char. At this point here this should have been the only char present in \@nextchar which put into via a \def.

```
  \edef\@nextchar{\expandafter\string\@nextchar}%
  \@chnum
  \if \@nextchar c\z@ \else
   \if \@nextchar l\@ne \else
    \if \@nextchar r\tw@ \else
```

If it is a different token, we know that the class was not 0. We assign the value 0 to \@chnum because this value is needed for the |–token. Now we must check the remaining classes. Note that the value of \@chnum is insignificant here for most classes.

```
  \z@ \@chclass
  \if\@nextchar |\@ne \else
   \if \@nextchar !6 \else
    \if \@nextchar @7 \else
     \if \@nextchar <8 \else
      \if \@nextchar >9 \else
```

The remaining permitted tokens are p, m and b (class 10).

```
 10
 \@chnum
 \if \@nextchar m\thr@@\else
  \if \@nextchar p4 \else
   \if \@nextchar b5 \else
```

Now the only remaining possibility is a forbidden token, so we choose class 0 and number 0 and give an error message. Then we finish the macro by closing all \if's.

```
  \z@ \@chclass \z@ \@preamerr \z@ \fi \fi \fi \fi
  \fi \fi  \fi  \fi  \fi  \fi  \fi \fi \fi \fi \fi \fi}
```

The preamble will be enlarged with the help of \xdef, but the arguments of >, <, ! and @ are not supposed to be expanded during the construction (we want an implementation that doesn't need a \protect). So we have to find a way to inhibit the expansion of those arguments.
We will solve this problem with token registers. We need one register for every ! and @, while we need two for every c, l, r, m, p or b. This limits the number of columns of a table because there are only 256 token registers. But then, who needs tables with more than 100 columns?
So how do we proceed? Let us assume that we had !{foo} in the user preamble and say we saved foo in token register 5. Then we call \@addtopreamble{\the@toks5} where \the@toks is defined in a way that it does not expand (for example it could be equivalent to \relax). Every following call of \@addtopreamble leaves \the@toks5 unchanged in \@preamble. If the construction of the preamble is completed we change the definition of \the@toks to \the\toks and expand \@preamble for the last time. During this process all parts of the form \the@toks⟨*Number*⟩ will be substituted by the contents of the respective token registers.
As we can see from this informal discussion the construction of the preamble has to take place within a group, so that the token registers we use will be freed later on. For that reason we keep all assignments to \@preamble global; therefore the replacement text of this macro will remain the same after we leave the group.

\count@ We further need a count register to remember which token register is to be used next. This will be initialized with -1 if we want to begin with the token register 0. We use the PLAIN TEX scratch register \count@ because everything takes place locally. All we have to do is insert \the@toks \the \count@ into the preamble. \the@toks will remain unchanged and \the\count@ expands into the saved number.

\prepnext@tok The macro \prepnext@tok is in charge of preparing the next token register. For that purpose we increase \count@ by 1:

```
\def\prepnext@tok{\advance \count@ \@ne
```

Then we locally delete any contents the token register might have.

```
  \toks\count@{}}
```

\save@decl During the construction of the preamble the current token is always saved in the macro \@nextchar (see the definition of \@mkpream). The macro \save@decl saves it into the next free token register, i.e. in \toks\count@. We do not assume that the token register is free, we add the new declarations to the front of the register. This is to allow user preambles of the form, >{foo}>{bar}... Users are not encouraged to enter such expressions directly, but they may result from the rewriting of \newcolumntype's.

```
\def\save@decl{\toks \count@ = \expandafter\expandafter\expandafter
                  {\expandafter\@nextchar\the\toks\count@}}
```

How does the situation look like, if we want to add another column to the preamble, i.e. if we have found a `c`, `l`, `r`, `p`, `m` or `b` in the user preamble? In this case we have the problem of the `token` register from `>{..}` and `<{..}` having to be inserted at this moment because formating instructions like `\hfil` have to be set around them. On the other hand it is not known yet, if any `<{..}` instruction will appear in the user preamble at all.
We solve this problem by adding two `token` registers at a time. This explains, why we have freed the `token` registers in `\prepnext@tok`.

`\insert@column` `\@sharp` We now define the macro `\insert@column` which will do this work for us.

```
\def\insert@column{%
```

Here, we assume that the `count` register `\@tempcnta` has saved the value `\count@`-1.

```
  \the@toks \the \@tempcnta
```

Next follows the `#` sign which specifies the place where the text of the column shall be inserted. To avoid errors during the expansions in `\@addtopreamble` we hide this sign in the command `\@sharp` which is temporarily occupied with `\relax` during the build-up of the preamble. To remove unwanted spaces before and after the column text, we set an `\ignorespaces` in front and a `\unskip` afterwards.

```
  \ignorespaces \@sharp \unskip
```

Then the second `token` register follows whose number should be saved in `\count@`. We make sure that there will be no further expansion after reading the number, by finishing with `\relax`. The case above is not critical since it is ended by `\ignorespaces`.

```
  \the@toks \the \count@ \relax}
```

`\m@mold@addamp` `\m@m@addamp` `\@addamp` In the preamble a `&` has to be inserted between any two columns; before the first column there should not be a `&`. As the user preamble may start with a `|` we have to remember somehow if we have already inserted a `#` (i.e. a column). This is done with the boolean variable `\if@firstamp` that we test in `\@addamp`, the macro that inserts the `&`.
PW: Later on, for horizontal lines in 'continuous' tabulars, I need to know how many columns there are in a tabular. I need a modified kernel's `\@addamp` to do this and use the kernel's `\@curtab` (a counter used in tabbing) to store the number of columns.
`\m@mold@addamp` stores the kernel's definition of `\@addamp`, and `\m@m@addamp` is the revised definition.

```
%    \newif \@iffirstamp
\let\m@mold@addamp\@addamp
\newcommand*{\m@m@addamp}{%
  \if@firstamp
    \@firstampfalse
    \global\@curtab\@ne
  \else
```

```
    \@addtopreamble{&}
    \global\advance\@curtab\@ne
  \fi}
\let\@addamp\m@m@addamp
```

\@acol
\@acolampacol
\col@sep

We will now define some abbreviations for the extensions, appearing most often in the preamble build-up. Here \col@sep is a dimen register which is set equivalent to \arraycolsep in an array–environment, otherwise it is set equivalent to \tabcolsep.

```
\newdimen\col@sep
\def\@acol{\@addtopreamble{\hskip\col@sep}}
%    \def\@acolampacol{\@acol\@addamp\@acol}
```

\@mkpream
\the@toks

Now we can define the macro which builds up the preamble for the \halign. First we initialize \@preamble, \@lastchclass and the boolean variable \if@firstamp.

```
\def\@mkpream#1{\gdef\@preamble{}\@lastchclass 4 \@firstamptrue
```

During the build-up of the preamble we cannot directly use the # sign; this would lead to an error message in the next \@addtopreamble call. Instead, we use the command \@sharp at places where later a # will be. This command is at first given the meaning \relax; therefore it will not be expanded when the preamble is extended. In the macro \@array, shortly before the \halign is carried out, \@sharp is given its final meaning.
In a similar way, we deal with the commands \@startpbox and \@endpbox, although the reason is different here: these macros expand in many tokens which would delay the build-up of the preamble.

```
  \let\@sharp\relax \let\@startpbox\relax \let\@endpbox\relax
```

Now we remove possible *-forms and user-defined column specifiers in the user preamble by repeatedly executing the list \NC@list until the re-writes have no more effect. The expanded preamble will then be in the token register \@temptokena. Actually we need to know at this point that this is not \toks0.

```
  \@temptokena{#1}\@tempswatrue
  \@whilesw\if@tempswa\fi{\@tempswafalse\the\NC@list}%
```

Afterwards we initialize all registers and macros, that we need for the build-up of the preamble. Since we want to start with the token register 0, \count@ has to contain the value -1.

```
  \count@\m@ne
  \let\the@toks\relax
```

Then we call up \prepnext@tok in order to prepare the token register 0 for use.

```
  \prepnext@tok
```

Having expanded all tokens defined using \newcolumntype (including *), we evaluate the remaining tokens, which are saved in \@temptokena. We use the LaTeX–macro \@tfor to inspect each token in turn. The strange appearing

construction with `\expandafter` is based on the fact that we have to put the replacement text of `\@tempa` and not the macro `\@tempa` to this LaTeX–macro.

```
6081    \expandafter \@tfor \expandafter \@nextchar
6082     \expandafter :\expandafter =\the\@temptokena \do
```

The body of this loop (the group after the `\do`) is executed for one `token` at a time, whereas the current `token` is saved in `\@nextchar`. At first we evaluate the current `token` with the already defined macro `\@testpach`, i.e. we assign to `\@chclass` the character class and to `\@chnum` the character number of this `token`.

```
6083    {\@testpach
```

Then we branch out depending on the value of `\@chclass` into different macros that extend the preamble respectively.

```
6084    \ifcase \@chclass \@classz \or \@classi \or \@classii
6085      \or \save@decl \or \or \@classv \or \@classvi
6086      \or \@classvii \or \@classviii
```

Class 9 is equivalent to class 10.

```
6087      \or \@classx
6088      \or \@classx \fi
```

Two cases deserve our special attention: Since the current `token` cannot have the character class 4 (start) we have skipped this possibility. If the character class is 3, only the content of `\@nextchar` has to be saved into the current `token` register; therefore we call up `\save@decl` directly and save a macro name. After the preamble has been extended we assign the value of `\@chclass` to the counter `\@lastchclass` to assure that this information will be available during the next run of the loop.

```
6089    \@lastchclass\@chclass}%
```

After the loop has been finished space must still be added to the created preamble, depending on the last `token`. Depending on the value of `\@lastchclass` we perform the necessary operations.

```
6090    \ifcase\@lastchclass
```

If the last class equals 0 we add a `\hskip \col@sep`.

```
6091    \@acol \or
```

If it equals 1 we do not add any additional space so that the horizontal lines do not exceed the vertical ones.

```
6092    \or
```

Class 2 is treated like class 0 because a `<{...}` can only directly follow after class 0.

```
6093    \@acol \or
```

Most of the other possibilities can only appear if the user preamble was defective. Class 3 is not allowed since after a `>{..}` there must always follow a `c`, `l`, `r`, `p`,`m` or `b`. We report an error and ignore the declaration given by `{..}`.

```
6094    \@preamerr \thr@@ \or
```

If \@lastchclass is 4 the user preamble has been empty. To continue, we insert a # in the preamble.

```
6095    \@preamerr \tw@ \@addtopreamble\@sharp \or
```

Class 5 is allowed again. In this case (the user preamble ends with @{..}) we need not do anything.

```
6096    \or
```

Any other case means that the arguments to @, !, <, >, p, m or b have been forgotten. So we report an error and ignore the last token.

```
6097    \else  \@preamerr \@ne \fi
```

Now that the build-up of the preamble is almost finished we can insert the token registers and therefore redefine \the@toks. The actual insertion, though, is performed later.

```
6098    \def\the@toks{\the\toks}}
```

The preamble is extended by the macros \@classz to \@classx which are called by \@mkpream depending on \@lastchclass (i.e. the character class of the last token).

\@classx First we define \@classx because of its important rôle. When it is called we find that the current token is p, m or b. That means that a new column has to start.

```
6099 \def\@classx{%
```

Depending on the value of \@lastchclass different actions must take place:

```
6100    \ifcase \@lastchclass
```

If the last character class was 0 we separate the columns by \hskip\col@sep followed by & and another \hskip\col@sep.

```
6101    \@acolampacol \or
```

If the last class was class 1 — that means that a vertical line was drawn, — before this line a \hskip\col@sep was inserted. Therefore there has to be only a & followed by \hskip\col@sep. But this & may be inserted only if this is not the first column. This process is controlled by \if@firstamp in the macro \addamp.

```
6102    \@addamp \@acol \or
```

Class 2 is treated like class 0 because <{...} can only follow after class 0.

```
6103    \@acolampacol \or
```

Class 3 requires no actions because all things necessary have been done by the preamble token >.

```
6104    \or
```

Class 4 means that we are at the beginning of the preamble. Therefore we start the preamble with \hskip\col@sep and then call \@firstampfalse. This makes sure that a later \@addamp inserts the character & into the preamble.

```
6105    \@acol \@firstampfalse \or
```

For class 5 tokens only the character & is inserted as a column separator. Therefore we call \@addamp.

```
6106    \@addamp
```

Other cases are impossible. For an example `\@lastchclass = 6` —a s it might appear in a preamble of the form `...!p...` — `p` would have been taken as an argument of `!` by `\@testpach`.

```
\fi}
```

`\@classz` If the character class of the last token is 0 we have `c`, `l`, `r` or an argument of `m`, `b` or `p`. In the first three cases the preamble must be extended the same way as if we had class 10. The remaining two cases do not require any action because the space needed was generated by the last token (i.e. `m`, `b` or `p`). Since `\@lastchclass` has the value 10 at this point nothing happens when `\@classx` is called. So the macro `\@chlassz` may start like this:

```
\def\@classz{\@classx
```

According to the definition of `\insert@column` we must store the number of the token register in which a preceding `>{..}` might have stored its argument into `\@tempcnta`.

```
    \@tempcnta \count@
```

To have `\count@ = \@tmpcnta + 1` we prepare the next token register.

```
    \prepnext@tok
```

Now the preamble must be extended with the column whose format can be determinated by `\@chnum`.

```
    \@addtopreamble{\ifcase \@chnum
```

If `\@chnum` has the value 0 a centered column has to be generated. So we begin with stretchable space.

```
        \hfil
```

The command `\d@llarbegin` follows expanding into `\begingroup` (in the tabular–environment) or into `$`. Doing this (provided an appropriate setting of `\d@llarbegin`) we achieve that the contents of the columns of an array–environment are set in math mode while those of a tabular–environment are set in LR mode.

```
        \d@llarbegin
```

Now we insert the contents of the two token registers and the symbol for the column entry (i.e. `#` or more precise `\@sharp`) using `\insert@column`.

```
        \insert@column
```

We end this case with `\d@llarend` and `\hfil` where `\d@llarend` again is either `$` or `\endgroup`.

```
        \d@llarend \hfil \or
```

The templates for `l` and `r` (i.e. `\@chnum` 1 or 2) are generated the same way. Since one `\hfil` is missing the text is moved to the relevant side. The `\kern\z@` is needed in case of an empty column entry. Otherwise the `\unskip` in `\insert@column` removes the `\hfil`. Changed to `\hskip1sp` so that it interacts better with `\@bsphack`.

```
        \hskip1sp\d@llarbegin \insert@column \d@llarend \hfil \or
        \hfil\hskip1sp\d@llarbegin \insert@column \d@llarend \or
```

The templates for p, m and b mainly consist of a box. In case of m it is generated by \vcenter. This command is allowed only in math mode. Therefore we start with a $.

```
$\vcenter%$
```

The part of the templates which is the same in all three cases (p, m and b) is built by the macros \@startpbox and \@endpbox. \@startpbox has an argument: the width of the column which is stored in the current token (i.e. \@nextchar). Between these two macros we find the well known \insert@column.

```
\@startpbox{\@nextchar}\insert@column \@endpbox $\or%$
```

The templates for p and b are generated in the same way though we do not need the $ characters because we use \vtop or \vbox.

```
\vtop \@startpbox{\@nextchar}\insert@column \@endpbox \or
\vbox \@startpbox{\@nextchar}\insert@column \@endpbox
```

Other values for \@chnum are impossible. Therefore we end the arguments to \@addtopreamble and \ifcase. Before we come to the end of \@classz we have to prepare the next token register.

```
 \fi}\prepnext@tok}

```

\@classix Class 9 (>–token) prevented repeated > declarations for the same column. This restriction has been eased, making class 9 equivalent to class 10.

```
\let\@classix\relax

```

\@classviii If the current token is a < the last character class must be 0, or 2 (as repeated < expressions are allowed). In this case it is not necessary to extend the preamble. Otherwise we output an error message, set \@chclass to 6 and call \@classvi. By doing this we achieve that < is treated like !.

```
\def\@classviii{\ifnum \@lastchclass >\z@\ifnum\@lastchclass=\tw@\else
      \@preamerr 4\@chclass 6 \@classvi \fi\fi}

```

\@arrayrule There is only one incompatibility with the original definition: the definition of \@arrayrule. In the original a line without width (e.g., the space between cc and c|c is equal) is created by multiple insertions of \hskip .5\arrayrulewidth. We only insert a vertical line into the preamble. This is done to prevent problems with TEX's main memory when generating tables with many vertical lines in them (especially in the case of floats).

```
\def\@arrayrule{\@addtopreamble \vline}

```

\@classvii As a consequence it follows that in case of class 7 (@ token) the preamble need not to be extended. In the original definition \@lastchclass = 1 is treated by inserting \hskip .5\arrayrulewidth. We only check if the last token was of class 3 which is forbidden.

```
\def\@classvii{\ifnum \@lastchclass = \thr@@
```

If this is true we output an error message and ignore the declarations stored by the last `>{...}`, because these are overwritten by the argument of `@`.

```
\@preamerr \thr@@ \fi}
```

`\@classvi` If the current token is a regular `!` and the last class was 0 or 2 we extend the preamble with `\hskip\col@sep`. If the last token was of class 1 (for instance |) we extend with `\hskip \doublerulesep` because the construction `!{...}` has to be treated like |.

```
\def\@classvi{\ifcase \@lastchclass
      \@acol \or
      \@addtopreamble{\hskip \doublerulesep}\or
      \@acol \or
```

Now `\@preamerr...` should follow because a user preamble of the form `..>{..}!.` is not allowed. To save memory we call `\@classvii` instead which also does what we want.

```
      \@classvii
```

If `\@lastchclass` is 4 or 5 nothing has to be done. Class 6 to 10 are not possible. So we finish the macro.

```
      \fi}
```

`\@classii` `\@classiii` In the case of character classes 2 and 3 (i.e. the argument of < or >) we only have to store the current token (`\@nextchar`) into the corresponding token register since the preparation and insertion of these registers are done by the macro `\@classz`. This is equivalent to calling `\save@decl` in the case of class 3. To save command identifiers we do this call up in the macro `\@mkpream`.
Class 2 exhibits a more complicated situation: the token registers have already been inserted by `\@classz`. So the value of `\count@` is too high by one. Therefore we decrease `\count@` by 1.

```
\def\@classii{\advance \count@ \m@ne
```

Next we store the current token into the correct token register by calling `\save@decl` and then increase the value of `\count@` again. At this point we can save memory once more (at the cost of time) if we use the macro `\prepnext@tok`.

```
   \save@decl\prepnext@tok}
```

`\@classv` Class 5 is `@`-expressions (and is also called by class 1) We do not expand the `@`-expression, but instead explicitly replace an `\extracolsep` command by an assignment to `\tabskip` by a method similar to the `\newcolumntype` system described later.

```
\def\@classv{\save@decl
   \expandafter\NC@ecs\@nextchar\extracolsep{}\extracolsep\@@@
   \@addtopreamble{\d@llarbegin\the@toks\the\count@\relax\d@llarend}%
   \prepnext@tok}
```

\NC@ecs Rewrite the first occurrence of \extracolsep{1in} to \tabskip1in\relax. As a side effect discard any tokens after a second \extracolsep, there is no point in the user entering two of these commands anyway, so this is not really a restriction.

```
\def\NC@ecs#1\extracolsep#2#3\extracolsep#4\@@@{\def\@tempa{#2}%
  \ifx\@tempa\@empty\else\toks\count@={#1\tabskip#2\relax#3}\fi}
```

\@classi In the case of class 0 we were able to generate the necessary space between columns by using the macro \@classx. Analogously the macro \@classvi can be used for class 1.

```
\def\@classi{\@classvi
```

Depending on \@chnum a vertical line

```
   \ifcase \@chnum \@arrayrule \or
```

or (in case of !{...}) the current token — stored in \@nextchar — has to be inserted into the preamble. This corresponds to calling \@classv.

```
      \@classv \fi}
```

\@startpbox In \@classz the macro \@startpbox is used. The width of the parbox is passed as an argument. \vcenter, \vtop or \vbox are already in the preamble. So we start with the braces for the wanted box.

```
\def\@startpbox#1{\bgroup
```

The argument is the width of the box. This information has to be assigned to \hsize. Then we assain default values to several parameters used in a parbox.

```
   \setlength\hsize{#1}\@arrayparboxrestore
```

Our main problem is to obtain the same distance between succeeding lines of the parbox. We have to remember that the distance between two parboxes should be defined by \@arstrut. That means that it can be greater than the distance in a parbox. Therefore it is not enough to set a \@arstrut at the beginning and at the end of the parbox. This would dimension the distance between first and second line and the distance between the two last lines of the parbox wrongly. To prevent this we set an invisible rule of height \@arstrutbox at the beginning of the parbox. This has no effect on the depth of the first line. At the end of the parbox we set analogously another invisible rule which only affects the depth of the last line. It is necessary to wait inserting this strut until the paragraph actually starts to allow for things like \parindent changes via >{...}.

```
   \everypar{%
      \vrule \@height \ht\@arstrutbox \@width \z@
      \everypar{}}%
   }
```

\@endpbox If there are any declarations defined by >{...} and <{...} they now follow in the macro \@classz — the contents of the column in between. So the macro

\@endpbox must insert the specialstrut mentioned earlier and then close the group opened by \@startpbox.

```
\def\@endpbox{\@finalstrut\@arstrutbox \egroup\hfil}
```

17.1.2 Building and calling \halign

\@array After we have discussed the macros needed for the evaluation of the user preamble we can define the macro \@array which uses these macros to create a \halign. It has two arguments. The first one is a position argument which can be t, b or c; the second one describes the wanted preamble, e.g. it has the form |c|c|c|.

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

First we define a strut whose size basically corresponds to a normal strut multiplied by the factor \arraystretch. This strut is then inserted into every row and enforces a minimal distance between two rows. Nevertheless, when using horizontal lines, large letters (like accented capital letters) still collide with such lines. Therefore at first we add to the height of a normal strut the value of the parameter \extrarowheight.

```
  \@tempdima \ht \strutbox
  \advance \@tempdima by\extrarowheight
  \setbox \@arstrutbox \hbox{\vrule
              \@height \arraystretch \@tempdima
              \@depth \arraystretch \dp \strutbox
              \@width \z@}%
```

Then we open a group, in which the user preamble is evaluated by the macro \@mkpream. As we know this must happen locally. This macro creates a preamble for a \halign and saves its result globally in the control sequence \@preamble.

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

We again redefine \@preamble so that a call up of \@preamble now starts the \halign. Thus also the arguments of >, <, @ and !, saved in the token registers are inserted into the preamble. The \tabskip at the beginning and end of the preamble is set to 0pt (in the beginning by the use of \ialign). Also the command \@arstrut is build in, which inserts the \@arstrutbox, defined above. Of course, the opening brace after \ialign has to be implicit as it will be closed in \endarray or another macro.
The \noexpand in front of \ialign does no harm in standard LaTeX and was added since some experimental support for using text glyphs in math redefines \halign with the result that is becomes expandable with disastrous results in cases like this. In the kernel definition for this macro the problem does not surface because there \protect is set (which is not necessary in this implementation as there is no arbitrary user input that can get expanded) and the experimental code made the redefinition robust. Whether this is the right approach is open to question; consider the \noexpand a curtesy to allow an

unsupported redefinition of a TEX primitive for the moment (as people rely on that experimental code).

```
\xdef\@preamble{\noexpand \ialign \@halignto
                \bgroup \@arstrut \@preamble
                        \tabskip \z@ \cr}%
```

What we have not explained yet is the macro `\@halignto` that was just used. Depending on its replacement text the `\halign` becomes a `\halign to` ⟨*dimen*⟩. Now we close the group again. Thus `\@startpbox` and `\@endpbox` as well as all `token` registers get their former meaning back.

```
\endgroup
```

To support other packages we include a hook into this part of the code which is a no-op in the main package.

```
\@arrayleft
```

Now we decide depending on the position argument in which `box` the `\halign` is to be put. (`\vcenter` may be used because we are in math mode.)

```
\if #1t\vtop \else \if#1b\vbox \else \vcenter \fi \fi
```

Now another implicit opening brace appears; then definitions which shall stay local follow. While constructing the `\@preamble` in `\@mkpream` the `#` sign must be hidden in the macro `\@sharp` which is `\let` to `\relax` at that moment (see definition of `\@mkpream`). All these now get their actual meaning.

```
\bgroup
\let \@sharp ##\let \protect \relax
```

With the above defined `struts` we fix down the distance between rows by setting `\lineskip` and `\baselineskip` to `0pt`. Since there have to be set `$`'s around every column in the `array`–environment the parameter `\mathsurround` should also be set to `0pt`. This prevents additional space between the rows. The PLAIN TEX–macro `\m@th` does this.

```
\lineskip \z@
\baselineskip \z@
\m@th
```

Beside, we have to assign a special meaning (which we still have to specify) to the line separator `\\`. We also have to redefine the command `\par` in such a way that empty lines in `\halign` cannot do any damage. We succeed in doing so by choosing something that will disappear when expanding. After that we only have to call up `\@preamble` to start the wanted `\halign`.

```
\let\\\@arraycr \let\tabularnewline\\\let\par\@empty \@preamble}
```

`\extrarowheight` The `dimen` parameter used above also needs to be allocated. As a default value we use `0pt`, to ensure compatibility with standard LATEX.

```
\newdimen \extrarowheight
\extrarowheight=0pt
```

`\@arstrut` Now the insertion of `\@arstrutbox` through `\@arstut` is easy since we know exactly in which mode TEX is while working on the `\halign` preamble.

```
\def\@arstrut{\unhcopy\@arstrutbox}
```

17.1.3 The line separator \\

\@arraycr In the macro \@array the line separator \\ is \let to the command \@arraycr. Its definition starts with a special brace which I have directly copied from the original definition. It is necessary, because the \futurlet in \@ifnextchar might expand a following & token in a construction like \\ &. This would otherwise end the alignment template at a wrong time. On the other hand we have to be careful to avoid producing a real group, i.e., {}, because the command will also be used for the array environment, i.e., in math mode. In that case an extra {} would produce an ord atom which could mess up the spacing. For this reason we use a combination that does not really produce a group at all but modifies the master counter so that a & will not be considered belonging to the current \halign while we are looking for a * or [. For further information see [Knu84, Appendix D].

```
\def\@arraycr{\relax\iffalse{\fi\ifnum 0=‘}\fi
```

Then we test whether the user is using the star form and ignore a possible star (I also disagree with this procedure, because a star does not make any sense here).

```
  \@ifstar \@xarraycr \@xarraycr}
```

\@xarraycr In the command \@xarraycr we test if an optional argument exists.

```
\def\@xarraycr{\@ifnextchar [%
```

If it does, we branch out into the macro \@argarraycr if not we close the special brace (mentioned above) and end the row of the \halign with a \cr.

```
  \@argarraycr {\ifnum 0=‘{}\fi\cr}}
```

\@argarraycr If additional space is requested by the user this case is treated in the macro \@argarraycr. First we close the special brace and then we test if the additional space is positive.

```
\def\@argarraycr[#1]{\ifnum0=‘{}\fi\ifdim #1>\z@
```

If this is the case we create an invisible vertical rule with depth \dp\@arstutbox + ⟨*wanted space*⟩. Thus we achieve that all vertical lines specified in the user preamble by a | are now generally drawn. Then the row ends with a \cr.

If the space is negative we end the row at once with a \cr and move back up with a \vskip.

While testing these macros I found out that the \endtemplate created by \cr and & is something like an \outer primitive and therefore it should not appear in incomplete \if statements. Thus the following solution was chosen which hides the \cr in other macros when TEX is skipping conditional text.

```
  \expandafter\@xargarraycr\else
  \expandafter\@yargarraycr\fi{#1}}
```

\@xargarraycr The following macros were already explained above.

\@yargarraycr

```
\def\@xargarraycr#1{\unskip
  \@tempdima #1\advance\@tempdima \dp\@arstrutbox
  \vrule \@depth\@tempdima \@width\z@ \cr}
\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
```

17.1.4 Spanning several columns

\multicolumn If several columns should be held together with a special format the command \multicolumn must be used. It has three arguments: the number of columns to be covered; the format for the result column and the actual column entry.

```
\long\def\multicolumn#1#2#3{%
```

First we combine the given number of columns into a single one; then we start a new block so that the following definition is kept local.

```
  \multispan{#1}\begingroup
```

Since a \multicolumn should only describe the format of a result column, we redefine \@addamp in such a way that one gets an error message if one uses more than one c, l, r, p, m or b in the second argument. One should consider that this definition is local to the build-up of the preamble; an array– or tabular–environment in the third argument of the \multicolumn is therefore worked through correctly as well.

```
  \def\@addamp{\if@firstamp \@firstampfalse \else
               \@preamerr 5\fi}%
```

Then we evaluate the second argument with the help of \@mkpream. Now we still have to insert the contents of the token register into the \@preamble, i.e. we have to say \xdef\@preamble{\@preamble}. This is achieved shorter by writing:

```
  \@mkpream{#2}\@addtopreamble\@empty
```

After the \@preamble is created we forget all local definitions and occupations of the token registers.

```
  \endgroup
```

In the special situation of \multicolumn \@preamble is not needed as preamble for a \halign but it is directly inserted into our table. Thus instead of \sharp there has to be the column entry (#3) wanted by the user.

```
  \def\@sharp{#3}%
```

Now we can pass the \@preamble to TEX . For safety we start with an \@arstrut. This should usually be in the template for the first column however we do not know if this template was overwritten by our \multicolumn. We also add a \null at the right end to prevent any following \unskip (for example from \\[..]) to remove the \tabcolsep.

```
  \@arstrut \@preamble
  \null
  \ignorespaces}
```

17.1.5 The environment definitions

After these preparations we are able to define the environments. They only differ in the initialisations of \d@llar..., \col@sep and \@halignto.

\@halignto \d@llarbegin \d@llarend In order to relieve the save stack we assign the replacement texts for \@halignto globally. \d@llar has to be local since otherwise nested tabular and array

environments (via \multicolumn) are impossible. When the new font selection scheme is in force we have to we surround all \halign entries with braces. See remarks in TUGboat 10#2. Actually we are going to use \begingroup and \endgroup. However, this is only necessary when we are in text mode. In math the surrounding dollar signs will already serve as the necessary extra grouping level. Therefore we switch the settings of \d@llarbegin and \d@llarend between groups and dollar signs.

```
\let\d@llarbegin\begingroup
\let\d@llarend\endgroup
```

\array Our new definition of \array then reads:

```
\def\array{\col@sep\arraycolsep
  \def\d@llarbegin{$}\let\d@llarend\d@llarbegin\gdef\@halignto{}%$
```

Since there might be an optional argument we call another macro which is also used by the other environments.

```
  \@tabarray}
```

\@tabarray This macro tests for an optional bracket and then calls up \@@array or \@@array[c] (as default).

```
\def\@tabarray{\@ifnextchar[{\@@array}{\@@array[c]}}
```

\@@array This macro could then test an optional delimiter before the left brace of the main preamble argument. At this point it simply is let to be \@array.

```
\let\@@array\@array
```

\tabular \tabular* The environments tabular and tabular* differ only in the initialisation of the command \@halignto. Therefore we define

```
\def\tabular{\gdef\@halignto{}\@tabular}
```

and analoguesly for the star form. We evalute the argument first using \setlength so that users of the calc package can write code like \begin{tabular*}{(\columnwidth-1cm)/2}...

```
\expandafter\def\csname tabular*\endcsname#1{%
      \setlength\dimen@{#1}%
      \xdef\@halignto{to\the\dimen@}\@tabular}
```

\@tabular The rest of the job is carried out by the \@tabular macro:

```
\def\@tabular{%
```

First of all we have to make sure that we start out in hmode. Otherwise we might find our table dangling by itself on a line.

```
  \leavevmode
```

It should be taken into consideration that the macro \@array must be called in math mode. Therefore we open a box, insert a $ and then assign the correct values to \col@sep and \d@llar....

```
  \hbox \bgroup $\col@sep\tabcolsep \let\d@llarbegin\begingroup%$
                                    \let\d@llarend\endgroup
```

Now everything tabular specific is done and we are able to call the \@tabarray macro.

```
\@tabarray}
```

\endarray \@arrayleft \@arrayright

When the processing of array is finished we have to close the \halign and afterwards the surrounding box selected by \@array. To save token space we then redefine \@preamble because its replacement text isn't longer needed. We have to declare the hook, \@arrayleft, we put into \@array above. A similar hook '\@arrayright' is inserted into the \endarray to gain control. Both defaults to empty.

```
\def\endarray{\crcr \egroup \egroup \@arrayright \gdef\@preamble{}}
\let\@arrayleft\@empty
\let\@arrayright\@empty
```

\endtabular \endtabular*

To end a tabular or tabular* environment we call up \endarray, close the math mode and then the surrounding \hbox.

```
\def\endtabular{\endarray $\egroup}%$
\expandafter\let\csname endtabular*\endcsname=\endtabular
```

We should \let all macros to \relax that were used in the kernel but are no longer necessary.

```
\let\@ampacol=\relax        \let\@expast=\relax
\let\@arrayclassiv=\relax   \let\@arrayclassz=\relax
\let\@tabclassiv=\relax     \let\@tabclassz=\relax
\let\@arrayacol=\relax      \let\@tabacol=\relax
\let\@tabularcr=\relax      \let\@@endpbox=\relax
\let\@argtabularcr=\relax   \let\@xtabularcr=\relax
```

\@preamerr

We also have to redefine the error routine \@preamerr since new kind of errors are possible. The code for this macro is not perfect yet; it still needs too much memory.

```
\def\@preamerr#1{\def\@tempd{{..} at wrong position: }%
   \ClassError{memoir}{%
   \ifcase #1 Illegal pream-token (\@nextchar): `c' used\or %0
    Missing arg: token ignored\or                          %1
    Empty preamble: `l' used\or                            %2
    >\@tempd token ignored\or                              %3
    <\@tempd changed to !{..}\or                           %4
    Only one column-spec. allowed.\fi}\@ehc}               %5

```

17.1.6 Defining your own column specifiers

\newcolumntype

In newarray.sty the macro for specifying new columns was named \newcolumn. When the functionality was added to array.sty the command was renamed \newcolumntype. The \newcolumntype macro gives users the chance to define letters, to be used in the same way as the primitive column specifiers, 'c' 'p' etc.

```
\def\newcolumntype#1{%
```

`\NC@char` is so that active characters, like `@` in AMSLaTeX may be used. Note that we need to use the possibly active token, `#1`, in several places, as that is the token that actually appears in the preamble argument.

```
\edef\NC@char{\string#1}%
```

First we check whether there is already a definition for this column. Unlike `\newcommand` we give a warning rather than an error if it is defined. If it is a new column, add `\NC@do` ⟨*column*⟩ to the list `\NC@list`.

```
\@ifundefined{NC@find@\NC@char}%
  {\@tfor\next:=<>clrmbp@!|\do{\if\noexpand\next\NC@char
      \@memwarn{Redefining primitive column \NC@char}\fi}%
   \NC@list\expandafter{\the\NC@list\NC@do#1}}%
  {\@memwarn{Column \NC@char\space is already defined}}%
```

Now we define a macro with an argument delimited by the new column specifier, this is used to find occurences of this specifier in the user preamble.

```
\@namedef{NC@find@\NC@char}##1#1{\NC@{##1}}%
```

If an optional argument was not given, give a default argument of 0.

```
\@ifnextchar[{\newcol@{\NC@char}}{\newcol@{\NC@char}[0]}}
```

`\newcol@` We can now define the macro which does the rewriting, `\@reargdef` takes the same arguments as `\newcommand`, but does not check that the command is new. For a column, say 'D' with one argument, define a command `\NC@rewrite@D` with one argument, which recursively calls `\NC@find` on the user preamble after replacing the first token or group with the replacement text specified in the `\newcolumntype` command. `\NC@find` will find the next occurrence of 'D' as it will be `\let` equal to `\NC@find@D` by `\NC@do`.

```
\def\newcol@#1[#2]#3{\expandafter\@reargdef
    \csname NC@rewrite@#1\endcsname[#2]{\NC@find#3}}
```

`\NC@` Having found an occurence of the new column, save the preamble before the column in `\@temptokena`, then check to see if we are at the end of the preamble. (A dummy occurrence of the column specifier will be placed at the end of the preamble by `\NC@do`).

```
\def\NC@#1{%
  \@temptokena\expandafter{\the\@temptokena#1}\futurelet\next\NC@ifend}
```

`\NC@ifend` We can tell that we are at the end as `\NC@do` will place a `\relax` after the dummy column.

```
\def\NC@ifend{%
```

If we are at the end, do nothing. (The whole preamble will now be in `\@temptokena`.)

```
  \ifx\next\relax
```

Otherwise set the flag `\if@tempswa`, and rewrite the column.

```
  \else\@tempswatrue\expandafter\NC@rewrite\fi}
```

\NC@do If the user has specified 'C' and 'L' as new columns, the list of rewrites (in the token register \NC@list) will look like \NC@do * \NC@do C \NC@do L. So we need to define \NC@do as a one argument macro which initialises the rewriting of the specified column. Let us assume that 'C' is the argument.

```
\def\NC@do#1{%
```

First we let \NC@rewrite and \NC@find be \NC@rewrite@C and \NC@find@C respectively.

```
  \expandafter\let\expandafter\NC@rewrite
    \csname NC@rewrite@\string#1\endcsname
  \expandafter\let\expandafter\NC@find
    \csname NC@find@\string#1\endcsname
```

Clear the token register \@temptokena after putting the present contents of the register in front of the token \NC@find. At the end we place the tokens 'C\relax' which \NC@ifend will use to detect the end of the user preamble.

```
  \expandafter\@temptokena\expandafter{\expandafter}%
        \expandafter\NC@find\the\@temptokena#1\relax}
```

\showcols This macro is useful for debugging \newcolumntype specifications, it is the equivalent of the primitive \show command for macro definitions. All we need to do is locally redefine \NC@do to take its argument (say 'C') and then \show the (slightly modified) definition of \NC@rewrite@C. Actually as the the list always starts off with \NC@do * and we do not want to print the definition of the *-form, define \NC@do to throw away the first item in the list, and then redefine itsef to print the rest of the definitions.

```
\def\showcols{{\def\NC@do##1{\let\NC@do\NC@show}\the\NC@list}}
```

\NC@show If the column 'C' is defined as above, then \show\NC@rewrite@C would output \long macro: ->\NC@find >{$}c<{$}. We want to strip the \long macro: -> and the \NC@find. So first we use \meaning and then apply the macro \NC@strip to the tokens so produced and then \typeout the required string.

```
\def\NC@show#1{%
  \typeout{Column #1\expandafter\expandafter\expandafter\NC@strip
  \expandafter\meaning\csname NC@rewrite@#1\endcsname\@@}}
```

\NC@strip Delimit the arguments to \NC@strip with ':', '->', a space, and \@@ to pull out the required parts of the output from \meaning.

```
\def\NC@strip#1:#2->#3 #4\@@{#2 -> #4}
```

\NC@list Allocate the token register used for the rewrite list.

```
\newtoks\NC@list
```

17.1.7 The *–form

We view the *-form as a slight generalisation of the system described in the previous subsection. The idea is to define a * column by a command of the form:

```
\newcolumntype{*}[2]{%
   \count@=#1\ifnum\count@>0
      \advance\count@ by -1 #2*{\count@}{#2}\fi}
```

\NC@rewrite@* This does not work however as \newcolumntype takes great care not to expand anything in the preamble, and so the \if is never expanded. \newcolumntype sets up various other parts of the rewrite correctly though so we can define:

```
\newcolumntype{*}[2]{}
```

Now we must correct the definition of \NC@rewrite@*. The following is probably more efficient than a direct translation of the idea sketched above, we do not need to put a * in the preamble and call the rewrite recursively, we can just put #1 copies of #2 into \@temptokena. (Nested * forms will be expanded when the whole rewrite list is expanded again, see \@mkpream)

```
\long\@namedef{NC@rewrite@*}#1#2{%
```

Store the number.

```
  \count@#1\relax
```

Put #1 copies of #2 in the token register.

```
  \loop
  \ifnum\count@>\z@
    \advance\count@\m@ne
    \@temptokena\expandafter{\the\@temptokena#2}%
  \repeat
```

\NC@do will ensure that \NC@find is \let equal to \NC@find@*.

```
  \NC@find}

```

17.2 Getting the spacing around rules right

The Companion [GMS94, p.137] suggests two additional commands to control the allignments in case of tabulars with horizontal lines. They are now added to this package.

\extratabsurround The extra space around a table when \firsthline or \lasthline are used.

```
\newlength{\extratabsurround}
\setlength{\extratabsurround}{2pt}
```

\backup@length This register will be used internally by \firsthline and \lasthline.

```
\newlength{\backup@length}
```

\firsthline This code can probably be improved but for the moment it should serve.
We start by producing a single tabular row without any visible content that will produce the external reference point in case [t] is used.

```
\newcommand{\firsthline}{%
  \multicolumn1c{%
```

Within this row we calculate \backup@length to be the height plus depth of a standard line. In addition we have to add the width of the \hline, something that was forgotten in the original definition.

```
    \global\backup@length\ht\@arstrutbox
    \global\advance\backup@length\dp\@arstrutbox
    \global\advance\backup@length\arrayrulewidth
```

Finally we do want to make the height of this first line be a bit larger than usual, for this we place the standard array strut into it but raised by \extratabsurround

```
     \raise\extratabsurround\copy\@arstrutbox
```

Having done all this we end the line and back up by the value of \backup@length and then finally place our \hline. This should place the line exactly at the right place but keep the reference point of the whole tabular at the baseline of the first row.

```
    }\\[-\backup@length]\hline
}
```

\lasthline For \lasthline the situation is even worse and I got it completely wrong initially.
The problem in this case is that if the optional argument [b] is used we do want the reference point of the tabular be at the baseline of the last row but at the same time do want the the depth of this last line increased by \extratabsurround without changing the placement \hline.
We start by placing the rule followed by an invisible row.

```
\newcommand{\lasthline}{\hline\multicolumn1c{%
```

We now calculate \backup@length to be the height and depth of two lines plus the width of the rule.

```
    \global\backup@length2\ht\@arstrutbox
    \global\advance\backup@length2\dp\@arstrutbox
    \global\advance\backup@length\arrayrulewidth
```

This will bring us back to the baseline of the second last row:

```
    }\\[-\backup@length]%
```

Thus if we now add another invisible row the reference point of that row will be at the baseline of the last row (and will be the reference for the whole tabular). Since this row is invisible we can enlarge its depth by the desired amount.

```
    \multicolumn1c{%
       \lower\extratabsurround\copy\@arstrutbox
       }%
}
```

Beside a larger functionality array.sty has one important difference to the standard tabular and array environments: horizontal and vertical rules make a table larger or wider, e.g., \doublerulesep really denotes the space between two rules and isn't measured from the middle of the rules.

\@xhline For vertical rules this is implemented by the definitions above, for horizontal rules we have to take out the backspace.

```
\CheckCommand*\@xhline{\ifx\reserved@a\hline
               \vskip\doublerulesep
               \vskip-\arrayrulewidth
             \fi
      \ifnum0=‘{\fi}}
\renewcommand*\@xhline{\ifx\reserved@a\hline
               \vskip\doublerulesep
             \fi
      \ifnum0=‘{\fi}}

%%%%%%%%%% end Array package code %%%%%%%%%%%%%%%%%%%%
```

17.3 D column specifiers

The description and code are essentially copied from the dcolumn package [Car01].

```
%%%%%%%%% Dcolumn package code %%%%%%%%%%%%%%%%%%%%
%%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%%%%%%%%%%%%
```

The dcolumn package [Car01] defines `D` to be a column specifier with three arguments.
`D{`⟨*sep.tex*⟩`}{`⟨*sep.dvi*⟩`}{`⟨*decimal places*⟩`}` for columns which are to be aligned on a ‘decimal point’.
The basic ideas behind these macros have been explained earlier. However they use three tricks which may be useful in other contexts.

- The separator is surrounded in extra `{ }`, so that it is set with `\mathord` spacing, otherwise, for instance a ‘,’ would have extra space after it.
- The separator is not given its special definition by making it active, as this would not work for an entry such as `& .5 &`, as the first token of an alignment entry is read *before* the preamble part, incase it is an `\omit`, in which case the preamble is to be omitted. Instead we switch the mathcode to (hex) 8000, which makes the token act as if it were active.
- Although `\mathcode‘.="8000` makes `.` act as if it were active, it is still not allowed in constructions such as `\def.{}`, even in math-mode, so we have to construct an active version of the separator, this is done by making it the uppercase of `~`, and then using the construct `\uppercase{\def~}{`⟨*definition*⟩`}`.
  Note that the ⟨*definition*⟩ is not uppercased, so the definition can refer to the standard, non-active use of the separator.

`\DC@` Set up uppercase tables as required, and then grab the first part of the numerical argument into `\count@`.

```
\def\DC@#1#2#3{%
  \uccode`\~=`#1\relax
  \m@th
  \afterassignment\DC@x\count@#3\relax{#1}{#2}}
```

`\DC@x` If `\count@` is negative, centre on the decimal point. If it is positive either `#1` will be empty in which case bad out decimal part to the number of digits specified by `\count@` or it is none empty in which case `\count@` contains the number of digits to the left of the point, and `#1` contains a junk token (probably `.`) followed by the number of digits to the right of the point. In either of these latter cases, `\DC@right` is used.

```
\def\DC@x#1\relax#2#3{%
  \ifnum\z@>\count@
    \expandafter\DC@centre
  \else
    \expandafter\DC@right
  \fi
  {#2}{#3}{#1}}
```

`\DC@centre` If centering on the decimal point, just need to box up the two halves.

```
\def\DC@centre#1#2#3{%
  \let\DC@end\DC@endcentre
  \uppercase{\def~}{$\egroup\setbox\tw@=\hbox\bgroup${#2}}%$
  \setbox\tw@=\hbox{${\phantom{{#2}}}$}%
  \setbox\z@=\hbox\bgroup$\mathcode`#1="8000 }%$
```

`\DC@endcentre` and then pad out the smaller of the two boxes so there is the same amount of stuff either side of the point.

```
\def\DC@endcentre{$\egroup%$
    \ifdim \wd\z@>\wd\tw@
      \setbox\tw@=\hbox to\wd\z@{\unhbox\tw@\hfill}%
    \else
      \setbox\z@=\hbox to\wd\tw@{\hfill\unhbox\z@}\fi
    \box\z@\box\tw@}
```

`\DC@right` This deals with both the cases where a specified number of decimal places is given.

```
\def\DC@right#1#2#3{%
  \ifx\relax#3\relax
```

If `#3` is empty, add `\hfill` to right align the column, and Just set `\DC@rl` to begin a group, so nothing fancy is done with the whole number part.

```
    \hfill
    \let\DC@rl\bgroup
  \else
```

Otherwise set \DC@rl so that the whole number part is put in a box \count@ times as wide as a digit. In order to share code with the other branch, then move #3 (the number of decimal places) into \count@ throwing away the '.' from the user syntax.

```
    \edef\DC@rl{to\the\count@\dimen@ii\bgroup\hss\hfill}%
    \count@\@gobble#3\relax
  \fi
  \let\DC@end\DC@endright
```

Box 2 contains the decimal part, set to \dimen@ which is calculated below to be \count@ times the width of a digit, plus the width of the 'decimal point'.

```
  \uppercase{\def~}{$\egroup\setbox\tw@\hbox to\dimen@\bgroup${#2}}%
   \setbox\z@\hbox{$1$}\dimen@ii\wd\z@
   \dimen@\count@\dimen@ii
   \setbox\z@\hbox{${#2}$}\advance\dimen@\wd\z@
   \setbox\tw@\hbox to\dimen@{}%
```

Box 0 contains the whole number part, either just at its natural size for right aligned columns, or set to (the old value of) \count@ times the width of a digit. \DC@rl defined above determines the two cases.

```
   \setbox\z@\hbox\DC@rl$\mathcode`#1="8000 }%$
```

\DC@endright Just finish off the second box, and then put out both boxes.

```
\def\DC@endright{$\hfil\egroup\box\z@\box\tw@}%$
```

D The user interface, define the D column to take three arguments. For special purposes, you may need to directly access \DC@ rather than the D column, eg to get a bold version you could use

```
\newcolumntype{E}[3]{>{\boldmath\DC@{#1}{#2}{#3}}c<{\DC@end}}
```

```
\newcolumntype{D}[3]{>{\DC@{#1}{#2}{#3}}c<{\DC@end}}

%%%%%%%%%% end Dcolumn package code %%%%%%%%%%%%%%%%%%%%
```

17.4 Support for delimeters

The description and code are essentially from the delarray package [Car94], v1.01 1994/03/14.

```
%%%%%%%%%% Delarray package code %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%%%%%%%%%%%%
```

The array syntax is extended by supporting the notation of delimiters. To this end we extend the array parsing mechanism to include a hook which can be used by this (or another) package to do some additional parsing.

`\@@array` This macro tests for an optional delimiter before the left brace of the main preamble argument. If there is no delimiter, `\@arrayleft` and `\@arrayright` are made a no-ops, and `\@array` is called with the positional argument. Otherwise call `\@del@array`.

```
\def\@@array[#1]{\@ifnextchar\bgroup
  {\let\@arrayleft\relax\let\@arrayright\relax\@array[#1]}%
  {\@del@array[#1]}}
```

`\@del@array` We now know that we have an `array` (or `tabular`) with delimiters.

```
\def\@del@array[#1]#2#3#4{%
```

The following line is completely redundant but it does catch errors involving delimiters before the processing of the alignment begins. A common error is likely to be omiting the '.' in a `\cases`-type construction. This causes the first token of the alignment to be gobbled, possibly causing lots of spurious errors before the cause of the error, the missing delimiter, is discovered as `\@arrayright` puts the alignment and the delimiters together.

```
  \setbox\z@\hbox{$\left#2\right#4$}%
```

In the case of a 'c' argument we do not need to rebox the alignment, so we can define `\@arrayleft` and `\@arrayright` just to insert the delimiters.

```
  \if#1c\def\@arrayleft{\left#2}\def\@arrayright{\right#4}%
```

Otherwise we (should) have a `[t]` or `[b]` argument, so first we store the alignment, without delimiters in box0.

```
  \else\def\@arrayleft{\setbox\z@}%
```

Then after the alignment is finished:

```
  \def\@arrayright{%
```

Calculate the amount the box needs to be lowered (this will be negative in the case of `[b]`). A little bit of arithmetic cf. *The TEXbook*, Appendix G, rule 8. We calculate the amount this way, rather than just taking the difference between the depth of box0 and the depth of the box defined below, as the depth of that box may be affected by the delimiters if `\delimitershortfall` or `\delimiterfactor` have non-standard values.

```
    \dimen@=\dp\z@
    \advance\dimen@-\ht\z@
    \divide \dimen@ by \tw@
    \advance\dimen@ by\fontdimen22 \textfont\tw@
```

Now lower the alignment and the delimiters into place.

```
    \lower\dimen@\hbox{$\left#2\vcenter{\unvbox\z@}\right#4$}}%
```

End the `\if#1c`

```
  \fi
```

Now that we have defined `\@arrayleft` and `\@arrayright`, call `\@array`.

```
  \@array[#1]{#3}}

%%%%%%%%% end Delarray package code %%%%%%%%%%%%%%%%%%%%
```

17.5 The tabularx environment

The code and description are essentially from David Carlisle's tabularx package [Car99], v2.07 1999/01/07.

```
%%%%%%%%%% Tabularx package code %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% With acknowledgements to David Carlisle %%%%%%%%%%%%%%%%%%%%%
```

The tabularx package [Car99] implements a version of the `tabular` environment in which the widths of certain columns are calculated so that the table is is a specified width. The columns that may stretch are marked by an `X` in the preamble argument.
First some registers etc. that we need.

```
\newdimen\TX@col@width
\newdimen\TX@old@table
\newdimen\TX@old@col
\newdimen\TX@target
\newdimen\TX@delta
\newcount\TX@cols
\newif\ifTX@
```

Now a trick to get the body of an environment into a token register, without doing any expansion. This does not do any real checking of nested environments, so if you should need to nest one `tabularx` inside another, the inner one must be surrounded by `{ }`.

`\tabularx` Just save the width specification separately, then clear the token register `\toks@`. Finally call `\TX@get@body` to begin saving the body of the table. The `{\ifnum0=‘}\fi` allows `tabularx` to appear inside a `\halign`.[15]
This mechanism of grabbing an environment body does have the disadvantage (shared with the AMS alignment environments) that you can not make extension environments by code such as

```
\newenvironment{foo}{\begin{tabularx}{XX}}{\end{tabularx}}
```

as the code is looking for a literal string `\end{tabularx}` to stop scanning. One may avoid this problem by using `\tabularx` and `\endtabularx` directly in the definition:

```
\newenvironment{foo}{\tabularx{XX}}{\endtabularx}
```

The scanner now looks for the end of the current environment (`foo` in this example.) There are some restrictions on this usage, the principal one being that `\endtabularx` is the *first* token of the ‘end code’ of the environment.

```
\def\tabularx#1{%
```

[15] This adds an extra level of grouping, which is not really needed. Instead, I could use `\iffalse{\fi\ifnum0=‘}\fi` here, and `\ifnum0=‘{}\fi` below, however the code here would then have to be moved after the first line, because of the footnote to page 386 of *The TEXbook*, and I do not think I should be writing code that is so obscure as to be documented in a footnote in an appendix called “Dirty Tricks”!

Allow \tabularx \endtabularx (but not \begin{tabularx} \end{tabularx}) to be used in \newenvironment definitions.

```
\edef\TX@{\@currenvir}%
  {\ifnum0=`}\fi
```

\setlength lets you use the calc package so you could use a width of, say (\textwidth-12pt)/2.

```
  \setlength\TX@target{#1}%
  \TX@typeout{Target width: #1 = \the\TX@target.}%
  \toks@{}\TX@get@body}
```

\endtabularx This does not do very much...

```
\let\endtabularx\relax
```

\TX@get@body Place all tokens as far as the first \end into a token register. Then call \TX@find@end to see if we are at \end{tabularx}.

```
\long\def\TX@get@body#1\end
  {\toks@\expandafter{\the\toks@#1}\TX@find@end}
```

\TX@find@end If we are at \end{tabularx}, call \TX@endtabularx, otherwise add \end{...} to the register, and call \TX@get@body again.

```
\def\TX@find@end#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\TX@\expandafter\TX@endtabularx
  \else\toks@\expandafter
    {\the\toks@\end{#1}}\expandafter\TX@get@body\fi}
```

\TX@ The string tabularx as a macro for testing with \ifx.

```
\def\TX@{tabularx}
```

Now that all the parts of the table specification are stored in registers, we can begin the work of setting the table.
The algorithm for finding the correct column widths is as follows. Firstly set the table with each X column the width of the final table. Assuming that there is at least one X column, this will produce a table that is too wide. Divide the excess width by the number of X columns, and reduce the column width by this amount. Reset the table. If the table is not now the correct width, a \multicolumn entry must be 'hiding' one of the X columns, and so there is one less X column affecting the width of the table. So we reduce by 1 the number of X columns and repeat the process.

\TX@endtabularx Although I have tried to make tabularx look like an environment, it is in fact a command, all the work is done by this macro.

```
\def\TX@endtabularx{%
```

Define the X column, with an internal version of the \newcolumntype command. The \expandafter commands enable \NC@newcol to get the *expansion* of \tabularxcolumn{\TX@col@width} as its argument. This will be the definition of an X column.

```
  \expandafter\TX@newcol\expandafter{\tabularxcolumn{\TX@col@width}}%
```

Initialise the column width, and the number of `X` columns. The number of `X` columns is set to one, which means that the initial count will be one too high, but this value is decremented before it is used in the main loop.

```
\let\verb\TX@verb
```

Save the values of all LaTeX counters; the list `\cl@@ckpt` contains the names of all the LaTeX counters that have been defined so far. We expand `\setcounter` at this point, as it results in fewer tokens being stored in `\TX@ckpt`, but the actual resetting of the counters occurs when `\TX@ckpt` is expanded after each trial run. Actually, use something equivalent to the expansion of the original definition of `\setcounter`, so that `tabularx` works in conjunction with `calc.sty`.

```
\def\@elt##1{\global\value{##1}\the\value{##1}\relax}%
\edef\TX@ckpt{\cl@@ckpt}%
\let\@elt\relax
\TX@old@table\maxdimen
\TX@col@width\TX@target
\global\TX@cols\@ne
```

Typeout some headings (unless this is disabled).

```
\TX@typeout@
  {\@spaces Table Width\@spaces Column Width\@spaces X Columns}%
```

First attempt. Modify the `X` definition to count `X` columns.

```
\TX@trial{\def\NC@rewrite@X{%
          \global\advance\TX@cols\@ne\NC@find p{\TX@col@width}}}%
```

Repeatedly decrease column width until table is the correct width, or stops shrinking, or the columns become two narrow. If there are no multicolumn entries, this will only take one attempt.

```
\loop
  \TX@arith
  \ifTX@
  \TX@trial{}%
\repeat
```

One last time, with warnings back on (see appendix D) use `tabular*` to put it in a box of the right size, in case the algorithm failed to find the correct size. Locally make `\footnotetext` save its argument in a token register. `\toks@` contains the preamble specification, and possible optional argument, as well as the table body.

```
{\let\@footnotetext\TX@ftntext\let\@xfootnotenext\TX@xftntext
  \csname tabular*\expandafter\endcsname\expandafter\TX@target
    \the\toks@
  \csname endtabular*\endcsname}%
```

Now the alignment is finished, and the `}` has restored the original meaning of `\@footnotetext` expand the register `\TX@ftn` which will execute a series of `\footnotetext[<num>]{<note>}` commands. We need to be careful about clearing the register as we may be inside a nested `tabularx`.

```
\global\TX@ftn\expandafter{\expandafter}\the\TX@ftn
```

Now finish off the tabularx environment. Note that we need \end{tabularx} here as the \end{tabularx} in the user's file is never expanded. Now use \TX@ rather than tabularx.
We also need to finish off the group started by {\ifnum0=‘}\fi in the macro \tabularx.

```
\ifnum0=‘{\fi}%
\expandafter\end\expandafter{\TX@}}
```

\TX@arith Calculate the column width for the next try, setting the flag \ifTX@ to false if the loop should be aborted.

```
\def\TX@arith{%
  \TX@false
  \ifdim\TX@old@table=\wd\@tempboxa
```

If we have reduced the column width, but the table width has not changed, we stop the loop, and output the table (which will cause an over-full alignment) with the previous value of \TX@col@width.

```
    \TX@col@width\TX@old@col
    \TX@typeout@{Reached minimum width, backing up.}%
  \else
```

Otherwise calculate the amount by which the current table is too wide.

```
    \dimen@\wd\@tempboxa
    \advance\dimen@ -\TX@target
    \ifdim\dimen@<\TX@delta
```

If this amount is less than \TX@delta, stop. (\TX@delta should be non-zero otherwise we may miss the target due to rounding error.)

```
      \TX@typeout@{Reached target.}%
    \else
```

Reduce the number of effective X columns by one. (Checking that we do not get 0, as this would produce an error later.) Then divide excess width by the number of effective columns, and calculate the new column width. Temporarily store this value (times -1) in \dimen@.

```
      \ifnum\TX@cols>\@ne
        \advance\TX@cols\m@ne
      \fi
      \divide\dimen@\TX@cols
      \advance\dimen@ -\TX@col@width
      \ifdim \dimen@ >\z@
```

If the new width would be too narrow, abort the loop. At the moment too narrow, means less than 0 pt!
If the loop was aborted here and the X columns were left with the width of the previous run it may make the table far too wide as initial guesses are always too big. Force to \TX@error@width which defaults to be 1em.

```
        \@memwarn{X Columns too narrow (table too wide)\MessageBreak}%
        \TX@col@width\TX@error@width\relax
      \else
```

Otherwise save the old settings, and set the new column width. Set the flag to true so that the table will be set, and the loop will be executed again.

```
        \TX@old@col\TX@col@width
        \TX@old@table\wd\@tempboxa
        \TX@col@width-\dimen@
        \TX@true
      \fi
    \fi
  \fi}
```

`\TX@error@width` If the calculated width is negative, use this instead.

```
\def\TX@error@width{1em}
```

`\TX@delta` Accept a table that is within `\hfuzz` of the correct width.

```
\TX@delta\hfuzz
```

Initialise the `X` column. The definition can be empty here, as it is set for each `tabularx` environment.

```
\newcolumntype{X}{}
```

`\tabularxcolumn` The default definition of `X` is `p{#1}`.

```
\def\tabularxcolumn#1{p{#1}}
```

`\TX@newcol` A little macro just used to cut down the number of `\expandafter` commands needed.

```
\def\TX@newcol{\newcol@{X}[0]}
```

`\TX@trial` Make a test run.

```
\def\TX@trial#1{%
  \setbox\@tempboxa\hbox{%
```

Any extra commands. This is used on the first run to count the number of `X` columns.

```
    #1\relax
```

Make `\footnotetext` gobble its arguments. Also locally clear `\TX@vwarn` so that the warning is generated by the `final` run, and does not appear in the middle of the table if `\tracingtabularx`.

```
  \let\@footnotetext\TX@trial@ftn
  \let\TX@vwarn\@empty
```

Do not nest `tabularx` environments during trial runs. This would waste time, and the global setting of `\TX@cols` would break the algorithm.

```
  \expandafter\let\expandafter\tabularx\csname tabular*\endcsname
  \expandafter\let\expandafter\endtabularx\csname endtabular*\endcsname
```

Dissable `\write`s during a trial run. This trick is from *The TEXbook*.[16]

```
  \def\write{\begingroup
    \def\let{\afterassignment\endgroup\toks@}%
      \afterassignment\let\count@}%
```

[16] Actually the trick does not work correctly, so it has been changed.

Turn off warnings (see appendix D). Also prevent them being turned back on by setting the parameter names to be registers.

```
    \hbadness\@M
    \hfuzz\maxdimen
    \let\hbadness\@tempcnta
    \let\hfuzz\@tempdima
```

Make the table, and finish the hbox. `\toks@` contains the preamble specification, and possible optional argument, as well as the table body.

```
    \expandafter\tabular\the\toks@
    \endtabular}%
```

Reset all LaTeX counters, by executing `\TX@ckpt`.

```
  \TX@ckpt
```

Print some statistics.

```
  \TX@typeout@{\@spaces
     \expandafter\TX@align
        \the\wd\@tempboxa\space\space\space\space\space\@@
     \expandafter\TX@align
        \the\TX@col@width\space\space\space\space\space\@@
     \@spaces\the\TX@cols}}
```

`\TX@align` Macro to improve the printing of the tracing info.

```
\def\TX@align#1.#2#3#4#5#6#7#8#9\@@{%
  \ifnum#1<10 \space\fi
  \ifnum#1<100 \space\fi
  \ifnum#1<\@m\space\fi
  \ifnum#1<\@M\space\fi
  #1.#2#3#4#5#6#7#8\space\space}
```

`\arraybackslash` \\ hack.

```
\def\arraybackslash{\let\\\@arraycr}
```

`\tracingtabularx` Print statistics on column and table widths.

```
\def\tracingtabularx{%
  \def\TX@typeout{\ClassWarningNoLine{memoir}}%
  \def\TX@typeout@##1{\typeout{(tabularx) ##1}}}
```

`\TX@typeout` The default is to be to be quiet

```
\let\TX@typeout\@gobble
\let\TX@typeout@\@gobble
```

`\TX@ftn` A token register for saving footnote texts.

```
\newtoks\TX@ftn
```

`\TX@ftntext` `\TX@xftntext` Inside the alignment just save up the footnote text in a token register.

```
\long\def\TX@ftntext#1{%
  \edef\@tempa{\the\TX@ftn\noexpand\footnotetext
                      [\the\csname c@\@mpfn\endcsname]}%
```

```
  \global\TX@ftn\expandafter{\@tempa{#1}}}%
\long\def\TX@xftntext[#1]#2{%
  \global\TX@ftn\expandafter{\the\TX@ftn\footnotetext[#1]{#2}}}
```

`\TX@trial@ftn` On trial runs, gobble footnote texts.

```
\long\def\TX@trial@ftn#1{}
```

This last section was added at Version 1.02. Previous versions documentented the fact that `\verb` did not work inside `tabularx`, but that did not stop people using it! This usually put LaTeX into an irrecoverable error position, with error messages that did not mention the cause of the error. The 'poor man's `\verb`' (and `\verb*`) defined here is based on page 382 of *The TeXbook*. As explained there, doing verbatim this way means that spaces are not treated correctly, and so `\verb*` may well be useless, however I consider this section of code to be error-recovery, rather than a real implementation of verbatim.
The mechanism is quite general, and any macro which wants to allow a form of `\verb` to be used within its argument may `\let\verb=\TX@verb`. (Making sure to restore the real definition later!)
`\verb` and `\verb*` are subject to the following restictions:

1. Spaces in the argument are not read verbatim, but may be skipped according to TeX's usual rules.
2. Spaces will be added to the output after control words, even if they were not present in the input.
3. Unless the argument is a single space, any trailing space, whether in the original argument, or added as in (2), will be omitted.
4. The argument must not end with \, so `\verb|\|` is not allowed, however, because of (3), `\verb|\ |` produces \.
5. The argument must be balanced with respect to `{` and `}`. So `\verb|{|` is not allowed.
6. A comment character like `%` will not appear verbatim. It will act as usual, commenting out the rest of the input line!
7. The combinations `?‘` and `!‘` will appear as ¿ and ¡ if the `cmtt` font is being used.

`\TX@verb` The internal definition of `\verb`. Spaces will be replaced by `~`, so for the star-form, `\let ~` be `\verb*| |`, which we obtain as `\uppercase{*}`. Use `{\ifnum0=‘}\fi` rather than `\bgroup` to allow `&` to appear in the argument.

```
{\uccode‘\*=‘\ %
\uppercase{\gdef\TX@verb{%
  \leavevmode\null\TX@vwarn
  {\ifnum0=‘}\fi\ttfamily\let\\\ignorespaces
  \@ifstar{\let~*\TX@vb}{\TX@vb}}}}
```

`\TX@vb` Get the 'almost verbatim' text using `\meaning`. The '`!`' is added to the front of the user supplied text, to ensure that the whole argument does not consist of a single `{ }` group. TEX would strip the outer braces from such a group. The '`!`' will be removed later.
Originally I followed Knuth, and had `\def\@tempa{##1}`, however this did not allow `#` to appear in the argument. So I changed this to to use a token register, and `\edef`. This allows `#` appear, but makes each one appear twice!, so later we loop through, replacing `##` by `#`.

```
\def\TX@vb#1{\def\@tempa##1#1{\toks@{##1}\edef\@tempa{\the\toks@}%
    \expandafter\TX@v\meaning\@tempa\\ \\\ifnum0=`{\fi}}\@tempa!}
```

`\TX@v` Strip the initial segment of the `\meaning`, including the '`!`' added earlier.

```
\def\TX@v#1!{\afterassignment\TX@vfirst\let\@tempa= }
```

As explained above we are going to replace `##` pairs by `#`. To do this we need non-special `#` tokens. Make `*` into a parameter token so that we can define macros with arguments. The normal meanings will be restored by the `\endgroup` later.

```
\begingroup
\catcode`\*=\catcode`\#
\catcode`\#=12
```

`\TX@vfirst` As a special case, prevent the first character from being dropped. This makes `\verb*|  |` produce ␣. Then call `\TX@v@`. This is slightly tricky as I have to ensure that an actual `#` rather than a command `\let` to `#` is passed on if the first character is `#`.

```
\gdef\TX@vfirst{%
  \if\@tempa#%
    \def\@tempb{\TX@v@#}%
  \else
    \let\@tempb\TX@v@
    \if\@tempa\space~\else\@tempa\fi
  \fi
  \@tempb}
```

`\TX@v@` Loop through the `\meaning`, replacing all spaces by `~`. If the last character is a space it is dropped, so that `\verb*|\LaTeX|` produces `\LaTeX` not `\LaTeX␣`. The rewritten tokens are then further processed to replace `##` pairs.

```
\gdef\TX@v@*1 *2{%
  \TX@v@hash*1##\relax\if*2\\\else~\expandafter\TX@v@\fi*2}
```

`\TX@v@hash` The inner loop, replacing `##` by `#`.

```
\gdef\TX@v@hash*1##*2{*1\ifx*2\relax\else#\expandafter\TX@v@hash\fi*2}
```

As promised, we now restore the normal meanings of `#` and `*`.

```
\endgroup
```

\TX@vwarn Warn the user the first time this \verb is used.

```
\def\TX@vwarn{%
  \@warning{\noexpand\verb may be unreliable inside tabularx}%
  \global\let\TX@vwarn\@empty}

%%%%%%%%%% end Tabularx package code %%%%%%%%%%%%%%%%%%%%
```

At this point we had better stop anyone trying to load the several tabular-related packages (as effectively they are already loaded).

```
%%\@memfakeusepackage{array}
%%\@memfakeusepackage{dcolumn}
%%\@memfakeusepackage{delarray}
%%\@memfakeusepackage{tabularx}
```

17.6 Fear's rules

Simon Fear disapproves of the default LaTeX table rules and wrote the booktabs package [Fea03] to provide better horizontal rules. Like many typographers, he abhors vertical rules.

\bktabrule The rules in this section are based on \hrule \@height#1 but this does not work for continuous tabulars. Provide a version of this that can be \let when necessary.

```
\newcommand*{\bktabrule}[1]{%
  \hrule \@height#1}
```

The following is taken almost verbatim from the booktabs package, version 1.618, March 2003.

```
%%%%%%%%%% Booktabs package code       %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% slightly modified by PRW    %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% by permission of Simon Fear %%%%%%%%%%%%%%%%%%%%
```

Danie Els is now (2005) the official maintainer of the booktabs package. He provided me with some updates to synchronize memoir, booktabs and support for colortbl with booktabs. I made the appropriate changes/extensions on 2005/09/03 as part of version 1.618 of memoir.

\CT@arc@ This supports the bookhands/colortbl package combination.

```
\AtBeginDocument{%
  \providecommand*{\CT@arc@}{}}
```

\heavyrulewidth
\lightrulewidth
\cmidrulewidth
\belowrulesep
\belowbottomsep
\aboverulesep
\abovetopsep
\cmidrulesep
\cmidrulekern
\defaultaddspace

First we set up some new dimensions.

```
\newdimen\heavyrulewidth
\newdimen\lightrulewidth
\newdimen\cmidrulewidth
\newdimen\belowrulesep
\newdimen\belowbottomsep
\newdimen\aboverulesep
\newdimen\abovetopsep
\newdimen\cmidrulesep
\newdimen\cmidrulekern
\newdimen\defaultaddspace
\heavyrulewidth=.08em
\lightrulewidth=.05em
\cmidrulewidth=.03em
\belowrulesep=.65ex
\belowbottomsep=\z@
\aboverulesep=.4ex
\abovetopsep=\z@
\cmidrulesep=\doublerulesep
\cmidrulekern=.5em
\defaultaddspace=.5em
```

\@cmidla
\@cmidlb
\@aboverulesep
\@belowrulesep
\@thisruleclass
\@lastruleclass
\@thisrulewidth

And some internal counters and other things of no interest to the end user:

```
\newcount\@cmidla
\newcount\@cmidlb
\newdimen\@aboverulesep
\newdimen\@belowrulesep
\newcount\@thisruleclass
\newcount\@lastruleclass
\@lastruleclass=0
\newdimen\@thisrulewidth
```

which will be described as needed below.

\futurenonspacelet
\@BTfnslone
\@BTfnsltwo
\@BTfnslthree

Next we define a very useful macro (more-or-less straight from the *The TEXbook*'s Dirty Tricks chapter; documented there). Use `\futurenonspacelet` instead of `\futurelet` when looking for the next (non-space) token after a macro that has an argument. (After a macro without an argument, space is ignored anyway, so `\futurenonspacelet` wouldn't be needed.) This hack allows users to type white space between successive rule commands.

```
\def\futurenonspacelet#1{\def\@BTcs{#1}%
 \afterassignment\@BTfnslone\let\nexttoken= }
\def\@BTfnslone{\expandafter\futurelet\@BTcs\@BTfnsltwo}
\def\@BTfnsltwo{\expandafter\ifx\@BTcs\@sptoken\let\next=\@BTfnslthree
 \else\let\next=\nexttoken\fi \next}
\def\@BTfnslthree{\afterassignment\@BTfnslone\let\next= }
```

17.6.1 Full width rules

When we are not in a `longtable` environment, we can simply implement the full width rules as a `\hrule` in a `\noalign{}` group. But within a `longtable`, the rule has to be drawn like a `\cmidrule{1-\LT@cols}` (the rationale for this is explained in the `longtable` documentation).
In order to allow for both, all the rule macros have to open a `\noalign` group immediately, while they work out whether they have been called within a `longtable`; if you don't do this, TEX's underlying `\halign` process gets hiccups. I use LATEX's dirty trick (`\ifnum=0‘}`) to fool the parser that the bracket count is OK. The bracket really gets closed after all the skipping at the end of the `\@BTendrule` macro.
The class 1 rules, and `\specialrule`, really only differ in the defaults for space above and below, and the width, passed to a common routine, `\@BTrule`, described below. The spaces, `\@aboverulesep` and `\@belowrulesep`, are set within the `\noalign` group, so are inherited by `\@BTrule`. Similarly, `\@BTrule` knows as much as it needs to about the routine that called it by examining the inherited `\@thisruleclass`. The optional width argument is parsed by `\@BTrule` after being set to default if absent.

`\toprule`
`\midrule`
`\bottomrule`
`\specialrule`

```
\def\toprule{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=\abovetopsep
  \global\@belowrulesep=\belowrulesep
  \global\@thisruleclass=\@ne
  \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}

\def\midrule{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=\aboverulesep
  \global\@belowrulesep=\belowrulesep
  \global\@thisruleclass=\@ne
  \@ifnextchar[{\@BTrule}{\@BTrule[\lightrulewidth]}}

\def\bottomrule{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=\aboverulesep
  \global\@belowrulesep=\belowbottomsep
  \global\@thisruleclass=\@ne
  \@ifnextchar[{\@BTrule}{\@BTrule[\heavyrulewidth]}}

\def\specialrule#1#2#3{\noalign{\ifnum0=‘}\fi
  \@aboverulesep=#2\global\@belowrulesep=#3\global\@thisruleclass=\tw@
  \@BTrule[#1]}
```

`\addlinespace` An `\addlinespace` is essentially a zero-width rule with zero space above and argument (or default) space below. But because the rule is not actually drawn, but is just a `\vskip`, there is no need to check if we're in a longtable, so we don't need to call `\@BTrule` as for 'real' rules. But we do share the `\@BTendrule`

lookahead and flagsetting code (described below), and the \vskip is done there.

```
\def\addlinespace{\noalign{\ifnum0=`}\fi
  \@ifnextchar[{\@addspace}{\@addspace[\defaultaddspace]}}
\def\@addspace[#1]{\global\@belowrulesep=#1\global\@thisruleclass=\tw@
  \futurelet\@tempa\@BTendrule}
```

\@BTrule All the rules (except \addlinespace) share this code.

```
\def\@BTrule[#1]{%
  \global\@thisrulewidth=#1\relax
```

Save the width argument (if the user didn't give one, then the calling routine will have called \@BTrule with the default) in a global variable for later use when drawing the rule.

```
  \ifnum\@thisruleclass=\tw@\vskip\@aboverulesep\else
```

Specialrules always insert specified space above. (Note: addlinespaces don't come here).

```
  \ifnum\@lastruleclass=\z@\vskip\@aboverulesep\else
  \ifnum\@lastruleclass=\@ne\vskip\doublerulesep\fi\fi\fi
```

After text (last rule class 0), precede the rule by \aboverulesep; but if immediately after a previous rule, insert a \doublerulesep.
Now we work out, by a very nasty hack, if we're within a longtable. It's easy if \longtable isn't even defined: then we can't be. But it is not enough just to check if longtable is loaded — we might be within an ordinary table rather than a longtable. So we look to see if \hline has been re-defined from its LaTeX definition to be the same as \LT@hline. (Longtable currently does this redefinition when it opens a longtable environment, but not globally, so it is cleared it when the environment closes.) Another package could potentially do this! And longtable might change the way it implements this! So, it is not entirely safe, but I have found no better way so far.
We set up \@BTswitch to call \@BTnormal or \@BLTrule, as appropriate, then call it.

```
  \ifx\longtable\undefined\let\@BTswitch\@BTnormal\else
  \ifx\hline\LT@hline\let\@BTswitch\@BLTrule
  \else\let\@BTswitch\@BTnormal\fi\fi
  \@BTswitch}
```

\@BTnormal This is when we're *not* within a longtable. We are already in a \noalign group, all we need do is draw an \hrule and gobble any trailing spaces, then call the closing routine with \@tempa set equal to the next token in the document.

```
% \def\@BTnormal{\hrule
%   \@height \@thisrulewidth\futurenonspacelet\@tempa\@BTendrule}
\def\@BTnormal{%
%%  \bktabrule{\@thisrulewidth}
  {\CT@arc@\bktabrule{\@thisrulewidth}}%
  \futurenonspacelet\@tempa\@BTendrule}
```

\@BLTrule This is for full width rule within a longtable. First we check if a kerning argument has been used; if so let \@@BLTrule read it, else call \@@BLTrule with an empty string:

```
\def\@BLTrule{\@ifnextchar({\@@BLTrule}{\@@BLTrule()}}
```

\@@BLTrule

```
\def\@@BLTrule(#1){\@setrulekerning{#1}%
\global\@cmidlb\LT@cols
```

The \@setrulekerning routine parses the kerning argument tokens and sets global kerning widths accordingly (or to defaults, if user hasn't set them explicitly). The global assignment to \@cmidlb sets up the column count for the \@cmidruleb macro, which is shared with cmidrules.

```
\ifnum0=‘{\fi}%
```

Close the currently open \noalign group. Within a longtable, rules are all to be drawn as leaders within a text box that is \LT@cols columns wide.

```
\@cmidruleb
```

Draw the rule. We share the \@cmidruleb code with ordinary \cmidrules.

```
\noalign{\ifnum0=‘}\fi
```

We have to open a new noalign immediately else TEX will start a new text box where we don't want one. Then, after gobbling any unwanted white space, we call the closing routine.

```
\futurenonspacelet\@tempa\@BTendrule}

```

\@BTendrule We look one step ahead (token is in \@tempa) to see if another rule follows (shame on user!). If so, we set \@lastruleclass equal to \@thisruleclass (thus setting it up for the following rule). If there isn't a following rule, we clear \@lastruleclass (ie set it to zero), which isn't technically true since we have just drawn a rule, but sets it up correctly for the next rule encountered, which must be following some intervening text.

```
\def\@BTendrule{%
  \ifx\@tempa\toprule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\midrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\bottomrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\cmidrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\specialrule\global\@lastruleclass=\@thisruleclass
  \else\ifx\@tempa\addlinespace\global\@lastruleclass=\@thisruleclass
  \else\global\@lastruleclass=\z@\fi\fi\fi\fi\fi
  \ifnum\@lastruleclass=\@ne\relax\else\vskip\@belowrulesep\fi
  \ifnum0=‘{\fi}}
```

17.6.2 Special subrules

\@setrulekerning The following code parses the trimming arguments (if there are any) for \cmidrule or a \BLTrule. The rule will be trimmed left and right by

\cmrkern@l and \cmrkern@l, which are zero by default, set to \cmidrulekern by the plain (lr) arguments, or user set as in (r{.5em}). We parse token by token through the arguments. The tokens r and l cause \cmrkern@r or \cmrkern@l to be set to \cmidrulekern. There is no lookahead to see if a width is the next token; this strategy is efficient for the plain commands, while inefficient for the qualified commands, but more importantly it is much easier to program. Tokens r and l also set \cmrswitch so that if the next token turns out to be {⟨*wd*⟩} then the kerning will be done on the side currently specified. I have been too lazy to program an error message should one encounter tokens other than r, l or {⟨*wd*⟩}.

```
\def\@setrulekerning#1{%
  \global\let\cmrkern@l\z@
  \global\let\cmrkern@r\z@
  \@tfor\@tempa :=#1\do
  {\def\@tempb{r}%
   \ifx\@tempa\@tempb
     \global\let\cmrkern@r\cmidrulekern
      \def\cmrsideswitch{\cmrkern@r}%
   \else
     \def\@tempb{l}%
     \ifx\@tempa\@tempb
       \global\let\cmrkern@l\cmidrulekern
       \def\cmrsideswitch{\cmrkern@l}%
     \else
       \global\expandafter\let\cmrsideswitch\@tempa
     \fi
   \fi}}

```

\cmidrule
\@cmidrule
\@@cmidrule

The \cmidrule re-uses \@lastruleclass in an entirely different way from the full width rules. (Maybe I should have used a different flag; it seemed efficient at the time . . .). This is (left) set to one if you are in the middle of a row of \cmidrules, or starting a new one (with \morecmidrules). Otherwise, when \@lastruleclass is zero, we precede the rule with \aboverulesep.

```
\def\cmidrule{\noalign{\ifnum0=`}\fi
    \@ifnextchar[{\@cmidrule}{\@cmidrule[\cmidrulewidth]}}
\def\@cmidrule[#1]{\@ifnextchar({\@@cmidrule[#1]}{\@@cmidrule[#1]()}}
\def\@@cmidrule[#1](#2)#3{\@@@cmidrule[#3]{#1}{#2}}

```

\@@@cmidrule

The above is fiddling around to set defaults for missing optional arguments. We also pass to \@@@cmidrule in a different order, namely =[a-b]=**{width required}=** \verbkerning commands= (this being the order in which the arguments are actually processed):

```
\def\@@@cmidrule[#1-#2]#3#4{\global\@cmidla#1\relax
    \global\advance\@cmidla\m@ne
    \ifnum\@cmidla>0\global\let\@gtempa\@cmidrulea\else
    \global\let\@gtempa\@cmidruleb\fi
```

```
    \global\@cmidlb#2\relax
    \global\advance\@cmidlb-\@cmidla
```

This has set up a switch (`\@gtempa`) to call the relevant routine, `\@cmidrulea` or `\@cmidruleb`, depending on whether we start from column one or not.

```
    \global\@thisrulewidth=#3
```

That is, set per default or given argument. Then parse any trimming arguments to set, globally, `\cmrkern@r` and `\cmrkern@l` accordingly:

```
    \@setrulekerning{#4}
```

Now insert space above if needed, close the `\noalign`, then switch to appropriate rule drawing routine as determined above (`\let` to `\@gtempa`):

```
    \ifnum\@lastruleclass=\z@\vskip \aboverulesep\fi
    \ifnum0=`{\fi}\@gtempa
```

Having now drawn the rule, open another `\noalign`, and call the closing routine:

```
    \noalign{\ifnum0=`}\fi\futurenonspacelet\@tempa\@xcmidrule}
```

`\@xcmidrule` In this closing routine, see if another `\cmidrule` follows; if so, backspace vertical so it will line up with the one you just drew, and setting `\@lastruleclass` to 1 will suppress adding space above the next. If a `\morecimdrules` follows, we add (positive) `\cmidrulesep` (and again set `\@lastruleclass` to one). Otherwise this is the last rule of the current group and we can just add `\belowrulesep`. Finally, we close the `\noalign`.

```
\def\@xcmidrule{\ifx\@tempa\cmidrule\vskip-\@thisrulewidth
    \global\@lastruleclass=\@ne\else
    \ifx\@tempa\morecmidrules\vskip \cmidrulesep
    \global\@lastruleclass=\@ne\else
    \vskip \belowrulesep\global\@lastruleclass=\z@\fi\fi
    \ifnum0=`{\fi}}

```

`\@cmidrulea` `\@cmidruleb` This code (called below) actually draws the rules. They are drawn as boxes in text, rather than in a `\noalign` group, which permits the left and right kerning.

```
\def\@cmidrulea{%
  \multispan\@cmidla&\multispan\@cmidlb
%%    \unskip\hskip \cmrkern@l\leaders\bktabrule{\@thisrulewidth}\hfill
    \unskip\hskip \cmrkern@l{%
    \CT@arc@\leaders\bktabrule{\@thisrulewidth}\hfill}%
    \hskip \cmrkern@r\cr}
\def\@cmidruleb{%
  \multispan\@cmidlb
%%    \unskip\hskip \cmrkern@l\leaders\bktabrule{\@thisrulewidth}\hfill
  \unskip\hskip \cmrkern@l{%
    \CT@arc@\leaders\bktabrule{\@thisrulewidth}\hfill}%
    \hskip \cmrkern@r\cr}

```

`\morecmidrules` This is really a dummy command; all the work is done above within the `\cmidrule` routine. We look one step ahead there to see if a `\morecmidrules`

follows the current \cmidrule, and if so set the flag. Otherwise, \morecmidrules itself does nothing.

```
\def\morecmidrules{\noalign{\relax}}

%%%%%%%%%% end of Booktabs package code %%%%%%%%%%%%%%%%%%%%
```

Ensure that the real booktabs package can't get called.

```
%%\@memfakeusepackage{booktabs}

```

17.7 Input files into tabulars, and etex

M.J. Williams (CTT *\noalign problem with \input and tabular in memoir class*, 28 Aug 2007) reported that using \input in a tabular resulted in errors. His example was:

```
...
\begin{tabular}{c|c} \hline
1 & 2 \\ \hline
\input{data} \hline
5 & 6 \\ \hline
\end{tabular}
```

where data.tex contains the single line (and no newline)
3 & 4 \\
Morten Høgholm noted that the problem was caused by memoir's extension to \input and gave a solution that depends on processing via etex rather than tex. Here's Morten's fix. Quoting, in part:

> memoir extends the file loading mechanism by providing \AtBeginFile and \AtEndFile whose arguments are executed as the names indicate. ... after inputting a file the storage bins are emptied. The problem [in your case] is that this emptying is an assignment and so starts a new tabular cell and \hline is only allowed after \\ or other \hlines.
>
> ... the tabular environment wraps each tabular cell in additional groups so one can do an explicit test for which type of group we are in. If tested to be in an align group *and* in vertical mode it means TeX hasn't found something to start a new cell and then \noalign can be used.

```
\ifetex
  \renewcommand*{\killm@matf}[1]{%
    \ifnum 6=\currentgrouptype
      \ifvmode
        \expandafter\expandafter\expandafter\@firstoftwo
        \expandafter\expandafter\expandafter\noalign
```

```
      \fi
    \fi
    \@firstofone
    {\@namelet{#1-m@mfb}\relax
     \@namelet{#1-m@mfe}\relax
    }%
  }
\fi

```

17.8 Continuous tabulars

The standard `tabular` environment is wrapped inside a box (see `\@tabular`), which means that it cannot break across pages. Equivalent environments are provided which are not boxed and so can continue across page boundaries.

`\ctableftskip` `\ctabrightskip` These skips are inserted at the left and right of continuous tabulars so that their horizontal location can be specified.

```
\newskip\ctableftskip \ctableftskip=\fill
\newskip\ctabrightskip \ctabrightskip=\fill

```

`\ctabular*` `\@ctabularstar` This starts off the `ctabular*` continuous tabular environment. The default location is centered. NOTE: This is not for release.

```
\expandafter\def\csname ctabular*\endcsname{%
  \@ifnextchar[ {\@ctabularstar}{\@ctabularstar[c]}}
\def\@ctabularstar[#1]#2{\global\@curtab\@ne
  \ctableftskip\fill
  \ctabrightskip\fill
  \if l#1% left
    \ctableftskip\z@
  \else
    \if r#1% right
       \ctabrightskip\z@
    \fi
  \fi
  \setlength\dimen@{#2}%
  \xdef\@halignto{to\the\dimen@}\NC@tabular}

```

`\ctabular` `\ctabular` is similar to `\ctabular*` except that the width is known to be `\hsize`.

```
\newcommand*{\ctabular}[1][c]{\global\@curtab\@ne
  \ctableftskip\fill
  \ctabrightskip\fill
  \if l#1% left
    \ctableftskip\z@
  \else
    \if r#1% right
      \ctabrightskip\z@
```

```
    \fi
  \fi
  \gdef\@halignto{to\hsize}\NC@tabular}

\NC@tabular  \NC@tabular
\newcommand*{\NC@tabular}{%
  \par
  \addvspace{\topsep}
  \col@sep\tabcolsep
  \let\d@llarbegin\begingroup
  \let\d@llarend\endgroup
  \@NCtabarray}

 \@NCialign
\newcommand*{\@NCialign}{\everycr{}\tabskip\ctableftskip\halign}

\@NCtabarray  Effectively a modified version of \@array
\newcommand*{\@NCtabarray}[1]{%
  \@tempdima \ht\strutbox
  \advance\@tempdima\extrarowheight
  \setbox \@arstrutbox \hbox{\vrule
    \@height \arraystretch \@tempdima
    \@depth \arraystretch \dp\strutbox
    \@width\z@}%
  \begingroup
%%    \@mkpream{@{\hspace{\@totalleftmargin}}#1@{}}%
    \@mkpream{#1}%
    \xdef\@preamble{\@NCialign \@halignto
                    \bgroup & \tabskip\z@
                      \@arstrut
                      \@preamble
                      \tabskip\ctabrightskip
                      \cr}%
  \endgroup
  \let\@sharp ##\let\protect\relax
  \lineskip\z@
  \baselineskip\z@
  \let\\\@arraycr
  \let\tabularnewline\\%
  \let\par\@empty
  \ctabsetlines
  \@preamble
}

\endctabular  End the environments.
\endctabular*
```

```
\def\endctabular{%
  \crcr \egroup
  \gdef\@preamble{}%
  \addvspace{\topsep}
  \noindent}
\expandafter\let\csname endctabular*\endcsname=\endctabular

```

17.8.1 Horizontal lines

The standard \hline can produce odd results when used with continuous tabulars, but \hhline or \cline are OK. I want a version of \hline that works with continuous tabulars and has an adjustable width (i.e., thickness). This will be based on a modified \cline
For reference, here are the essentials of the kernel's definitions.

```
\def\hline{%
   \noalign{\ifnum0=`}\fi\hrule \@height \arrayrulewidth \futurelet
   \reserved@a\@xhline}
\def\@xhline{%
   \ifx\reserved@a\hline
     \vskip\doublerulesep
     \vskip-\arrayrulewidth
   \fi
   \ifnum0=`{\fi}}
\def\cline#1{\@cline#1\@nil}
\def\@cline#1-#2\@nil{%
  \omit
  \@multicnt#1%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#2%
  \advance\@multicnt-#1%
  \advance\@multispan\@ne
  \leaders\hrule\@height\arrayrulewidth\hfill
  \cr
  \noalign{\vskip-\arrayrulewidth}}
```

\memcline This is a version of the kernel \cline code that provides a variable width.

\m@m@cline

```
\newcommand*{\memcline}[2]{\m@m@cline[#1]#2\@nil}
\def\m@m@cline[#1]#2-#3\@nil{%
  \omit
  \@multicnt#2%
  \advance\@multispan\m@ne
  \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi
  \@multicnt#3%
  \advance\@multicnt-#2%
  \advance\@multispan\@ne
  \leaders\hrule\@height #1\hfill  % <- variable \@height value
```

```
  \cr
  \noalign{\vskip- #1}}            % <- variable \@height value
```

\memhline
\m@mhline
A version of \hline, based on \memcline, taking a rule width as argument. It draws a \memcline from the first to the last (\@curtab) column.

```
\newcommand*{\memhline}[1][\arrayrulewidth]{\memcline{#1}{1-\@curtab}}
\newcommand*{\m@mhline}{\cline{1-\@curtab}}
```

Fear's rules also do not play well within continuous tabulars.

\m@m@BTnormal
Special version of \@BTnormal

```
\def\m@m@BTnormal{%
  \ifnum0=`{\fi}   % closes the \noalign
  \multispan{\@curtab} \leaders\bktabrule{\@thisrulewidth}\hfill\cr
  \noalign{\ifnum0=`}\fi
  \futurenonspacelet\@tempa\@BTendrule}
```

\ctabsetlines
Continuous tabulars cannot use the standard horizontal lines.

```
\def\ctabsetlines{%
  \let\hline\m@mhline
  \let\@BTnormal\m@m@BTnormal}
```

17.9 Automated tabulations

It can be convenient, especially while drafting a document, to have a list of items put into a tabular without having to mark the ends of the rows.
The following is based on plain TEX code given on pages 307–308 of *TeX for the Impatient* [ABH90], which provides code for typesetting in columns.

\abovecolumnspenalty
\@linestogo
\@cellstogo
\@cellsincolumn
\crtok
Counters and such.

```
\newcount\abovecolumnspenalty
  \abovecolumnspenalty=10000
\newcount\@linestogo           % lines remaining to be procesed
\newcount\@cellstogo           % cells remaining in column or row
\newcount\@cellsincolumn         % number of lines per column
\newtoks\crtok
  \crtok = {\cr}%
```

\@mincolumnwidth
\c@lleftskip
\c@lrightskip
More things

```
\newdimen\@mincolumnwidth
\let\c@lleftskip\hfil   % left skip within a column
\let\c@lrightskip\hfil % right skip within a column
```

\preautotab
\postautotab
Hooks into the auto tabulations.

```
\let\preautotab\relax
\let\postautotab\relax
```

\autocols
`\autocols[`⟨*width*⟩`]{`⟨*pos*⟩`}{`⟨*num*⟩`}{`⟨*style*⟩`}{`⟨*comma separated list*⟩`}` arranges the elements in the ⟨*comma separated list*⟩ into ⟨*num*⟩ columns, the elements filling each column before moving to the next. That is, the elements are ordered top to bottom and left to right.

```
\newcommand{\autocols}[5][0pt]{\par\begingroup
  \ctabsetlines
```

Set the table position

```
  \if l#2
    \raggedright
  \else
    \if r#2
      \raggedleft
    \else
      \centering
    \fi
  \fi
```

Set the column position style

```
  \let\c@llefskip\hfil
  \let\c@lrightskip\hfil
  \if l#4
    \let\c@llefskip\relax
  \else
    \if r#4
      \let\c@lrightskip\relax
    \fi
  \fi
```

Count the number of entries and the minimum width (max entry width) for the columns.

```
  \@mincolumnwidth\z@
  \TX@cols=#3
  \@curtab=#3
  \@linestogo\z@
  \@for\@tempa:=#5\do{
    \advance\@linestogo\@ne
    \settowidth{\@tempdima}{\@tempa}
    \ifdim\@tempdima>\@mincolumnwidth
      \@mincolumnwidth=\@tempdima
    \fi
  }
  \advance\@mincolumnwidth\tabcolsep
  \linespercol
```

Specify what is to be done after every entry

```
\def\@endcolumnactions{%
  \global\advance\@linestogo\m@ne
  \ifnum\@cellstogo<\tw@
    \global\advance\TX@cols\m@ne
    \ifnum\TX@cols>\z@\linespercol\fi
    \the\crtok
  \else
      &\global\advance\@cellstogo\m@ne
  \fi}%
```

Calculate the width of the columns

```
\ifdim #1 > \z@
  \TX@col@width=#1
  \divide\TX@col@width \TX@cols
\else
  \TX@col@width=\@mincolumnwidth
\fi
\penalty\abovecolumnspenalty
\noindent% usually not a paragraph
```

Create most of the preamble by looping to add `\@cellsincolumn`-1 slots, then the last one which is different.

```
\def\@preamble{}%
\begingroup
  \let\@sharp\relax
  \ifnum\@cellsincolumn>\@ne
    \loop
      \g@addto@macro{\@preamble}{%
        \hb@xt@ \TX@col@width{%
             \c@lleftskip\strut\@sharp\c@lrightskip} &}%
      \advance\@cellsincolumn\m@ne
    \ifnum\@cellsincolumn>\@ne
    \repeat
  \fi
  \g@addto@macro{\@preamble}{%
    \hb@xt@ \TX@col@width{\c@lleftskip\strut\@sharp\c@lrightskip}}%
\endgroup
\let\@sharp ##
```

Start the `\valign`

```
  \tabskip\ctableftskip
%%  \tabskip\z@
  \valign \bgroup
    \tabskip\z@
    \@preamble
    \tabskip\ctabrightskip\cr
```

Add all the entries then finish off.

```
    \@for\@tempa:=#5\do{
      \@tempa\unskip\space\@endcolumnactions}%
```

```
6908     \the\crtok \egroup \par \endgroup}
6909
```

\linespercol `\linespercol` calculates the maximum number of lines that go into a column, where there are `\TX@cols` columns and `\@linestogo` lines, so that the columns are balanced as well as possible. The result is `\@cellstogo`

```
6910 \newcommand*{\linespercol}{%
6911   \@cellsincolumn=\@linestogo
6912   \divide\@cellsincolumn \TX@cols
6913   \@cellstogo=\@cellsincolumn
6914   \multiply\@cellstogo \TX@cols
6915   \@tempcnta=\@linestogo
6916   \advance\@tempcnta -\@cellstogo
6917   \ifnum \@tempcnta>\z@
6918     \advance\@cellsincolumn \@ne
6919   \fi
6920   \global\@cellstogo=\@cellsincolumn}
6921
```

\autorows `\autorows[`⟨*width*⟩`]{`⟨*pos*⟩`}{`⟨*num*⟩`}{`⟨*style*⟩`}{`⟨*comma separated list*⟩`}` arranges the elements in the ⟨*comma separated list*⟩ into ⟨*num*⟩ columns, the elements filling each row before moving to the next. That is, the elements are ordered left to right and top to bottom. By default, each column is the same width, enough for the widest entry. If ⟨*width*⟩ is a negative length (e.g., -1pt) the columns are set to their natural widths. If ⟨*width*⟩ is positive (e.g., `\textwidth`), column widths are equal widths so that the overal width is ⟨*width*⟩. The sideways location of thre tabular is ⟨*pos*⟩ (l, c, or r), and the column style id ⟨*style*⟩ (l, c, or r).

```
6922 \newcommand{\autorows}[5][0pt]{\par\begingroup
6923  \ctabsetlines
```

Set the table position

```
6924   \ctableftskip\fill
6925   \ctabrightskip\fill
6926   \if l#2
6927     \ctableftskip\z@
6928   \else
6929     \if r#2
6930       \ctabrightskip\z@
6931     \fi
6932   \fi
```

Set the column position style

```
6933   \let\c@llefttskip\hfil
6934   \let\c@lrightskip\hfil
6935   \if l#4
6936     \let\c@llefttskip\relax
6937   \else
6938     \if r#4
```

```
      \let\c@lrightskip\relax
    \fi
  \fi
```

Count the number of entries and the minimum width (max entry width) for the columns.

```
  \TX@cols=#3\relax
  \@curtab=#3\relax
  \@cellstogo = \TX@cols
  \@mincolumnwidth\z@
  \@linestogo\z@
  \@for\@tempa:=#5\do{%
    \advance\@linestogo\@ne
    \settowidth{\@tempdima}{\@tempa}
    \ifdim\@tempdima>\@mincolumnwidth
      \@mincolumnwidth=\@tempdima
    \fi}%
  \advance\@mincolumnwidth\tabcolsep
```

Specify what is to be done after every entry

```
  \def\@endcolumnactions{%
    \global\advance\@linestogo\m@ne
    \global\advance\@cellstogo\m@ne
    \ifnum\@cellstogo<\@ne
      \global\@cellstogo=\TX@cols
      \the\crtok
    \else
      &
    \fi}%
```

Calculate the width of the columns

```
  \ifdim #1>\z@
    \TX@col@width=#1
  \else
    \TX@col@width=\hsize
  \fi
  \divide\TX@col@width \TX@cols
  \ifdim #1=\z@
    \TX@col@width=\@mincolumnwidth
  \fi
  \penalty\abovecolumnspenalty
  \noindent % usually not a paragraph
  \vskip -\z@ % don't know why we need this, but looks bad without it
```

Create most of the preamble by looping to add `\@cellsincolumn`-1 slots, then the last one which is different.

```
  \def\@preamble{}%
  \begingroup
    \let\@sharp\relax
    \ifnum\TX@cols>\@ne
      \loop
```

```
        \ifdim #1<\z@
          \g@addto@macro{\@preamble}{%
            \strut\c@lleftskip\@sharp\c@lrightskip &}%
        \else
          \g@addto@macro{\@preamble}{%
            \hb@xt@ \TX@col@width{%
                    \strut\c@lleftskip\@sharp\c@lrightskip} &}%
        \fi
        \advance\TX@cols\m@ne
      \ifnum\TX@cols>\@ne
      \repeat
    \fi
    \ifdim #1<\z@
      \g@addto@macro{\@preamble}{%
        \strut\c@lleftskip\@sharp\c@lrightskip}%
    \else
      \g@addto@macro{\@preamble}{%
        \hb@xt@ \TX@col@width{\strut\c@lleftskip\@sharp\c@lrightskip}}%
    \fi
  \endgroup
  \let\@sharp ##
```

Start the \halign

```
  \tabskip\ctableftskip
  \halign to \hsize \bgroup
    \tabskip\z@
    \@preamble
%%    \tabskip\ctabrightskip\cr \preautotab
    \tabskip\ctabrightskip\cr
```

Add all the entries then finish off.

```
    \@for\@tempa:=#5\do{%
      \@tempa\unskip\space\@endcolumnactions}%
%%    \the\crtok \postautotab \the\crtok \egroup \endgroup \par
    \the\crtok \egroup \endgroup \par}

```

18 Floating objects

The file `latex.dtx` only defines a number of tools with which floating objects can be defined. This is done in the document class. It needs to define the following macros for each floating object of type `TYPE` (e.g., `TYPE` = figure).

`\fps@TYPE` The default placement specifier for floats of type `TYPE`.

`\ftype@TYPE` The type number for floats of type `TYPE`. Each `TYPE` has associated a unique positive `TYPE` number, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

`\ext@TYPE` The file extension indicating the file on which the contents list for float type `TYPE` is stored. For example, `\ext@figure` = 'lof'.

`\fnum@TYPE` A macro to generate the figure number for a caption. For example, `\fnum@TYPE` == 'Figure `\thefigure`'.

`\@makecaption`⟨***num***⟩⟨***text***⟩ A macro to make a caption, with ⟨*num*⟩ the value produced by `\fnum@...` and ⟨*text*⟩ the text of the caption. It can assume it's in a `\parbox` of the appropriate width. This will be used for *all* floating objects.

The actual environment that implements a floating object such as a figure is defined using the macros `\@float` and `\end@float`, which are defined in `latex.dtx`.
An environment that implements a single column floating object is started with `\@float{TYPE}[`⟨*placement*⟩`]` of type `TYPE` with ⟨*placement*⟩ as the placement specifier. The default value of ⟨*PLACEMENT*⟩ is defined by `\fps@TYPE`.
The environment is ended by `\end@float`. E.g., `\figure` == `\@float{figure}`, `\endfigure` == `\end@float`.

18.1 Floats

To define a float environment, say `fenv`, the following macros must be defined:

- `\fps@fenv` The default placement specifier (normally `tbp`).
- `\ftype@fenv` The type number which is an integer and a power of 2.
- `\ext@fenv` The file extension for the contents list.
- `\c@fenv` A counter for the environment (for caption numbering).
- `\fnum@fenv` A macro to generate the caption 'number'.
- `\l@fenv` A macro to produce an entry in a list of....
- `\flegtocfenv` A macro to write a `\namedlegend` title to a listof file.
- `\flegfenv` A macro to typeset the name of a `\namedlegend`.
- `\toclevel@fenv` Holding a bookmark level (required if the hyperref package will be used).

`newflo@tctr` A counter for the type number of a new float. Normally figures are of type 1, tables type 2, and the next float type is then 4, and so on.

```
\newcounter{newflo@tctr}
  \setcounter{newflo@tctr}{1}
```

\newfloat `\newfloat`oargwithin{⟨*fenv*⟩}{⟨*ext*⟩}{⟨*capname*⟩} creates the commands for a new float environment, ⟨*fenv*⟩ (aka `X`), using ⟨*ext*⟩ (aka `Z`) as the file extension and ⟨*capname*⟩ for the caption name.

```
\newcommand{\newfloat}[4][\@empty]{%
```

\ftype@X Define the float type, set it to the float counter, and double the counter afterwards.

```
%%%  \@namedef{ftype@#2}{\value{newflo@tctr}}
%%%  \addtocounter{newflo@tctr}{\value{newflo@tctr}}
  \expandafter\edef\csname ftype@#2\endcsname{\the\c@newflo@tctr}%
  \advance\c@newflo@tctr \c@newflo@tctr
```

\c@X Create the counter for the caption, which must not have been previously defined.

```
  \@ifundefined{c@#2}{% counter is not defined
    \ifx \@empty#1\relax
      \newcounter{#2}
    \else
      \newcounter{#2}[#1]
      \expandafter\edef\csname the#2\endcsname{%
  \expandafter\noexpand\csname the#1\endcsname.\noexpand\arabic{#2}}
    \fi}{}
  \setcounter{#2}{0}

```

\ext@X Define `\ext@X` for the file extension and set the new `Zdepth` depth counter to 1.

\c@Zdepth

```
  \@namedef{ext@#2}{#3}%     file extension
  \@ifundefined{c@#3depth}{\newcounter{#3depth}}{}
  \setcounter{#3depth}{1}

```

\fps@X \fnum@X \flegX \flegtocX `\fps@X` is the default float placement specification, `\fnum@X` typesets the caption name and number, and `\flegX` and `\flegtocX` are for named legends.

```
  \@namedef{fps@#2}{tbp}                    % position
  \@namedef{fnum@#2}{#4~\@nameuse{the#2}}   % caption naming
  \@namedef{fleg#2}{#4}                     % legend naming
  \@namedef{flegtoc#2}##1{}                 % legend name in ToC

```

X X* Finally define the new float environment, in both normal and starred forms.

```
  \newenvironment{#2}{\@float{#2}}{\end@float}
  \newenvironment{#2*}{\@dblfloat{#2}}{\end@dblfloat}
```

This ends the definition of `\newfloat`.

```
} % end \newfloat

```

\setfloatlocations `\setfloatlocations`{⟨*float*⟩}{⟨*locs*⟩} sets the defualt locations for the ⟨*float*⟩ class of floats (e.g., `figure`) to ⟨*locs*⟩ (initially `tbp`). For tables you might want to use:

\setfloatlocations{table}{htbp}

```
\newcommand*{\setfloatlocations}[2]{\@namedef{fps@#1}{#2}}
```

To define subcaptions for use in a new float environment, say `fenv`, the following macros must be defined [Coc02]:

- A new counter `subfenv` for subcaption numbering.
- A new counter `extdepth`, where `ext` is the file extension for the contents list of `fenv`, for setting the contents depth.
- `\thesubfenv` for the formatting of the subcaption number.
- `\@thesubfenv` for typesetting the number.
- `\@@thesubfenv` for alternative label reference.
- `\p@subfenv` for prepending to the subcaption number when it is referenced.
- `\ext@subfenv` the file extension for the contents list.
- `\l@subfenv` for formatting the contents list entry.
- `\@makesubfloatcaption` for typesetting the subcaption.
- `\toclevel@subfenv` for hyperref bookmarks

`\newsubfloat` `\newsubfloat{⟨fenv⟩}` creates the commands for a new subfloat for ⟨*fenv*⟩ (aka `X`).

```
\newcommand{\newsubfloat}[1]{%
```

Call `\newlistentry[X]{subX}{extX}{1}` to get most of the work done.

```
  \newlistentry[#1]{sub#1}{\@nameuse{ext@#1}}{1}
```

`\ext@subX` And now for the rest of the commands for subcaptions.

`\thesubX` `\@thesubX` `\@@thesubX` `\p@subX` `\toclevel@subX`

```
  \@namedef{ext@sub#1}{\csname ext@#1\endcsname}
  \@namedef{thesub#1}{(\alph{sub#1})}
  \@namedef{@thesub#1}{\@nameuse{thesub#1}%
            \if@tightsubcap\hskip\subfloatlabelskip\else\space\fi}
  \@namedef{@@thesub#1}{\@nameuse{thesub#1}}
  \@namedef{p@sub#1}{\csname the#1\endcsname}
  \@namedef{toclevel@sub#1}{1}}
```

By permission of Steven Douglas Cochran the class provides similar functionality as the subfigure package [Coc02]. This requires some changes to be made to `\@float` and `\end@float`.

`\ifdonemaincaption` This is set TRUE after the `\(cont)caption` has been called in a float.

```
\newif\ifdonemaincaption
  \donemaincaptionfalse
```

\@float
\@dbfloat

The kernel \@float and \@dblfloat macros are redefined to set \ifdonemaincaption to FALSE, and also to zero the subfloat counter, if it is defined. Ignasi Furió reported[17] that floats embedded in text created an extra space. This was caused by a missing %.

```
\let\@memoldfloat\@float
\renewcommand{\@float}[1]{\donemaincaptionfalse
  \@ifundefined{c@sub#1}{}{\csname c@sub#1\endcsname = 0\relax}%
  \@memoldfloat{#1}}
\let\@memolddblfloat\@dblfloat
\renewcommand{\@dblfloat}[1]{\donemaincaptionfalse
  \@ifundefined{c@sub#1}{}{\csname c@sub#1\endcsname = 0\relax}%
  \@memolddblfloat{#1}}
```

\end@float
\end@dblfloat

The kernel \end@float and \end@dblfloat macros are redefined to dump out any subcaptions that have not yet been processed.

```
\let\@memoldefloat\end@float
\def\end@float{%
  \@memlistsubcaptions{\@captype}\@memoldefloat}
\let\@memoldedblfloat\end@dblfloat
\def\end@dblfloat{%
  \@memlistsubcaptions{\@captype}\@memoldedblfloat}
```

Unfortunately the fixltx2e package, version 1.1h (current as of 2005/09/03) makes assumptions about \end@float which do not hold for memoir. The code has to be reverted back to that in version 1.0b of fixltx2e.

```
\AtBeginDocument{\@ifpackageloaded{fixltx2e}{%
  \def\end@dblfloat{%
    \if@twocolumn
      \@endfloatbox
      \ifnum\@floatpenalty<\z@
        \@largefloatcheck
        \global\dp\@currbox1sp %
        \@cons\@deferlist\@currbox
      \fi
      \ifnum\@floatpenalty=-\@Mii \@Esphack\fi
    \else
      \end@float
    \fi}}{}}
```

18.2 Captions

The caption styling is accomplished by redefining the \@makecaption command. First, though, define and initialise the user-level commands.

[17] Private email from ignasi.furio@uib.es, 2003/10/17

\if@contcw
\if@conthang
\if@contindent

For use when checking caption width and captioning styles styles.

```
\newif\if@contcw
\newif\if@conthang
\newif\if@contindent
```

\captiondelim
\@contdelim

For the caption delimeter.

```
\newcommand{\captiondelim}[1]{\def\@contdelim{#1}}
\captiondelim{: }
```

\captionnamefont
\@contnfont

The font for the caption name.

```
\newcommand{\captionnamefont}[1]{\def\@contnfont{#1}}
\captionnamefont{}
```

\captiontitlefont
\@conttfont

The font for the caption title.

```
\newcommand{\captiontitlefont}[1]{\def\@conttfont{#1}}
\captiontitlefont{}
```

\captionstyle
\@contcstyle

The paragraphing style for the caption.

```
\newcommand*{\captionstyle}[1]{\def\@contcstyle{#1}}
\captionstyle{}
```

\captionstyle
\@memcshort
\@memcnom
\@contcshortstyle
\@contcstyle

The paragraphing style for the caption.
I had email from Jørgen Larsen (jl@dirac.ruc.dk), 2003/04/09, asking for separate controls for the short and long captions.
\captionstyle[⟨*short*⟩]{⟨*normal*⟩}.

```
\renewcommand{\captionstyle}{%
  \@ifnextchar[ {\@memcshort}{\@memcnorm}}
\def\@memcshort[#1]#2{%
  \def\@contcshortstyle{#1}
  \def\@contcstyle{#2}}
\def\@memcnorm#1{%
  \def\@contcshortstyle{#1}
  \def\@contcstyle{#1}}
\captionstyle{}
```

\@contcwidth
\captionwidth
\changecaptionwidth
\normalcaptionwidth

The macros for dealing with the caption width.

```
\newlength{\@contcwidth}
\newcommand{\captionwidth}[1]{\setlength{\@contcwidth}{#1}}
\captionwidth{\linewidth}
\newcommand{\changecaptionwidth}{\@contcwtrue}
\newcommand{\normalcaptionwidth}{\@contcwfalse}
\normalcaptionwidth
```

\@contindw
\hangcaption
\indentcaption
\normalcaption

The macros for hanging and indented captions.

```
\newlength{\@contindw}
\newcommand{\hangcaption}{\@conthangtrue\@contindentfalse}
\newcommand{\indentcaption}[1]{\setlength{\@contindw}{#1}%
  \@conthangfalse\@contindenttrue}
\newcommand{\normalcaption}{\@conthangfalse\@contindentfalse}
\normalcaption
```

\precaption
\@contpre
\postcaption
\@contpost
\midbicaption
\@contmidbi

The macros for the pre- and post-caption text/commands, and for the mid-caption command for bilingual captions.

```
\newcommand{\precaption}[1]{\def\@contpre{#1}}
\precaption{}
\newcommand{\postcaption}[1]{\def\@contpost{#1}}
\postcaption{}
\newcommand{\midbicaption}[1]{\def\@contmidbi{#1}}
\midbicaption{}
```

\captiontitlefinal

\captiontitlefinal{⟨*stuff*⟩} will put ⟨*stuff*⟩ immediately at the end of a \caption's title text but it will not appear in the LoF/LoT/etc. For example \captiontitlefinal{.}
The code was supplied by Frederic Connes.

```
\newcommand*{\captiontitlefinal}[1]{\def\@contfinal{#1}}
  \captiontitlefinal{}
```

\abovecaptionskip
\belowcaptionskip

Vertical space above and below a caption. Make them sum to an integral number of lines.

```
\newlength{\abovecaptionskip}
  \setlength{\abovecaptionskip}{0.5\onelineskip}
\newlength{\belowcaptionskip}
  \setlength{\belowcaptionskip}{0.5\onelineskip}
```

\caption

For subfloat support, the (kernel) \caption macro needs to note that it has been called.

```
\let\@memoldcaption\caption
\def\caption{\donemaincaptiontrue\@memoldcaption}
```

\memcaptioninfo

\memcaptioninfo{type}{\thetypenum}{short}{long}

```
\newcommand{\memcaptioninfo}[4]{}
```

\@caption

For title referencing support, the (kernel) \@caption macro needs to store the title.

```
\let\@memold@caption\@caption
```

```
\long\def\@caption#1[#2]#3{%
  \M@gettitle{#2}%
  \memcaptioninfo{#1}{\csname the#1\endcsname}{#2}{#3}%
  \@memold@caption{#1}[{#2}]{#3}}
```

`\@makecaption` This is a reimplementation of the kernel `\@makecaption` command. As well as including the caption typesetting commands it enables captions that include forced newlines (e.g., by \\).

The first part is due to Donald Arseneau[18] from postings to the CTT newsgroup and Email discussions. The `\topskip` strut is used whenever the caption is the first part of the float. This means, among other things, that if a caption comes at the top of a page, then the first line of the caption will be aligned with the normal first line of a page. The `\abovecaptionskip` is only used when there is something above the caption in the current float.

```
\long\def\@makecaption#1#2{\let\@memtempa\relax
  \ifdim\prevdepth>-99\p@ \vskip\abovecaptionskip
  \else \def\@memtempa{\vbox to\topskip{}}\fi
```

`\@contfnote` `\@contfmark` The caption title will be typeset twice, firstly to measure its width and secondly to actually typeset it. To avoid problems caused by a footnote in the caption getting processed twice, we temporarily disable the expected relevant commands.

```
  \let\@contfnote\footnote \renewcommand{\footnote}[2][]{}
  \let\@contfmark\footnotemark \renewcommand{\footnotemark}[1][]{}
```

Now measure the width of the total caption, not forgetting to take account of the font specifications, and then restore the footnoting.

```
  \sbox\@tempboxa{\@contnfont #1\@contdelim \@conttfont #2\@contfinal}
  \let\footnote\@contfnote
  \let\footnotemark\@contfmark
```

If the caption is less than one line, then the whole caption needs to be centered on the page (otherwise the short caption may be typeset flushleft).

```
  \ifdim\wd\@tempboxa<\linewidth \centering \fi
  \if@contcw
```

For typesetting at anything other than the normal width, put the caption into a `\parbox` of the specified width. This must be centered.

```
    \centering
    \parbox{\@contcwidth}{%
```

Henrik Holm[19] proposed adding the next line to center short, narrow captions.

```
    \ifdim\wd\@tempboxa<\@contcwidth \centering \fi
  \fi
```

Hanging and indenting doesn't apply to short captions, so do these now.

```
  \ifdim\wd\@tempboxa<\linewidth
```

[18] Email: `asnd@triumf.ca`

[19] Email from `henrik@tele.ntnu.no` on 2002/02/10.

```
    \@contpre
    {\@contnfont #1\@contdelim}\@memtempa
    {\@contcshortstyle \@conttfont #2\@contfinal\par}
  \else
    \if@conthang
```

For a hanging caption we have to measure the width of the caption name, then typeset the whole caption in a hanging paragraph.

```
      \sbox\@tempboxa{\@contnfont #1\@contdelim}
      \@contpre%
      {\@contcstyle\hangindent=\wd\@tempboxa
       \noindent\box\@tempboxa\@memtempa \@conttfont #2\@contfinal\par}
    \else
      \if@contindent
```

An indented caption is similar, except the amount of indentation is kept in `\@contindw`.

```
        \@contpre%
        {\@contnfont #1\@contdelim}\@memtempa
        {\@contcstyle\hangindent=\@contindw
                     \hangafter=\@ne\@conttfont #2\@contfinal\par}% <- v1.4
      \else
```

For the normal style, just typeset the caption.

```
        \@contpre%
        {\@contnfont #1\@contdelim}\@memtempa
        {\@contcstyle \@conttfont #2\@contfinal\par}
      \fi
    \fi
  \fi
```

Finish off the typesetting by processing the post-text, and if not using the normal width then close off the `\parbox`, and lastly put in some vertical space.

```
  \@contpost
  \if@contcw
    \par
    }  % end of the \parbox
  \fi
  \vskip\belowcaptionskip}

```

18.2.1 Continuation captions and legends

`\contcaption` `\contcaption{`⟨*text*⟩`}` is a user-level command. It is a simplified version of the normal `\caption` command as it doesn't have to deal with numbering or list of ... entries.

However, Brent Lievers[20] requested that `\label` should pick up the correct caption number after a `\contcaption`.

```
\newcommand{\contcaption}{%
```

[20] (`lieversb@post.queensu.ca`) in CTT thread *Figures*, 2003/11/14.

```
\addtocounter{\@captype}{\m@ne}\refstepcounter{\@captype}%
\@contcaption\@captype}
```

\@@contcaption
\@contcaption

\@contcaption is the workhorse for the \contcaption command. In turn, it uses the \@makecaption command (defined in the usual classes) to do most of its work. It uses the number of the previous \caption command in the same type of float and its implementation includes much of the code used in the LaTeX \@caption command.

First specify \@@contcaption{⟨*type*⟩}{⟨*title*⟩} to save some repetitive code.

```
\long\def\@@contcaption#1#2{%
  \par
  \begingroup
     \@parboxrestore
     \if@minipage
       \@setminipage
     \fi
     \normalsize
     \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #2}\par
  \endgroup}
```

Now for \@contcaption{⟨*type*⟩}{⟨*title*⟩}. It has to flush out any subcaptions at the appropriate time, as well as typesetting the caption.

```
\long\def\@contcaption#1#2{%
  \if@contbotsub
    \@memlistsubcaptions{#1}%
    \@@contcaption{#1}{#2}%
  \else
    \@@contcaption{#1}{#2}%
    \@memlistsubcaptions{#1}%
  \fi}
```

\memlegendinfo
\legend

The macro \legend{⟨*text*⟩}is intended to be used in a float environment for an 'anonymous' caption, but can be used anywhere.

The implementation is similar to the \caption command but we have to eliminate printing of a delimeter.

```
\newcommand{\memlegendinfo}[1]{}
\newcommand{\legend}[1]{%
  \M@gettitle{#1}%
  \memlegendinfo{#1}%
  \par
  \begingroup
     \@parboxrestore
     \if@minipage
       \@setminipage
     \fi
     \normalsize
```

```
7233      \captiondelim{\mbox{}}
7234      \@makecaption{}{\ignorespaces #1}\par
7235    \endgroup}
7236
```

\namedlegend
\memnamedlegendinfo

\namedlegend[⟨*short-title*⟩]{⟨*long-title*⟩} is like the \caption command except that it does not number the caption.
\memnamedlegendinfo{⟨*type*⟩}{⟨*short-title*⟩}{⟨*long-title*⟩}

```
7237 \newcommand{\namedlegend}{\@dblarg{\@legend\@captype}}
7238 \newcommand{\memnamedlegendinfo}[3]{}
7239
```

\@legend

\@legend{⟨*type*⟩}[⟨*short-title*⟩]{⟨*long-title*⟩} is the workhorse for the \namedlegend command. In turn, it calls \@makelegend. It requires two commands to have been defined, namely \flegtoctype and \flegtype. The command \flegtoctype{⟨*text*⟩} is responsible for writing a title text to the appropriate listof file. \flegtype is responsible for typeseting the name of the legend.

```
7240 \long\def\@legend#1[#2]#3{%
7241   \M@gettitle{#2}%
7242   \memnamedlegendinfo{#1}{#2}{#3}%
7243   \par
7244   \csname flegtoc#1\endcsname{#2}%
7245   \begingroup
7246     \@parboxrestore
7247     \if@minipage
7248       \@setminipage
7249     \fi
7250     \normalsize
7251     \@makecaption{\csname fleg#1\endcsname}{\ignorespaces #3}\par
7252   \endgroup}
7253
```

18.2.2 Non-float captions

\newfixedcaption
\renewfixedcaption
\providefixedcaption

These commands are defined in terms of their \...command counterparts. Call as \...fixedcaption[⟨*capcommand*⟩]{⟨*command*⟩}{⟨*env*⟩}

```
7254 \newcommand{\newfixedcaption}[3][\caption]{%
7255   \newcommand{#2}{\def\@captype{#3}#1}}
7256 \newcommand{\renewfixedcaption}[3][\caption]{%
7257   \renewcommand{#2}{\def\@captype{#3}#1}}
7258 \newcommand{\providefixedcaption}[3][\caption]{%
7259   \providecommand{#2}{\def\@captype{#3}#1}}
7260
```

18.2.3 Bilingual captions

The bilingual caption commands all use internal grouping so that any changes are kept local. This has the unfortunate side-effect that any \label command

must be within the grouping otherwise the wrong number is picked up. To make the coding, if not necessarily the use, of the commands simpler, I have not used the traditional style of square brackets for optional caption text arguments. Instead, empty 'required' arguments are used as the implementation means.

\membitwonumcaptioninfo
\membionenumcaptioninfo
\membicaptioninfo

```
\membitwonumcaptioninfo{type}{\thetypenum}{⟨short1⟩}{⟨long1⟩}
                                {⟨name2⟩}{⟨short2⟩}{⟨long2⟩}
\membionenumcaptioninfo{type}{\thetypenum}{⟨short1⟩}{⟨long1⟩}
                                {⟨name2⟩}{⟨short2⟩}{⟨long2⟩}
\membicaptioninfo{type}{\thetypenum}{⟨short1⟩}{⟨long1⟩}
                                {⟨name2⟩}{⟨long2⟩}
```

```
\newcommand{\membitwonumcaptioninfo}[7]{}
\newcommand{\membionenumcaptioninfo}[7]{}
\newcommand{\membicaptioninfo}[6]{}
```

\bitwonumcaption

The 6 arguments are: optional label, short and long in language 1, name in language 2, and short and long in language 2. Both texts are put into the List of as numbered entries.

```
\newcommand{\bitwonumcaption}[6][\@empty]{%
  \begingroup
  \let\memcaptioninfo\@gobblefour
```

Check if the first language argument is vacuous, then call the normal \caption for language 1.

```
  \@ifmtarg{#2}{\def\m@mscapi{#3}\caption{#3}}%
               {\def\m@mscapi{#2}\caption[#2]{#3}}%
```

Do the optional labeling.

```
  \ifx \@empty #1\else
    \label{#1}%
  \fi
```

Remove any extra spacing between the captions, and set the NAME for the second caption. Use a command to transfer the NAME to the renewal code to avoid circularity if for example, we are trying to redefine \tablename as \tablename. Decrement the caption counter.

```
  \setlength{\abovecaptionskip}{0pt}%
  \setlength{\belowcaptionskip}{0pt}%
  \edef\@memtempc{#4}%
  \expandafter\renewcommand\csname \@captype name\endcsname{\@memtempc}%
  \addtocounter{\@captype}{-1}%
```

Now repeat for the second language caption.

```
  \@contmidbi
  \@ifmtarg{#5}{\def\m@mscapii{#6}\caption{#6}}%
               {\def\m@mscapii{#5}\caption[#5]{#6}}%
  \membitwonumcaptioninfo{\@captype}{\@nameuse{the\@captype}}%
                         {\m@mscapi}{#3}{#4}{\m@mscapii}{#6}%
  \endgroup}
```

```
7284
```

\bionenumcaption The 6 arguments are: optional labelling, short and long in language 1, name in language 2, and short and long in language 2. Both texts are put into the List of, but only the first is numbered.

```
7285 \newcommand{\bionenumcaption}[6][\@empty]{%
7286   \begingroup
7287   \let\memcaptioninfo\@gobblefour
```

Check if the first language argument is vacuous, then call the normal \caption for language 1.

```
7288   \@ifmtarg{#2}{\def\m@mscapi{#3}\caption{#3}}%
7289                {\def\m@mscapi{#2}\caption[#2]{#3}}%
```

Do the optional labeling.

```
7290   \ifx \@empty #1\else
7291     \label{#1}%
7292   \fi
```

Do the between captions code.

```
7293   \setlength{\abovecaptionskip}{0pt}%
7294   \setlength{\belowcaptionskip}{0pt}%
7295   \edef\@memtempc{#4}%
7296   \expandafter\renewcommand\csname \@captype name\endcsname{\@memtempc}
```

Use a continuation caption for the second language, not forgetting to add the appropriate unnumbered text to the List.

```
7297   \@contmidbi
7298   \contcaption{#6}%
7299   \@ifmtarg{#5}{%
7300     \def\m@mscapii{#6}%
7301     \addcontentsline{\csname ext@\@captype\endcsname}{\@captype}%
7302       {\protect\numberline{}{\ignorespaces #6}}}{%
7303     \def\m@mscapii{#5}%
7304     \addcontentsline{\csname ext@\@captype\endcsname}{\@captype}%
7305       {\protect\numberline{}{\ignorespaces #5}}}%
7306   \membionenumcaptioninfo{\@captype}{\@nameuse{the\@captype}}%
7307                          {\m@mscapi}{#3}{#4}{\m@mscapii}{#6}%
7308   \endgroup}
7309
```

\bicaption The 5 arguments are: optional labelling, short and long in language 1, name in language 2, and long in language 2. Only the first text is put into the List.

```
7310 \newcommand{\bicaption}[5][\@empty]{%
7311   \begingroup
7312   \let\memcaptioninfo\@gobblefour
```

Check if the first language argument is vacuous, then call the normal \caption for language 1.

```
7313   \@ifmtarg{#2}{\def\m@mscapi{#3}\caption{#3}}%
7314                {\def\m@mscapi{#2}\caption[#2]{#3}}%
```

Do the optional labeling.

```
\ifx \@empty #1\else
  \label{#1}%
\fi
```

Do the between captions code and finally just use \contcaption for the second language.

```
\setlength{\abovecaptionskip}{0pt}%
\setlength{\belowcaptionskip}{0pt}%
\edef\@memtempc{#4}
\expandafter\renewcommand\csname \@captype name\endcsname{\@memtempc}%
\@contmidbi
\contcaption{#5}%
\membicaptioninfo{\@captype}{\@nameuse{the\@captype}}%
                 {\m@mscapi}{#3}{#4}{#5}%
\endgroup}

```

\bicontcaption The 3 arguments are long in language 1, name in language 2, and long in language 2.

```
\newcommand{\bicontcaption}[3]{%
  \begingroup
```

Call \contcaption for language 1.

```
  \contcaption{#1}%
```

Do the between captions code and use \contcaption for the second language.

```
  \setlength{\abovecaptionskip}{0pt}%
  \setlength{\belowcaptionskip}{0pt}%
  \edef\@memtempc{#2}%
  \expandafter\renewcommand\csname \@captype name\endcsname{\@memtempc}%
  \@contmidbi
  \contcaption{#3}%
  \endgroup}

```

18.2.4 Support for the subfigure package functionality

Much of the code in this section is based on the subfigure package code, by kind permission of its author, Steven Douglas Cochran. To try and avoid clashes with the real subfigure code I have used different macro names, especially when I have copied the code.

\subcaptionstyle The paragraphing style for subcaptions.

\@contsubcstyle

```
\newcommand{\subcaptionstyle}[1]{\def\@contsubcstyle{#1}}
\subcaptionstyle{}

```

\if@shortsubcap For dealing with short and hanging subcaptions. Analagous to the subfigure nooneline and hang options. The default is normal subcaptions.

\if@hangsubcap

\shortsubcaption

\hangsubcaption

\normalsubcaption

```
\newif\if@shortsubcap
\newif\if@hangsubcap
\newcommand*{\shortsubcaption}{\@shortsubcaptrue}
\newcommand*{\hangsubcaption}{\@hangsubcaptrue}
\newcommand*{\normalsubcaption}{\@shortsubcapfalse\@hangsubcapfalse}
\normalsubcaption
```

`\subfloattopskip` `\subfloatcapskip` `\subfloatcaptopadj` `\subfloatbottomskip` `\subfloatlabelskip` `\subfloatcapmargin`

These `\subfloat...` lengths are analagous to the subfigure `\subfig...` lengths.

```
\newskip\subfloattopskip
\newskip\subfloatcapskip
\newskip\subfloatcaptopadj
\newskip\subfloatbottomskip
\newskip\subfloatlabelskip
\newdimen\subfloatcapmargin
```

`\if@tightsubcap` `\loosesubcaptions` `\tightsubcaptions`

Unlike the subfigure package the class provides no options for subcaptions. These macros provide the subfigure loose/tight option functions. Set the default to tight.

```
\newif\if@tightsubcap
\newcommand{\loosesubcaptions}{%
  \subfloattopskip = 10\p@
  \subfloatcapskip  = 10\p@
  \subfloatcaptopadj = \z@
  \subfloatbottomskip = 10\p@
  \subfloatlabelskip = 0.33em
  \subfloatcapmargin = 10\p@
  \@tightsubcapfalse
}

\newcommand{\tightsubcaptions}{%
  \subfloattopskip = 5\p@
  \subfloatcapskip  = \z@
  \subfloatcaptopadj = 3\p@
  \subfloatbottomskip = 5\p@
  \subfloatlabelskip = 0.33em \@plus 0.07em \@minus 0.03em
  \subfloatcapmargin = \z@
  \@tightsubcaptrue
}
\tightsubcaptions
```

`\subcaptionsize` `\@subcapsize` `\subcaptionlabelfont` `\@subcaplabelfont` `\subcaptionfont` `\@subcapfont`

These macros set the size and fonts for the subcaptions. Set the defaults to `\footnotesize`, and the normal roman font.

```
\newcommand*{\subcaptionsize}[1]{\def\@subcapsize{#1}}
\newcommand*{\subcaptionlabelfont}[1]{\def\@subcaplabelfont{#1}}
\newcommand*{\subcaptionfont}[1]{\def\@subcapfont{#1}}
\subcaptionsize{\footnotesize}
\subcaptionlabelfont{\normalfont}
```

```
\subcaptionfont{\normalfont}
```

`\@contkeep` `\@contkeep` stores the current subfloat number in counter `@contsubnum` and
`\@contset` `\@contset` sets the subfloat number to the value of `@contsubnum`.
`\subconcluded` `\subconcluded` sets the subfloat number to zero.

```
\newcounter{@contsubnum}
\newcommand{\@contkeep}{%
  \setcounter{@contsubnum}{\value{sub\@captype}}}
\newcommand{\@contset}{%
  \setcounter{sub\@captype}{\value{@contsubnum}}}
\newcommand{\subconcluded}{%
  \setcounter{sub\@captype}{0}}
```

`\if@contbotsub` A flag indicating whether the subcaption is to be at the bottom or top of the subfloat; TRUE for the subcaption at the bottom.

```
\newif\if@contbotsub
  \@contbotsubtrue
```

`\subcaption` `\subcaption[`⟨*list-entry*⟩`]{`⟨*caption*⟩`}` is a generic subcaption. There is no subfigure equivalent.

```
\newcommand{\subcaption}{%
  \bgroup
    \let\label=\memsub@label
    \ifdonemaincaption\else
      \advance\csname c@\@captype\endcsname\@ne
    \fi
    \refstepcounter{sub\@captype}\@contkeep
    \@ifnextchar [%
      {\@memsubcap{sub\@captype}}%
      {\@memsubcap{sub\@captype}[\@empty]}}
```

`\@memsubcap` This handles the optional argument to `\subcaption`. It sets `\@tempdima` to `\hsize` as later on `\@makesubfloatcaption` uses this.

```
\long\def\@memsubcap#1[#2]#3{%
  \@tempdima=\hsize
  \vskip\subfloatcapskip
  \ifx \@empty #2
    \@memsubcaption{#1}{#3}{#3}%
  \else
    \@memsubcaption{#1}{#2}{#3}%
  \fi
  \vskip\subfloatcapskip
  \egroup}
```

`\@memsubcaption` `\@memsubcaption{`⟨*type*⟩`}{`⟨*list-entry*⟩`}{`⟨*caption*⟩`}` typesets a subcaption. This is a copy of the subfigure `\@subcaption` macro.

```
\newcommand{\@memsubcaption}[3]{%
  \ifx \relax#2\relax \else
    \bgroup
      \let\label\@gobble
      \let\protect\string
      \def\@memsubcaplabel{\@nameuse{@@the#1}}%
      \xdef\@memsubfigcaptionlist{%
        \@memsubfigcaptionlist,%
  {\protect\numberline{\@memsubcaplabel}\noexpand{\ignorespaces #2}}}%
    \egroup
  \fi
  \@makesubfloatcaption{\@nameuse{@the#1}}{#3}}
```

`\contsubcaption` `\contsubcaption[⟨caption⟩]` is the continued version of `\subcaption`.

```
\newcommand{\contsubcaption}{%
  \bgroup
    \let\label=\memsub@label
    \@contset
    \refstepcounter{sub\@captype}\@contkeep
    \@ifnextchar [%
      {\@memsubcap{sub\@captype}}%
      {\@memsubcap{sub\@captype}[\@empty]}}
```

`subfloat` The subfigure documentation suggests a way of defining a `subfloat` environment. This is a trivial implementation because the `\subcaption` and `\contsubcaption` commands are provided by the class, and can be used within a `subfloat`.

```
\newenvironment{subfloat}{}{}
```

`\subbottom` `\subbottom[⟨list-entry⟩][⟨subcaption⟩]{⟨text⟩}` typesets a subcaption below the `\@memsubbody` ⟨*text*⟩. Most of the work is performed by the `\@memsubbody` macro.

```
\newcommand{\subbottom}{%
  \@contbotsubtrue
  \@memsubbody}

\newcommand{\@memsubbody}{%
  \bgroup
  \let\label=\memsub@label
  \ifdonemaincaption\else
    \advance\csname c@\@captype\endcsname\@ne
  \fi
  \refstepcounter{sub\@captype}\@contkeep%
  \leavevmode
  \@ifnextchar [%
    {\@memsubfig}%
    {\@memsubfig[\@empty]}}
```

\contsubbottom
\@memcontsubbody
These are the continued versions of \subbottom and \@memsubbody.

```
\newcommand{\contsubbottom}{%
  \@contbotsubtrue
  \@memcontsubbody}

\newcommand{\@memcontsubbody}{%
  \bgroup
  \let\label=\memsub@label
  \@contset
  \refstepcounter{sub\@captype}\@contkeep%
  \leavevmode
  \@ifnextchar [%
    {\@memsubfig}%
    {\@memsubfig[\@empty]}}

```

\subtop
\contsubtop
These are similar to \subbottom and \contsubbottom except that they put the subcaption on top of the ⟨*text*⟩.

```
\newcommand{\subtop}{%
  \@contbotsubfalse
  \@memsubbody}

\newcommand{\contsubtop}{%
  \@contbotsubfalse
  \@memcontsubbody}

```

\@memsubfig
This is a revised version of the subfigure \@subfigure command — just the called macro names are changed.

```
\def\@memsubfig[#1]{%
  \@ifnextchar [%
    {\@memsubfloat{sub\@captype}[#1]}%
    {\@memsubfloat{sub\@captype}[\@empty #1][#1]}}

```

\@memsubfloat
This is a modified version of the subfigure \@subfloat command Essentially the \csname if#1topcap\endcsname constructs are replaced by \if@contbotsub.

```
\def\@memsubfloat#1[#2][#3]#4{%
  \@tempcnta=\@ne
  \if@tightsubcap
    \if@minipage
      \@tempcnta=\z@
    \else
      \ifdim\lastskip=\z@
        \@tempcnta=\@ne
      \else
        \@tempcnta=\tw@
      \fi
    \fi
```

```
  \fi
  \if@contbotsub
    \def\subfig@top{\subfloattopskip}%
    \def\subfig@bottom{\subfloatbottomskip}%
  \else
    \def\subfig@top{\subfloatbottomskip}%
    \def\subfig@bottom{\subfloattopskip}%
  \fi
  \setbox\@tempboxa \hbox{#4}%
  \@tempdima=\wd\@tempboxa
  \vtop\bgroup
    \vbox\bgroup
    \ifcase\@tempcnta
      \@minipagefalse
    \or
      \vspace{\subfig@top}
    \or
      \ifdim \lastskip=\z@ \else
        \@tempskipb\subfig@top\@xaddvskip
      \fi
    \fi
    \if@contbotsub
      \box\@tempboxa\egroup
      \ifx \@empty#3\relax \else
        \vskip\subfloatcapskip
        \@memsubcaption{#1}{#2}{#3}%
      \fi
    \else
      \ifx \@empty#3\relax \else
        \@memsubcaption{#1}{#2}{#3}%
        \vskip\subfloatcapskip
        \vskip\subfloatcaptopadj
      \fi\egroup
      \box\@tempboxa
    \fi
    \vspace{\subfig@bottom}
  \egroup
\egroup}

```

The following series of macros, from subfigure, control the typesetting of the subcaptions.

`\@memsubfigcaptionlist` A copy of `\@subfigcaptionlist`.

```
\newcommand*{\@memsubfigcaptionlist}{}
```

`\memlistsubcaptions` A copy of `\listsubcaptions`.

```
\newcommand*{\memlistsubcaptions}{%
  \@ifstar
    {\gdef\@memsubfigcaptionlist{}}%
```

```
    {\@memlistsubcaptions{\@captype}}}

```

`\@memlistsubcaptions` A copy of `\@listsubcaptions`.

```
\newcommand*{\@memlistsubcaptions}[1]{%
  \@ifundefined{@captype}{}{%
    \@ifundefined{ext@sub#1}{}{%
      \@for \@tempa:=\@memsubfigcaptionlist \do {%
        \ifx \@empty\@tempa\relax \else
          \addcontentsline{\@nameuse{ext@sub#1}}{sub#1}{\@tempa}%
        \fi}}}%
  \gdef\@memsubfigcaptionlist{}}

```

`\@makesubfloatcaption` This is a copy of `\@makesubfigurecaption`.

```
\newcommand{\@makesubfloatcaption}[2]{%
  \setbox\@tempboxa\hbox{%
    \@subcapsize
    {\@subcaplabelfont #1}{\@subcapfont\ignorespaces #2}}%
  \@tempdimb=-\subfloatcapmargin
  \multiply\@tempdimb\tw@
  \advance\@tempdimb\@tempdima
  \hb@xt@\@tempdima{%
    \hss
    \ifdim \wd\@tempboxa >\@tempdimb
      \memsubfig@caption{#1}{#2}%
    \else
      \if@shortsubcap
        \memsubfig@caption{#1}{#2}%
      \else
        \box\@tempboxa
      \fi
    \fi
    \hss}}

```

`\memsubfig@caption` This is a copy of `\subfig@caption`.

```
\newcommand{\memsubfig@caption}[2]{%
  \if@hangsubcap
    \sbox{\@tempboxa}{\@subcapsize\@subcaplabelfont #1}%
    \addtolength{\@tempdimb}{-\wd\@tempboxa}%
    \usebox{\@tempboxa}%
    \memsubfig@captionpar{\@tempdimb}{%
      {\@subcapfont\ignorespaces #2}}%
  \else
    \memsubfig@captionpar{\@tempdimb}{{\@subcaplabelfont #1}%
      {\@subcapfont\ignorespaces #2}}%
  \fi}

```

\memsubfig@captionpar This replaces \subfig@captionpar.

```
\newcommand{\memsubfig@captionpar}[2]{%
  \parbox[t]{#1}{\@subcapsize\@contsubcstyle #2}}
```

\memsub@label These are copies of \sub@label and \subref.

```
\newcommand{\memsub@label}{%
  \@ifnextchar (%
    {\sf@memsub@label}%
    {\sf@memsub@label(Sub\@captype\space
                      \@nameuse{p@sub\@captype}%
                      \@nameuse{thesub\@captype})}}
```

\sf@memsub@label This is a copy of \sf@sub@label.

```
\def\sf@memsub@label(#1)#2{%
  \protected@edef\mem@currentlabelname{#1}%
  \sf@@memsub@label{#2}}
```

\sf@@memsub@label This is an expanded copy of \sf@@sub@label processed after any packages may have been loaded.

```
\AtBeginDocument{%
  \@ifpackageloaded{nameref}{%
```

The nameref package is loaded.

```
    \newcommand*{\sf@@memsub@label}[1]{%
      \@bsphack
      \protected@write\@auxout{}{%
        \string\newlabel{#1}%
          {{\@nameuse{p@sub\@captype}\@nameuse{@@thesub\@captype}}%
          {\thepage}%
          {\mem@currentlabelname\relax}%
          {\@currentHref}{}}}%
      \protected@write\@auxout{}{%
        \string\newlabel{sub@#1}%
          {{\@nameuse{@@thesub\@captype}}%
          {\thepage}%
          {\mem@currentlabelname\relax}%
          {\@currentHref}{}}}%
  \@esphack}
  }{\@ifpackageloaded{hyperref}{%
```

The hyperref package is loaded, but not nameref.

```
    \newcommand*{\sf@@memsub@label}[1]{%
      \@bsphack
      \protected@write\@auxout{}{%
        \string\newlabel{#1}%
          {{\@nameuse{p@sub\@captype}\@nameuse{@@thesub\@captype}}%
          {\thepage}%
          {\mem@currentlabelname\relax}%
```

```
          {\@currentHref}{}}}%
      \protected@write\@auxout{}{%
        \string\newlabel{sub@#1}%
          {{\@nameuse{@@thesub\@captype}}%
          {\thepage}%
          {\mem@currentlabelname\relax}%
          {\@currentHref}{}}}%
  \@esphack}
  }{%
```

Neither the hyperref nor the nameref package is loaded.

```
    \let\@memoldlabel\label
    \newcommand*{\sf@@memsub@label}[1]{%
      \@bsphack
      \@memoldlabel{#1}%
      \protected@write\@auxout{}{%
        \string\newlabel{sub@#1}%
          {{\@nameuse{@@thesub\@captype}}%
          {\thepage}}}%
  \@esphack}
  }{}%
  }
}
```

`\subcaptionref` This is a copy of the `\subref` macro.

```
\DeclareRobustCommand{\subcaptionref}{%
  \@ifstar{\ssc@ref}{\sc@ref}}
```

`\ssc@ref` `\sc@ref` The implementation of the starred and unstarred forms of `\subcaptionref`.

```
\newcommand*{\ssc@ref}[1]{\ref{sub@#1}}
\newcommand*{\sc@ref}[1]{{\@subcaplabelfont\ref{sub@#1}}}
```

18.2.5 Side captions

`\m@mscap@capbox` `\m@mscap@fbox` We need two save boxes, one to hold the caption and the other for the float material.

```
\newsavebox{\m@mscap@capbox}
\newsavebox{\m@mscap@fbox}
```

`\sidecapsep` `\sidecapwidth` `\sidecapsep` is the space between the text and the caption, which is set in a box `\sidecapwidth` wide. These are initialized to the `\marginpar...` values.

```
\newdimen\sidecapsep
  \sidecapsep=\marginparsep
\newdimen\sidecapwidth
  \sidecapwidth=\marginparwidth
```

\setsidecaps `\setsidecaps{⟨sep⟩}{⟨width⟩}` sets the `\sidecapsep` and `\sidecapwidth` (Ivars Finvers noted that the initial values don't change if the `\marginpar...` values change, such as at `\checkandfixthelayout`).

```
\newcommand*{\setsidecaps}[2]{%
  \setlength{\sidecapsep}{#1}\@memznegtest{\sidecapsep}%
  \setlength{\sidecapwidth}{#2}\@memznegtest{\sidecapwidth}}
```

\m@m@tempdima Lengths for internal use. `\m@m@tempdima` is meant for general temporary use.

\m@mscapraise

```
\newdimen\m@m@tempdima
\newdimen\m@mscapraise
```

\sidecapraise Length to make (small) adjustments to the position of the caption wrt the float.

```
\newdimen\sidecapraise
  \sidecapraise \z@
```

\setsidecappos `\setsidecappos{⟨pos⟩}`, where ⟨*pos*⟩ is one of `t`, `c`, or `b`, sets the vertical position of the caption in relation to the float and the result is saved as `\m@mscappos`. The default is `c`.

\m@mscappos

```
\newcommand*{\setsidecappos}[1]{%
  \def\m@mscappos{#1}\def\@tempb{t}%
  \ifx\@tempb\m@mscappos
  \else
    \def\@tempb{b}%
    \ifx\@tempb\m@mscappos
    \else
      \def\@tempb{c}%
      \ifx\@tempb\m@mscappos
      \else
        \@memerror{Argument to \string\setsidecappos\space is not t or c or b.
                   \MessageBreak Set to c}{\@ehc}%
        \def\m@mscappos{c}%
      \fi
    \fi
  \fi}
\setsidecappos{c}
```

\sidecapmargin `\sidecapmargin{⟨margin⟩}`, where ⟨*marg*⟩ is one of `left`, `right`, `inner`, or `outer`, controls the margin where the caption will be put. The result is saved as `\m@mscapmarg` as a number. The default is `left`.

\m@mscapmarg

```
\newcommand{\sidecapmargin}[1]{%
  \def\@tempa{#1}\def\@tempb{left}%
    \ifx\@tempb\@tempa
      \def\m@mscapmarg{0}%   left
    \else
      \def\@tempb{right}%
```

```
      \ifx\@tempb\@tempa
        \def\m@mscapmarg{1}%  right
      \else
        \def\@tempb{outer}%
        \ifx\@tempb\@tempa
          \def\m@mscapmarg{2}%  outer
        \else
          \def\@tempb{inner}%
          \ifx\@tempb\@tempa
            \def\m@mscapmarg{3}%  inner
          \else
            \@memerror{Unrecognized argument for \string\sidecapmargin}%
                      {\@ehc}%
            \def\m@mscapmarg{-1}% error
          \fi
        \fi
      \fi
    \fi}
\sidecapmargin{left}

```

`\ifscapmargleft` `\scapmarglefttrue` `\scapmargleftfalse`

`\ifscapmargleft` is TRUE the caption should be in the left hand margin, otherwise in the right hand margin.

```
\newif\ifscapmargleft

```

`\sidecapfloatwidth` `\m@mscapmainwidth`

`\sidecapfloatwidth` is the width of the box holding the float. Note that this is a macro, not a length, so must be changed using `\renewcommand*`. The default is `\linewidth`. Later, `\m@mscapmainwidth` will be set to the current length specification from `\sidecapfloatwidth`.

```
\def\sidecapfloatwidth{\linewidth}
\newdimen\m@mscapmainwidth

```

`\m@mscaplkern` `\setm@mscaplkern`

`\setm@mscaplkern` is a utility macro to calculate the kern (`\m@mscaplkern`) required when the caption is in the left margin.

```
\newdimen\m@mscaplkern
\newcommand*{\setm@mscaplkern}{%
  \m@mscaplkern=\sidecapwidth
  \advance\m@mscaplkern \sidecapsep
  \advance\m@mscaplkern \m@mscapmainwidth}

```

`\sidecapstyle`

`\sidecapstyle` is called just before the caption is set. It can be redefined to set different caption style parameters. The default is raggedleft for left margin captions and raggedright for right margin captions.

```
\newcommand*{\sidecapstyle}{%
%%%  \captionnamefont{\bfseries}%
  \ifscapmargleft
```

```
    \captionstyle{\raggedleft}%
  \else
    \captionstyle{\raggedright}%
  \fi}
```

\sidecaption The whole shebang is in the sidecaption environment but it is more convenient to specify this via macros rather than directly as an environment. \sidecaption[⟨*fortoc*⟩]{⟨*title*⟩}[⟨*label*⟩] is what it looks like to the user, but internally further macros handle all the arguments.

```
\newcommand*{\sidecaption}{%
  \@ifnextchar [{\@sidecaption}{\@sidecaption[]}}
```

\@sidecaption \@sidecaption[⟨*fortoc*⟩]{⟨*title*⟩} grabs the first two arguments.

```
\def\@sidecaption[#1]#2{%
  \@ifnextchar [{\@@sidecaption{#1}{#2}}{\@@sidecaption{#1}{#2}[]}}
```

\@mem@scap@beforehook \@mem@scap@afterhook Two hooks into the 'scap' internals. By default they do nothing, but can be used make say sidecaption align towards the spine in twosided documents. This is done by redefining *before* to \checkoddpage\ifoddpage\else\raggedleft\fi and redefining *after* to \par. Note that since 'sidecontcaption', 'sidenamedlegend' and 'sidelegend' all share the same basic internals redefining the two hooks will affect those as well. The user might want to create their own special environment, say, like this:

```
\makeatletter
\newenvironment{sidecaption*}{
  \renewcommand\@mem@scap@beforehook{\checkoddpage\ifoddpage\else\raggedleft\fi}
  \renewcommand\@mem@scap@afterhook{\par}
  \begin{sidecaption}}
  {\end{sidecaption}}
\makeatother
```

```
\newcommand\@mem@scap@beforehook{}
\newcommand\@mem@scap@afterhook{}
```

\@@sidecaption \@@sidecaption{⟨*fortoc*⟩}{⟨*title*⟩}[⟨*label*⟩] is the last macro in the chain and handles all three arguments. This does all the work for \begin{sidecaption}

\m@mscap@fortoc \m@mscap@forcap \m@mscaplabel First, save all the arguments as macros.

```
\def\@@sidecaption#1#2[#3]{%
  \ifx\@empty#1\@empty
    \def\m@mscap@fortoc{#2}%
  \else
    \def\m@mscap@fortoc{#1}%
  \fi
  \def\m@mscap@forcap{#2}%
  \ifx\@empty#3\@empty
```

```
    \def\m@mscaplabel{}%
  \else
    \def\m@mscaplabel{\@bsphack\label{#3}\@esphack}%
  \fi
```

Set the float width, calculate the left margin kern, and start a `minipage` to hold the float, saving it in box `\m@mscap@fbox`.

```
  \m@mscapstart@fbox}

```

`\m@mscapstart@fbox` `\m@mscapend@fbox`

`\m@mscapstart@fbox` is the macro that actually sets the float width, calculates the left margin kern, and starts the float's `minipage`. The macro `\m@mscapend@fbox` ends the box.

```
\newcommand*{\m@mscapstart@fbox}{%
  \@mem@scap@beforehook%
  \setlength{\m@mscapmainwidth}{\sidecapfloatwidth}%
  \setm@mscaplkern
  \begin{lrbox}{\m@mscap@fbox}%
    \begin{minipage}[c]{\m@mscapmainwidth}}
\newcommand*{\m@mscapend@fbox}{%
    \end{minipage}%
  \end{lrbox}}

```

`\endsidecaption`

`\endsidecaption` does the work for `\end{sidecaption}`.
Finish the float minipage, then increment the caption counter and call `\label` via `\m@mscaplabel`.

```
\def\endsidecaption{%
  \m@mscapend@fbox
  \refstepcounter\@captype
  \m@mscaplabel
```

Set the caption inside a minipage, saving it in box `\m@mscap@capbox`.

```
  \begin{lrbox}{\m@mscap@capbox}%
    \begin{minipage}[c]{\sidecapwidth}%
      \sidecapstyle
      \@caption\@captype[\m@mscap@fortoc]{\m@mscap@forcap}
    \end{minipage}%
  \end{lrbox}%
```

Output the float and caption.

```
  \m@mscapopboxes}

```

`\m@mscapopboxes`

Having determined how high the caption box must be raised with respect to the float box, output the boxes.

```
\newcommand*{\m@mscapopboxes}{%
  \m@mcalcscapraise
```

Set the float (from box \m@mscap@fbox) then the caption (from box \m@mscap@capbox) kerning it to the left or right as appropriate.

```
\usebox{\m@mscap@fbox}\m@mscapcheckside
\ifscapmargleft%
  \rlap{\kern-\m@mscaplkern
        \raisebox{\m@mscapraise}{\usebox{\m@mscap@capbox}}}%
\else%
  \rlap{\kern\sidecapsep
        \raisebox{\m@mscapraise}{\usebox{\m@mscap@capbox}}}%
\fi
```

Finally, make \m@mscapthisside a no-op.

```
\gdef\m@mscapthisside{}%
\@mem@scap@afterhook%
}
```

\m@mcalcscapraise Calculate the amount the caption might have to be raised wrt the float. This depends on the position:
t: raise by the difference in heights
c: shouldn't have to do anything as the minipages are meant to center aligned
b: lower by the difference in depths
However, experiments showed that a little bit of tweaking might help. The final adjustment, \sidecapraise is controlled by the user.

```
\newcommand*{\m@mcalcscapraise}{%
  \def\@tempb{t}%
  \ifx\m@mscappos\@tempb
    \settoheight{\m@m@tempdima}{\strut\usebox{\m@mscap@capbox}}%
    \settoheight{\m@mscapraise}{\usebox{\m@mscap@fbox}}%
    \advance\m@mscapraise -\m@m@tempdima
    \advance\m@mscapraise  0.5ex
  \else
    \def\@tempb{b}%
    \ifx\m@mscappos\@tempb
      \settodepth{\m@m@tempdima}{\usebox{\m@mscap@fbox}}%
      \settodepth{\m@mscapraise}{\strut\usebox{\m@mscap@capbox}}%
      \advance\m@mscapraise -\m@m@tempdima
    \else
      \m@mscapraise=\z@
      \advance\m@mscapraise 0.25ex
    \fi
  \fi
  \advance\m@mscapraise  \sidecapraise}
```

\m@mscapcheckside This macro determines whether the caption should be set in the left or right margin. In twocolumn documents the caption for a single column float is always set in the adjacent margin. Starred floats are treated as regular floats as in a onecolumn document.

```
\newcommand*{\m@mscapcheckside}{%
  \if@twocolumn
    \ifdim\hsize=\textwidth%  float*
      \m@mscapcheckregside
    \else
      \if@firstcolumn
        \scapmarglefttrue
      \else
        \scapmargleftfalse
      \fi
    \fi
  \else
    \m@mscapcheckregside
  \fi
```

Finally apply any user's override.

```
  \m@mscapthisside}
```

`\m@mscapcheckregside` This performs the margin calculation for onecolumn documents.

```
\newcommand*{\m@mscapcheckregside}{%
    \if@twoside
      \checkoddpage
      \ifnum\m@mscapmarg<\@ne%      % left
        \scapmarglefttrue
      \else
        \ifnum\m@mscapmarg=\@ne%    % right
          \scapmargleftfalse
        \else
          \ifnum\m@mscapmarg=\tw@%  % outer
            \scapmarglefttrue
            \ifoddpage
              \scapmargleftfalse
            \fi
          \else%                    % inner
            \scapmargleftfalse
            \ifoddpage
              \scapmarglefttrue
            \fi
          \fi
        \fi
      \fi
    \else%     oneside
      \scapmarglefttrue
      \ifnum\m@mscapmarg>\@ne
        \ifnum\m@mscapmarg<\thr@@
          \scapmargleftfalse
        \fi
      \fi
    \fi}
```

\overridescapmargin
\m@mscapthisside

User macro to override the calculated caption margin. Call as either: \overridescapmargin{left} or \overridescapmargin{right}

```
\newcommand*{\overridescapmargin}[1]{%
  \def\@tempb{#1}\def\@tempa{left}%
  \ifx\@tempa\@tempb
    \def\m@mscapthisside{\scapmarglefttrue}%
  \else
    \def\@tempa{right}%
    \ifx\@tempa\@tempb
      \def\m@mscapthisside{\scapmarglefttrue}%
    \else
      \@memerror{Argument to \string\overridescapmargin\space neither
                 left nor right}{\@ehc}%
      \def\m@mscapthisside{}%
    \fi
  \fi}
\newcommand*{\m@mscapthisside}{}
```

Now for the other kinds of captions.

\sidecontcaption

\sidecontcaption{⟨*title*⟩}[⟨*label*⟩] is for a continuation sidecaption.

```
\newcommand*{\sidecontcaption}{%
  \@sidecontcaption}
```

\@sidecontcaption

\@sidecontcaption{⟨*title*⟩} grabs the first argument.

```
\def\@sidecontcaption#1{%
  \@ifnextchar [{\@@sidecontcaption{#1}}{\@@sidecontcaption{#1}[]}}
```

\@@sidecontcaption

\@@sidecontcaption{⟨*title*⟩}[⟨*label*⟩] is the last macro in the chain and handles all two arguments. This does all the work for \begin{sidecontcaption}

```
\def\@@sidecontcaption#1[#2]{%
  \def\m@mscap@forcap{#1}%
  \ifx\@empty#2\@empty
    \def\m@mscaplabel{}%
  \else
    \def\m@mscaplabel{\@bsphack\label{#2}\@esphack}%
  \fi
  \m@mscapstart@fbox}
```

\endsidecontcaption

\endsidecontcaption does the work for \end{sidecontcaption}.

```
\def\endsidecontcaption{%
  \m@mscapend@fbox
  \addtocounter{\@captype}{\m@ne}\refstepcounter\@captype
  \m@mscaplabel
  \begin{lrbox}{\m@mscap@capbox}%
    \begin{minipage}[c]{\sidecapwidth}%
      \sidecapstyle
```

```
      \@contcaption\@captype{\m@mscap@forcap}
    \end{minipage}%
  \end{lrbox}%
  \m@mscapopboxes}
```

`\sidenamedlegend` `\sidenamedlegend[`⟨*fortoc*⟩`]{`⟨*title*⟩`}` is for a namedlegend sidecaption.

`\@sidenamedlegend`

```
\newcommand*{\sidenamedlegend}{%
  \@ifnextchar [{\@sidenamedlegend}{\@sidenamedlegend[]}}
\def\@sidenamedlegend[#1]#2{%
  \@@sidenamedlegend{#1}{#2}}
```

`\@@sidenamedlegend` `\@@sidenamedlegend{`⟨*fortoc*⟩`}{`⟨*title*⟩`}` is the last macro in the chain and handles all arguments. This does all the work for `\begin{sidenamedlegend}`

```
\def\@@sidenamedlegend#1#2{%
  \ifx\@empty#1\@empty
    \def\m@mscap@fortoc{#2}%
  \else
    \def\m@mscap@fortoc{#1}%
  \fi
  \def\m@mscap@forcap{#2}%
  \def\m@mscaplabel{}%
  \m@mscapstart@fbox}
```

`\endsidenamedlegend` `\endsidenamedlegend` does the work for `\end{sidenamedlegend}`.

```
\def\endsidenamedlegend{%
  \m@mscapend@fbox
  \begin{lrbox}{\m@mscap@capbox}%
    \begin{minipage}[c]{\sidecapwidth}%
      \sidecapstyle
      \@legend\@captype[\m@mscap@fortoc]{\m@mscap@forcap}
    \end{minipage}%
  \end{lrbox}%
  \m@mscapopboxes}
```

`\sidelegend` `\sidelegend{`⟨*title*⟩`}` is for a legend sidecaption.

```
\newcommand*{\sidelegend}{%
  \@@sidelegend}
```

`\@@sidelegend` `\@@sidelegend{`⟨*title*⟩`}` is the last macro in the chain and handles all arguments. This does all the work for `\begin{sidelegend}`

```
\def\@@sidelegend#1{%
  \def\m@mscap@forcap{#1}%
  \m@mscapstart@fbox}
```

\endsidelegend `\endsidelegend` does the work for `\end{sidelegend}`.

```
\def\endsidelegend{%
  \m@mscapend@fbox
  \begin{lrbox}{\m@mscap@capbox}%
    \begin{minipage}[c]{\sidecapwidth}%
      \sidecapstyle
      \legend{\m@mscap@forcap}
    \end{minipage}%
  \end{lrbox}%
  \m@mscapopboxes}

```

19 Epigraphs

This code comes from the epigraph package [Wil00a].

\beforeepigraphskip
\afterepigraphskip
\epigraphwidth
\epigraphrule
The several length commands, which can be changed by the user with `\setlength`.

```
\newlength{\beforeepigraphskip}
  \setlength{\beforeepigraphskip}{.5\baselineskip}
\newlength{\afterepigraphskip}
  \setlength{\afterepigraphskip}{.5\baselineskip}
\newlength{\epigraphwidth}
  \setlength{\epigraphwidth}{.4\textwidth}
\newlength{\epigraphrule}
  \setlength{\epigraphrule}{.4\p@}

```

\epigraphsize The size of the font to be used.

```
\newcommand{\epigraphsize}{\small}

```

\epigraphflush
\textflush
\sourceflush
The three commands to position epigraphs in the textblock and to position the components of the epigraph.

```
\newcommand{\epigraphflush}{flushright}
\newcommand{\textflush}{flushleft}
\newcommand{\sourceflush}{flushright}

```

\epigraphfontsize
\epigraphposition
\epigraphtextposition
\epigraphsourceposition
These are declarative forms of the above. It's a bit late now, but the previous macros should have been internal.

```
\newcommand{\epigraphfontsize}[1]{\def\epigraphsize{#1}}
\newcommand{\epigraphposition}[1]{\long\def\epigraphflush{#1}}
\newcommand{\epigraphtextposition}[1]{\def\textflush{#1}}
\newcommand{\epigraphsourceposition}[1]{\def\sourceflush{#1}}

```

\@epirule The internal command to draw a rule between text and source.

```
\newcommand{\@epirule}{\rule[.5ex]{\epigraphwidth}{\epigraphrule}}
```

`\@epitext` The internal command to typeset the ⟨*text*⟩. Put it into a minipage of the right size and typeset per `\textflush`.

```
\newcommand{\@epitext}[1]{%
  \begin{minipage}{\epigraphwidth}\begin{\textflush} #1\par
```

Draw a rule if it will be visible, otherwise add some extra vertical space.

```
    \ifdim\epigraphrule>\z@ \@epirule \else \vspace*{1ex} \fi
  \end{\textflush}\end{minipage}}
```

`\@episource` The internal command for typesetting the ⟨*source*⟩, which is put into a minipage and typeset according to `\sourceflush`.

```
\newcommand{\@episource}[1]{%
  \begin{minipage}{\epigraphwidth}
    \begin{\sourceflush} #1\par
  \end{\sourceflush}\end{minipage}}

```

`\epigraph` Having got the preliminaries out of the way, here's the user command for a single epigraph. This is set in a minipage to prevent breaking across a page. Position it according to `\epigraphflush`.

```
\newcommand{\epigraph}[2]{\vspace{\beforeepigraphskip}
  {\epigraphsize\begin{\epigraphflush}\begin{minipage}{\epigraphwidth}
    \@epitext{#1}\\ \@episource{#2}
    \end{minipage}\end{\epigraphflush}
    \vspace{\afterepigraphskip}}}
```

`\qitem` `\qitem` is the epigraph list version of `\item`. Set everything inside a minipage.

`\qitemlabel`

```
\newcommand{\qitem}[2]{{%
  \raggedright\item \begin{minipage}{\epigraphwidth}
    \@epitext{#1}\\ \@episource{#2}
  \end{minipage}}}
```

`\qitemlabel` is needed for a list as well. It is not going to typeset anything.

```
\newcommand{\qitemlabel}[1]{\hfill}
```

`epigraphs` Now for the epigraph list. This is defined in terms of a `list` environment.

```
\newenvironment{epigraphs}{%
```

Do the vertical space, set the font size, position according to `\epigraphflush`, and put everything into a minipage.

```
  \vspace{\beforeepigraphskip}\begin{\epigraphflush}
  \epigraphsize
  \begin{minipage}{\epigraphwidth}
   \list{}%
```

Make the list just fit the minipage (i.e., no indents).

```
    {\itemindent\z@ \labelwidth\z@ \labelsep\z@
     \leftmargin\z@ \rightmargin\z@
     \let\makelabel\qitemlabel}}%
  {\endlist\end{minipage}\end{\epigraphflush}
   \vspace{\afterepigraphskip}}
```

19.1 Epigraphs before a chapter title

\@epichapapp
\dropchapter
\undodrop

Commands to drop and restore positions of chapter titles. Dropping is accomplished by inserting vertical space before the \@chapapp command.

```
\newcommand{\dropchapter}[1]{%
  \let\@epichapapp\@chapapp
  \renewcommand{\@chapapp}{\vspace*{#1}\@epichapapp}}
\newcommand{\undodrop}{\let\@chapapp\@epichapapp}
```

Placing an epigraph before a chapter title uses the scheme outlined by Piet van Oostrum [Oos96]. This is to put a zero sized picture into the page header.

\if@epirhs
\if@epicenter

Two booleans for testing whether an epigraph is to be at the RH margin, centered, or at the LH margin. The default is RH margin.

```
\newif\if@epirhs     \@epirhstrue
\newif\if@epicenter  \@epicentertrue
```

\@epipos

This routine sets the \if@epi... booleans according to the value of \epigraphflush. If \epigraphflush is neither center nor flushleft then it defaults to flushright. We have to use this to be upward compatible with \epigraphflush being set by the user with \renewcommand.

```
\newcommand{\@epipos}{
  \long\def\@ept{flushleft}
  \ifx\epigraphflush\@ept
    \@epirhsfalse \@epicenterfalse
  \else
    \long\def\@ept{center}
    \ifx\epigraphflush\@ept
      \@epirhsfalse \@epicentertrue
    \else
      \@epirhstrue  \@epicenterfalse
    \fi
  \fi}
```

\epigraphhead

\epigraphhead[⟨*distance*⟩]{⟨*text*⟩} puts ⟨*text*⟩ at ⟨*distance*⟩ (a number, not a length) below the header at the page position specified by \epigraphflush.

```
\newcommand{\epigraphhead}[2][95]{%
```

We have to use \def instead of the normal LaTeX definition commands as we will keep on (re)defining things. For reasons that are not fully clear to me LaTeX doesn't seem to like me using a \savebox for storing the epigraph text, so I'll use a command instead.

```
  \def\@epitemp{\begin{minipage}{\epigraphwidth}#2\end{minipage}}
```

Define an epigraph page style.

```
  \def\ps@epigraph{\let\@mkboth\@gobbletwo
```

There are three possible definitions for \@oddhead depending on the value of \epigraphflush. We call \@epipos to decide which one to do.

```
    \@epipos
```

```
    \if@epirhs
      \def\@oddhead{\hfil\begin{picture}(0,0)
                          \put(0,-#1){\makebox(0,0)[r]{\@epitemp}}
                          \end{picture}}
    \else
      \if@epicenter
        \def\@oddhead{\hfil\begin{picture}(0,0)
                            \put(0,-#1){\makebox(0,0)[b]{\@epitemp}}
                            \end{picture}\hfil}
      \else
        \def\@oddhead{\begin{picture}(0,0)
                            \put(0,-#1){\makebox(0,0)[l]{\@epitemp}}
                            \end{picture}\hfil}
      \fi
    \fi
    \let\@evenhead\@oddhead
    \def\@oddfoot{\reset@font\hfil\thepage\hfil}
    \let\@evenfoot\@oddfoot}
```

Make `epigraph` be the page style for this page.

```
  \thispagestyle{epigraph}}

```

The above produces a `plain` pagestyle with the epigraph. Life is more complex if someone wants a fancy style with the epigraph. They will have to do some work, though.

`\the@epigraph` `\@epidrop` `\epigraphforheader`

`\the@pigraph` is a macro to store the contents of an epigraph, and `\@pidrop` stores the ⟨*distance*⟩ number. `\epigraphforheader[`⟨*distance*⟩`]{`⟨*text*⟩`}` defines the internal macros appropriately.

```
\newcommand{\the@epigraph}{}
\newcommand{\@epidrop}{95}
\newcommand{\epigraphforheader}[2][95]{%
  \def\@epidrop{#1}\long\def\the@epigraph{#2}}

```

`\epigraphpicture`

`\epigraphpicture` puts `\the@epigraph` into a zero-sized picture at location `(0,-\@epidrop)`. This can then be used as part of a fancy chapter header. The coding is similar to `\epigraphhead`.

```
\newcommand{\epigraphpicture}{%
  \def\@epitemp{%
    \begin{minipage}{\epigraphwidth}\the@epigraph\end{minipage}}%
  \@epipos
  \if@epirhs
    \begin{picture}(0,0)%
      \put(0,-\@epidrop){\makebox(0,0)[r]{\@epitemp}}%
    \end{picture}%
  \else
    \if@epicenter
      \begin{picture}(0,0)%
```

```
        \put(0,-\@epidrop){\makebox(0,0)[b]{\@epitemp}}%
      \end{picture}%
    \else
      \begin{picture}(0,0)%
        \put(0,-\@epidrop){\makebox(0,0)[l]{\@epitemp}}%
      \end{picture}%
    \fi
  \fi}

```

20 The deprecated font commands

The class does not support the old font changing commands unless the oldfontcommands option is used.

\@memoldfonterr
\@memoldfontwarn

Macros for old font class error and warning. E.g., \@memoldfonterr{\tt}{\ttfamily}{\texttt}

```
\newcommand*{\@memoldfonterr}[3]{%
  \@mem@error{Font command \protect#1\space is not supported}{%
    Use \protect#2, or \protect#3{...}, or the oldfontcommands option}}
\newcommand*{\@memoldfontwarn}[3]{%
  \@mem@warn{The \protect#1\space font command is deprecated.
    \MessageBreak Use \protect#2{...} or {\protect#3... } instead}}

```

Just give a single warning when an old font command is used following the oldfontcommands option.

\@mem@rmwarn
\rm

The old command for roman font.

```
\if@memoldfont
  \def\@mem@rmwarn{\@memoldfontwarn{\rm}{\textrm}{\rmfamily}}
  \DeclareOldFontCommand{\rm}{\@mem@rmwarn\gdef\@mem@rmwarn{}%
    \normalfont\rmfamily}{\mathrm}
\else
  \def\rm{\@memoldfonterr{\rm}{\textrm}{\rmfamily}}
\fi

```

\@mem@sfwarn
\sf

The old sans font command.

```
\if@memoldfont
  \def\@mem@sfwarn{\@memoldfontwarn{\sf}{\textsf}{\sffamily}}
  \DeclareOldFontCommand{\sf}{\@mem@sfwarn\gdef\@mem@sfwarn{}%
    \normalfont\sffamily}{\mathsf}
\else
  \def\sf{\@memoldfonterr{\sf}{\textsf}{\sffamily}}
\fi

```

\@mem@ttwarn The old typewriter font command.
\tt
```
\if@memoldfont
  \def\@mem@ttwarn{\@memoldfontwarn{\tt}{\texttt}{\ttfamily}}
  \DeclareOldFontCommand{\tt}{\@mem@ttwarn\gdef\@mem@ttwarn{}%
    \normalfont\ttfamily}{\mathtt}
\else
  \def\tt{\@memoldfonterr{\tt}{\texttt}{\ttfamily}}
\fi
```

\@mem@bfwarn The old bold font command.
\bf
```
\if@memoldfont
  \def\@mem@bfwarn{\@memoldfontwarn{\bf}{\textbf}{\bfseries}}
  \DeclareOldFontCommand{\bf}{\@mem@bfwarn\gdef\@mem@bfwarn{}%
    \normalfont\bfseries}{\mathbf}
\else
  \def\bf{\@memoldfonterr{\bf}{\textbf}{\bfseries}}
\fi
```

\@mem@itwarn The old italic font command.
\it
```
\if@memoldfont
  \def\@mem@itwarn{\@memoldfontwarn{\it}{\textit}{\itshape}}
  \DeclareOldFontCommand{\it}{\@mem@itwarn\gdef\@mem@itwarn{}%
    \normalfont\itshape}{\mathit}
\else
  \def\it{\@memoldfonterr{\it}{\textit}{\itshape}}
\fi
```

\@mem@slwarn The old slanted font command.
\sl
```
\if@memoldfont
  \def\@mem@slwarn{\@memoldfontwarn{\sl}{\textsl}{\slshape}}
  \DeclareOldFontCommand{\sl}{\@mem@slwarn\gdef\@mem@slwarn{}%
    \normalfont\slshape}{\@nomath\sl}
\else
  \def\sl{\@memoldfonterr{\sl}{\textsl}{\slshape}}
\fi
```

\@mem@scwarn The old small caps font command.
\sc
```
\if@memoldfont
  \def\@mem@scwarn{\@memoldfontwarn{\sc}{\textsc}{\scshape}}
  \DeclareOldFontCommand{\sc}{\@mem@scwarn\gdef\@mem@scwarn{}%
    \normalfont\scshape}{\@nomath\sc}
\else
  \def\sc{\@memoldfonterr{\sc}{\textsc}{\scshape}}
\fi
```

\@mem@calwarn The old calligraphic font command.
\cal
```
\if@memoldfont
  \def\@mem@calwarn{%
    \@memwarn{The \protect\cal\space font command is deprecated.
    \MessageBreak Try to use \protect\mathcal\space instead}}
  \DeclareRobustCommand*\cal{\@mem@calwarn\gdef\@mem@calwarn{}%
    \@fontswitch\relax\mathcal}
\else
  \def\cal{%
    \@mememror{Font command \protect\cal\space is not supported}{%
    Use \protect\mathcal, or the oldfontcommands option}}
\fi

```

\@mem@mitwarn The old math italic font command.
\mit
```
\if@memoldfont
  \def\@mem@mitwarn{%
    \@memwarn{The \protect\mit\space font command is deprecated.
    \MessageBreak Try to use \protect\mathnormal\space instead}}
  \DeclareRobustCommand*\mit{\@mem@mitwarn\gdef\@mem@mitwarn{}%
    \@fontswitch\relax\mathnormal}
\else
  \def\mit{%
  \@mememror{Font command \protect\mit\space is not supported}{%
             Use \protect\mathnormal, or the oldfontcommands option}}
\fi

```

\em The old emphasis font command (the original \em is defined in the kernel file
\eminnershape ltfssini.dtx, coded here as \@m@m@m).
\emph
```
\DeclareRobustCommand{\em}{%
  \@nomath\em
  \ifdim\fontdimen\@ne\font > \z@
    \eminnershape
  \else
    \itshape
  \fi}
\providecommand{\eminnershape}{\upshape}
\DeclareTextFontCommand{\emph}{\em}

```

21 Cross Referencing

21.1 Label referencing

\fref These are named references to labeled figures, tables and pages. I find these
\tref useful to ensure consistency throughout the document — I don't have to
\pref remember whether it is 'see Figure ...' or 'figure' or 'Fig.' or ....

```
\newcommand*{\fref}[1]{\figurerefname~\ref{#1}}
\newcommand*{\tref}[1]{\tablerefname~\ref{#1}}
\newcommand*{\pref}[1]{\pagerefname~\pageref{#1}}
```

\Aref
\Bref
\Pref
\Cref
\Sref

These are named references to labeled Part, Chapter and Sectional divisions.

```
\newcommand*{\Aref}[1]{\appendixrefname\ref{#1}}
\newcommand*{\Bref}[1]{\bookrefname\ref{#1}}
\newcommand*{\Pref}[1]{\partrefname\ref{#1}}
\newcommand*{\Cref}[1]{\chapterrefname\ref{#1}}
\newcommand*{\Sref}[1]{\sectionrefname\ref{#1}}
```

21.2 Title referencing

This is based on DA's titleref package [Ars01a]. The following remarks are taken from that package.

Titles for numbered sectioning units and for floats with captions are the same as the respective TOC, LOF, or LOT entry (even when no TOC is printed). When a short title is provided (`\section[short]{long}`) it will be used for the `\titleref`. This is especially useful for figure captions. Unnumbered sections take their title reference from the printed title. Beware! This turns the title into a moving argument when it normally is not, and will cause weird errors if there are 'fragile' commands present.

Enumerated lists do not have titles and simply inherit the title of their section.

The format of the title reference is controlled by the command `\theTitleReference`, which can be redefined with `\renewcommand`. It takes two parameters: the number and the title. The number is just the regular `\ref` and it is WRONG in unnumbered sections. Beware! The default definition is the unadorned title. You could do, for example, `\renewcommand{\theTitleReference}[2]{#1\ \emph{#2}}`.

The title of the current section is also available without `\label`: Use the command `\currenttitle` to generate a `\titleref` to the current section. If you have redefined `\theTitleReference` to print the number with the title, be aware that the two may not correspond: In a numbered list the number will show the current item but the title will show the current section, but in an unnumbered section the number will show some previous section number.

The big problem with the `[usetoc]` method is that `\titleref` will not refer to a `\label` which was given in the title or caption itself; the label must be placed after the sectioning command or the caption. If you make a title-reference to a label given in a title, you will get a warning message.

\ifheadnameref
\headnameref
\tocnameref

`\ifheadnameref`: FALSE use ToC entry, TRUE use header entry

```
\newif\ifheadnameref
\newcommand*{\headnameref}{\headnamereftrue}
\newcommand*{\tocnameref}{\headnamereffalse}
  \tocnameref
```

\theTitleReference `\theTitleReference{⟨num⟩}{⟨title⟩}` is the style for typesetting a referenced (number and) title.

```
\newcommand{\theTitleReference}[2]{#2}
```

\label This redefinition of `\label` is intended to work with other redirections of `\label`, if they record extra information in a similar way.

```
\let\@mem@old@label\label
\def\label#1{\@bsphack\begingroup
  \protected@edef\@currentlabel{\protect\M@TitleReference
      {\@currentlabel}{\M@currentTitle}}%
  \@mem@old@label{#1}%
  \endgroup \@esphack}%
```

\@mem@@gettitle `\@mem@@gettitle{⟨title⟩}` grabs a title text.

```
\def\@mem@@gettitle#1{\begingroup \let\protect\@unexpandable@protect
    \let\label\@mem@nestwarn
    \let\index\@gobble \let\glossary\@gobble
    \let\markboth\@gobbletwo \let\@mkboth\@gobbletwo
    \let\markright\@gobble
    \edef\@tempa{\noexpand\def\noexpand\M@currentTitle{#1}}%
  \expandafter\endgroup\@tempa}
```

\@mem@nestwarn
\M@TitleReference

```
\let\@mem@nestwarn\@gobble
\let\M@TitleReference\@firstoftwo
```

\titleref `\titleref{⟨key⟩}` prints the title corresponding to `\label{key}`.
\currenttitle `\currenttitle` prints the latest title. `\titleref` extended to support a starred
\@mem@titleref version that will not provide hyperlinks when using hyperref. The hyperref
\@mem@titlerefnolink definition of `\@mem@titlerefnolink` is found in memhfixc.

```
\newcommand*\@mem@titleref[1]{\begingroup
  \let\numberline\@gobble
  \let\M@TitleReference\@mem@theTR % interrupt recursion of \ref
  \ref{#1}\endgroup}
\let\@mem@titlerefnolink\@mem@titleref

\DeclareRobustCommand\titleref{\@ifstar{\@mem@titlerefnolink}{\@mem@titleref}}

\DeclareRobustCommand{\currenttitle}{\begingroup
  \let\numberline\@gobble
  \theTitleReference\@currentlabel\M@currentTitle\endgroup}

% \DeclareRobustCommand\@mem@nestwarn[1]{\@memwarn%
%  {Label \string"#1\string" was put in a title,\MessageBreak
```

```
  %   so the \noexpand\titleref is incorrect}}
```

\M@currentTitle

```
8181 \let\M@currentTitle\@empty
8182
```

\@mem@theTR proper definition:

```
8183 \def\@mem@theTR{\let\M@TitleReference\@firstoftwo \theTitleReference}
8184
```

\namerefon
\namerefoff Named references has turned some arguments (e.g., \legend{text}) into moving ones (Sven.Hartrumpf@FernUni-Hagen.de reported a problem but not a cause, March 2003). Not everyone needs named references.

```
8185 \newcommand*{\namerefon}{\let\M@gettitle\@mem@@gettitle}
8186 \newcommand*{\namerefoff}{\let\M@gettitle\@gobble}
8187   \namerefon
```

22 Table of Contents, etc.

A \section command writes a \contentsline{section}{⟨*title*⟩}{⟨*page*⟩} command on the .toc file, where ⟨*title*⟩ contains the contents of the entry and ⟨*page*⟩ is the page number. If sections are being numbered, then ⟨*title*⟩ will be of the form \numberline{⟨*num*⟩}{⟨*heading*⟩} where ⟨*num*⟩ is the number produced by \thesection. Other sectioning commands work similarly.
A \caption command in a 'figure' environment writes \contentsline{figure}{\numberline{⟨*num*⟩}{⟨*caption*⟩}}{⟨*page*⟩} on the .lof file, where ⟨*num*⟩ is the number produced by \thefigure and ⟨*caption*⟩ is the figure caption. It works similarly for a 'table' environment.
The command \contentsline{⟨*name*⟩} expands to \l@⟨*name*⟩. So, to specify the table of contents, we must define \l@chapter, \l@section, \l@subsection, ...; to specify the list of figures, we must define \l@figure; and so on. Most of these can be defined with the \@dottedtocline command, which works as follows.
\@dottedtocline{⟨*level*⟩}{⟨*indent*⟩}{⟨*numwidth*⟩}{⟨*title*⟩}{⟨*page*⟩}

⟨***level***⟩ An entry is produced only if⟨*level*⟩ <= value of the *tocdepth* counter. Note, \chapter is level 0, \section is level 1, etc.

⟨***indent***⟩ The indentation from the outer left margin of the start of the contents line.

⟨***numwidth***⟩ The width of a box in which the section number is to go, if ⟨*title*⟩ includes a \numberline command.

\@pnumwidth
\@tocrmarg
\@dotsep This command uses the following three parameters, which are set with a \newcommand (so em's can be used to make them depend upon the font).

\@pnumwidth The width of a box in which the page number is put.

\@tocrmarg The right margin for multiple line entries. One wants \@tocrmarg ≥ \@pnumwidth

\@dotsep Separation between dots, in mu units. Should be defined as a number like 2 or 1.7

```
\newcommand{\@pnumwidth}{1.55em}
\newcommand{\@tocrmarg} {2.55em}
\newcommand{\@dotsep}{4.5}
```

\tocentryskip \tocbaseline \tocskip We define two lengths and a utility command.

```
\newlength{\tocentryskip} \setlength{\tocentryskip}{1em}
\newlength{\tocbaseline} \setlength{\tocbaseline}{20pt}
\newcommand{\tocskip}[1]{%
    \addtocontents{toc}{\protect\vspace{#1}}}
```

22.1 New List of ...

It is apparent that users want to do at least two things that are not readily provided by the standard classes: (a) change the appearance of the Table of Contents, etc., headings, and (b) create new List of.... This class provides a means of creating new Lists whose headings are parameterized, thereby killing two birds with one stone.
In the standard classes the tables of contents, figures etc. are always set in single-column style. In this class you can choose one- or two-column ToCs, etc. The titles are added to the ToC, unless the starred versions of the commands are used.

\ensureonecol \restorefromonecol These two macros cooperate to switch from two-columns to one column, and back again.

\onecoltocetc \twocoltocetc \doccoltocetc These macros define \ensureonecol and \restorefromonecol such that

- \onecoltocetc ToCs, etc., will be set in one column (the default)
- \twocoltocetc ToCs, etc., will be set in two columns
- \doccoltocetc ToCs, etc., will be set in column(s) corresponding to the document's (onecolumn/twocolumn) option.

```
\newcommand*{\onecoltocetc}{%
  \def\ensureonecol{%
    \if@twocolumn
      \@restonecoltrue\onecolumn
    \else
      \@restonecolfalse
    \fi}%
  \def\restorefromonecol{\if@restonecol\twocolumn\fi}}
```

```
\newcommand*{\twocoltocetc}{%
  \def\ensureonecol{%
    \if@twocolumn
      \@restonecoltrue
    \else
      \@restonecolfalse\twocolumn
    \fi}%
  \def\restorefromonecol{\if@restonecol\else\onecolumn\fi}}
\newcommand*{\doccoltocetc}{%
  \let\ensureonecol\relax
  \let\restorefromonecol\relax}
```

Set the default

```
\onecoltocetc
```

`\cftparskip` The `\parskip` local to the ToC, etc, is set to the length `\cftparskip`.

```
\newlength{\cftparskip}
\setlength{\cftparskip}{0pt}
```

`\newlistof` `\newlistof{`⟨*listofcmmd*⟩`}{`⟨*ext*⟩`}{`⟨*listofname*⟩`}` creates the command `\listofcmmd` to typeset a new List of, where the external file has the extension `.ext` and the heading title is ⟨*listofname*⟩. The code for this is a heavily modified part of the tocloft package.

```
\newcommand{\newlistof}[3]{%
```

In the following, `X` stands for the value of ⟨*listofcmmd*⟩ and `Z` stands for the value of ⟨*ext*⟩.

`\ext@Z` The file extension and listing depth, which is set to level 1.

`\Zdepth`

```
  \@namedef{ext@#2}{#2}
  \@ifundefined{c@#2depth}{\newcounter{#2depth}}{}
  \setcounter{#2depth}{1}
```

`\Zmark` The heading marks for the listing.

```
  \@namedef{#2mark}{\markboth{#3}{#3}}
```

`\X` Typeset the listing title and entries, with both a normal and starred version.

```
  \@namedef{#1}{\@ifstar{\@nameuse{mem@#1}{01}}{\@nameuse{mem@#1}{00}}}
```

`\mem@X` `\cftZbeforelisthook` `\cftZafterlisthook` In earlier version of the class, we used two macros `\@starZ` and `\@plainZ`, to reduce code we now only use one, the argument of it takes care of differentiating between the stuff that are different. Two hooks is added just before and after importing the list file contents.

```
  \@namedef{cft#2beforelisthook}{}%
  \@namedef{cft#2afterlisthook}{}%
  \@namedef{mem@#1}##1{%
    \ensureonecol
```

```
    \par
    \begingroup
      \@nameuse{@#2maketitle}
      \if##1
        \ifmem@em@starred@listof\else
          \phantomsection
          \addcontentsline{toc}{chapter}{#3}
        \fi
      \fi
      \parskip\cftparskip
      \@nameuse{cft#2beforelisthook}%
      \@starttoc{#2}%
      \@nameuse{cft#2afterlisthook}%
    \endgroup
    \restorefromonecol}
```

`\@Zmaketitle` This macro typesets the title.

```
  \@namedef{@#2maketitle}{%
    \@nameuse{#2headstart}
   {\parindent\z@
%%%%  \parskip\cftparskip
    \interlinepenalty\@M
    \@nameuse{print#2nonum}%
    \@nameuse{print#2title}{#3}%
    \@nameuse{#2mark}%
    \thispagestyle{chapter}%
    \@nameuse{after#2title}
   }
    \@afterheading}
```

`\Zheadstart` `\afterZtitle` The macros `\Zheadstart` and `\afterZtitle` control what goes before and after the title. They default to the corresponding macros for chapters.

```
  \@namedef{#2headstart}{\chapterheadstart}
  \@namedef{after#2title}{\afterchaptertitle}
```

`\printZnonum` This typesets something before the title and defaults to `\printchapternonum`.

```
  \@namedef{print#2nonum}{\printchapternonum}
```

`\printZtitle` The title is typeset by the macro `\printZtitle{`⟨*title*⟩`}` which defaults to `\printchaptertitle`.

```
  \@namedef{print#2title}##1{\printchaptertitle{##1}}
```

This is the end of the definition of `\newlistof`.

```
} % end \newlistof

```

`\ifmem@em@starred@listof` `\KeepFromToc` The class handles the created 'new lists' a little different than other classes. We automatically add it to the main table of contents. This us usually what one

wants. If not, one can always use the starred version. But in some cases other packages will use `\tableofcontents` to typeset their own 'list of...', in which case it becomes a little hard to add a star. Therefore we add a switch that can be used to emulate the starred list of. After the macro `\KeepFromToc` the memoir created (non-starred) 'list of' will behave as their starred counterpart.

```
\newif\ifmem@em@starred@listof
\newcommand\KeepFromToc{\mem@em@starred@listoftrue}
```

Tip: use `\KeepFromToc` as an environment, i.e.

```
\begin{KeepFromToc}
 \listoffigures
\end{KeepFromToc}
```

to locally remove a list from the TOC.

`\@starttoc` A list of macro calls `\@starttoc` to read the appropriate file. I have changed the kernel definition to allow a file to be read multiple times by delaying killing the file until the end of the document.

```
\renewcommand{\@starttoc}[1]{%
  \begingroup\makeatletter
    \@input{\jobname.#1}%
    \if@filesw
      \AtEndDocument{%
```

If a ToC, or other ListOf, is called more than once then we have to stop opening yet another, redundant, output file. The check against `\relax` seems to meet the requirement.

```
        \expandafter\ifx\csname tf@#1\endcsname\relax
          \expandafter\newwrite\csname tf@#1\endcsname
          \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
        \fi
      }%
    \fi
  \@nobreakfalse
  \endgroup}

```

22.2 Table of Contents

`\tableofcontents` These macros request that LaTeX produces a table of contents. The ToC heading
`\tableofcontents*` is added to the ToC unless the starred version is used.

```
\newlistof{tableofcontents}{toc}{\contentsname}
```

The List of Figures and List of Tables are defined later.

22.3 List entries

Each command in the body of the text that makes an entry in the ToC, or LoF or LoT needs an addtional macro to format the entry, as described above. Users often want to change the formatting of the entries but it is not immediately obvious how to do that.
I have borrowed and modified more of the code from the tocloft package to enable easy creation and modification the formatting of the entries.

`\setpnumwidth` `\setrmarg` User commands for setting `\@pnumwidth` and `\@tocrmarg`.

```
\newcommand*{\setpnumwidth}[1]{\renewcommand{\@pnumwidth}{#1}}
\newcommand*{\setrmarg}[1]{\renewcommand{\@tocrmarg}{#1}}
```

`\cftdot` `\cftdotfill` In the default ToC, a dotted line can be used to provide a leader between a title and the page number. The definition of this leader is buried in the `\@dottedtocline` command. The `\cftdotfill{`⟨*sep*⟩`}` command provides a parameterised version of the leader code, where ⟨*sep*⟩ is the separation between the dots in mu units. The symbol used for the 'dots' in the leader is given by the value of `\cftdot`.

```
\providecommand{\cftdot}{.}
\providecommand{\cftdotfill}[1]{%
  \leaders\hbox{$\m@th\mkern #1 mu\hbox{\cftdot}\mkern #1 mu$}\hfill}
```

`\cftdotsep` `\cftnodots` `\cftdotsep` holds the default dot separation. If the kerns in `\cftdotfill` are large enough, then no dots will be printed. `\cftnodots` should be 'large enough'.

```
\providecommand{\cftdotsep}{4.5}
\newcommand{\cftnodots}{2000}
```

Now for the trickier bits regarding the typesetting of the ToC entries.
A `.toc` (also `.lof` and `.lot`) file consists of a list of
`\contentsline{`⟨*kind*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`
commands, where ⟨*kind*⟩ is the kind of heading (e.g., `part` or `section` or `figure`), ⟨*title*⟩ is the title text (including the number), and ⟨*page*⟩ is the page number. The entries are inserted into the file by calling the
`\addcontentsline{`⟨*file*⟩`}{`⟨*kind*⟩`}{`⟨*title*⟩`}`
command, where ⟨*file*⟩ is the file extension (e.g., `toc`, `lot`) and the other arguments are the same as for the `\contentsline` command. (Arbitrary stuff may also be put into the file via the `\addtocontents{`⟨*file*⟩`}{`⟨*text*⟩`}` command). The typesetting of the `\contentsline` entries is performed by commands of the form `\l@kind`. The sectioning and captioning commands call `\addcontentsline` to insert their titles into the `.toc` etc., files.
For the purposes at hand it is generally impossible to treat the typesetting of a title and its number separately, as both are bundled into the ⟨*title*⟩ argument within `\contentsline`. They could be handled separately if the `\contentsline` command was suitably modified. If this was done, then the `\addtocontentsline` command would also need to be changed which would then require the sectioning and captioning commands to be modified as well. This is certainly possible, but

would cause problems if any other package also modified the sectioning or captioning commands, and there are several packages which do this.
I provide modified versions of the `\l@kind` commands. Essentially, my new definitions consist of inlined versions of the code for `\@dottedtocline`.

`\cftparfillskip` The `\l@kind` commands modify (locally) the value of `\parfillskip`. `\cftparfillskip` is a copy of the default *The TEXbook* `\parfillskip` definition.

```
\newcommand*{\cftparfillskip}{\parfillskip=0pt plus1fil}
```

`\@cftn@me` Lars Madsen suggested that macros like `\cftfigurename` be added to entries in the LoF, and so on.

```
\newcommand*{\@cftn@me}{}

```

`\numberline` `\numberlinehook` The purpose of the `\numberline{`⟨*secnum*⟩`}` command is to typeset ⟨*secnum*⟩ left justified in a box of width `\@tempdima`. I redefine it to add three additional parameters, namely `\@cftbsnum`, `\@cftasnum` and `\@cftasnumb` (see `ltsect.dtx` for the original definition). We also add a hook, that initially does nothing, but might be redefined to record say withs of ⟨*secnum*⟩s.

```
\newcommand*\numberlinehook[1]{}
\renewcommand*{\numberline}[1]{%
  \numberlinehook{#1}%
  \hb@xt@\@tempdima{\@cftn@me\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}
```

`\@cftbsnum` `\@cftasnum` `\@cftasnumb`

```
\newcommand{\@cftbsnum}{}
\newcommand{\@cftasnum}{}
\newcommand{\@cftasnumb}{}
```

`\newlistentry` `\newlistentry[`⟨*within*⟩`]{`⟨*counter*⟩`}{`⟨*ext*⟩`}{`⟨*level-1*⟩`}` creates a set of commands for typesetting a new kind of entry in a List of. ⟨*counter*⟩ is the name of the counter for the entry and must be the same as the name of the entry (e.g., `subsection`). The optional ⟨*within*⟩ is the name of a counter within which `counter` is defined. The file extension for the List of is ⟨*ext*⟩ and ⟨*level-1*⟩ is one less than the level of the entry in the List of.

```
\newcommand{\newlistentry}[4][\@empty]{%
```

In the following, `X` is used as the value of ⟨*counter*⟩ and `Z` as the value of ⟨*ext*⟩.

`\c@X` `\theX` Check if ⟨*within*⟩ and ⟨*counter*⟩ have been defined. It is an error if ⟨*within*⟩ has not been defined. ⟨*counter*⟩ will be created if it has not been previously defined. Set the default counter values.

```
  \@ifundefined{c@#2}{%    check & set the counter
    \ifx \@empty#1\relax
      \newcounter{#2}
    \else
      \newcounter{#2}[#1]%
      \expandafter\edef\csname the#2\endcsname{%
```

```
\expandafter\noexpand\csname the#1\endcsname.\noexpand\arabic{#2}}
  \fi}{}
\setcounter{#2}{0}
```

That finishes off the error checking. No matter what the result, the rest of the new commands are defined.

`\l@X` `\l@X{`⟨*title*⟩`}{`⟨*page*⟩`}` typesets the entry.

```
\@namedef{l@#2}##1##2{%
```

Only typeset if the `\Zdepth` is greater than ⟨*level-1*⟩.

```
  \ifnum \@nameuse{c@#3depth} > #4\relax
```

Add some vertical space.

```
    \vskip \@nameuse{cftbefore#2skip}
```

Start a group to keep paragraphing changes local. Set the `\leftskip` to the entry's indentation.

`\cftwhatismyname` `\cftwhatismyname` locally stores the type of toc entry. Useful for hooks into general macros like `\numberline`.

```
    {%\leftskip \@nameuse{cft#2indent}\relax
     \newcommand*\cftwhatismyname{#2}%
     \memRTLleftskip \@nameuse{cft#2indent}\relax
```

Set the `\rightskip` to `\@tocrmarg` to leave room for the page number.

```
%%%     \rightskip \@tocrmarg
    \memRTLrightskip \@tocrmarg
```

Ensure that the last line of an entry will be filled. Setting `\parfillskip` to a negative value prevents 'overfull box' messages.

```
%%%     \parfillskip -\rightskip
    \parfillskip -\memRTLrightskip
```

Set the paragraph's indentation to the entry's indentation.

```
    \parindent \@nameuse{cft#2indent}\relax\@afterindenttrue
```

Try and prevent breaks between lines in a multiline entry.

```
    \interlinepenalty\@M
```

Make sure we have left vertical mode.

```
    \leavevmode
```

Our version of `\numberline` expects that the width of the number box is in `\@tempdima`, and that the three macros `\@cftbsnum`, `\@cftaqsnum`, and `\@cftasnumb` are defined. We set all these to the values for this entry.

```
    \settowidth{\@tempdima}{\@nameuse{cft#2font}\@nameuse{cft#2name}}%
    \addtolength{\@tempdima}{\@nameuse{cft#2numwidth}}%
\expandafter\let\expandafter\@cftbsnum\csname cft#2presnum\endcsname
\expandafter\let\expandafter\@cftasnum\csname cft#2aftersnum\endcsname
\expandafter\let\expandafter\@cftasnumb\csname cft#2aftersnumb\endcsname
\expandafter\let\expandafter\@cftn@me\csname cft#2name\endcsname
```

Arrange that the (entry number and) first line of the title is set at the current indent, and that any subsequent lines will be further indented.

```
%%%       \advance\leftskip\@tempdima \null\nobreak\hskip -\leftskip
      \advance\memRTLleftskip\@tempdima \null\nobreak\hskip -\memRTLleftskip
```

Print the (number and) title, prohibiting any breaking.

```
      {\@nameuse{cft#2font}##1}\nobreak
```

Print the leader and the page number, and then close the group.

```
      \@nameuse{cft#2fillnum}{##2}}
    \fi
  }% end of \l@#2

```

Now define all the layout commands used by `\l@X`. The default values of these print the entry in a normal font with a dotted line between the title and the page number.

`\cftbeforeXskip` The skip before the title.

```
  \expandafter\newlength\csname cftbefore#2skip\endcsname
    \setlength{\@nameuse{cftbefore#2skip}}{\z@ \@plus .2\p@}
```

`\cftXindent` `\cftXnumwidth` The indent and width for the number.

```
  \expandafter\newlength\csname cft#2indent\endcsname
  \expandafter\newlength\csname cft#2numwidth\endcsname
```

Set the default values for the indent and numwidth depending on the entry's level. A level of 1 corresponds to a figure entry (no indent, and space for a number like N.N).

```
  \ifcase #4\relax  % 0   (level 1)
    \setlength{\@nameuse{cft#2indent}}{0em}
    \setlength{\@nameuse{cft#2numwidth}}{2.3em}
  \or               % 1   (level 2)
    \setlength{\@nameuse{cft#2indent}}{2.3em}
    \setlength{\@nameuse{cft#2numwidth}}{3.2em}
  \or               % 2   (level 3)
    \setlength{\@nameuse{cft#2indent}}{5.5em}
    \setlength{\@nameuse{cft#2numwidth}}{4.1em}
  \or               % 3   (level 4)
    \setlength{\@nameuse{cft#2indent}}{8.5em}
    \setlength{\@nameuse{cft#2numwidth}}{5.0em}
  \else             % anything else
    \setlength{\@nameuse{cft#2indent}}{10.5em}
    \setlength{\@nameuse{cft#2numwidth}}{6.0em}
  \fi
```

`\cftXfont` `\cftXname` And the remaining commands; the only ones that are not null are for the dotsep, the font and the leader

```
  \@namedef{cft#2font}{\normalfont}
  \@namedef{cft#2name}{}
```

`\cftXpresnum`
`\cftXaftersnum`
`\cftXaftersnumb`
`\cftXdotsep`
`\cftXleader`
`\cftXpagefont`
`\cftXafterpnum`

```
\@namedef{cft#2presnum}{}
\@namedef{cft#2aftersnum}{}
\@namedef{cft#2aftersnumb}{}
\@namedef{cft#2dotsep}{\cftdotsep}
\@namedef{cft#2leader}{\normalfont\cftdotfill{\@nameuse{cft#2dotsep}}}
\@namedef{cft#2pagefont}{\normalfont}
\@namedef{cft#2afterpnum}{}
```

`\toclevel@X` The hyperref package needs a command `\toclevel@X`, holding the ⟨*level-1*⟩ value.

```
\@namedef{toclevel@#2}{#4}
```

`\cftXformatpnum` `\cftXformatpnumhook` `\cftXfillnum` Typeset the leader and page number. We add a hook into the formating of the page number. This might later be used to record the widths of the numbers used. It will be given the page number as its argument. Does nothing by default.

```
\@namedef{cft#2formatpnumhook}##1{}
\@namedef{cft#2formatpnum}##1{%
  \@nameuse{cft#2formatpnumhook}{##1}%
  \hb@xt@\@pnumwidth{\hfil\@nameuse{cft#2pagefont}##1}}
\@namedef{cft#2fillnum}##1{%
  {\@nameuse{cft#2leader}}\nobreak
%%%   \hb@xt@\@pnumwidth{%
%%%     \hfil\@nameuse{cft#2pagefont}##1}
      \@nameuse{cft#2formatpnum}{##1}%
      \@nameuse{cft#2afterpnum}\par}
```

This ends the definition of `\newlistentry`.

```
} % end \newlistentry

```

`\cftsetindents` `\cftsetindents{`⟨*entry*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}` sets the *indent* and *numwidth* for entry ⟨*entry*⟩.

```
\newcommand*{\cftsetindents}[3]{%
  \setlength{\@nameuse{cft#1indent}}{#2}
  \setlength{\@nameuse{cft#1numwidth}}{#3}}

```

`\cftbookname` If you want a book entry in the ToC like:
BOOK I Title ...
then `\renewcommand*{\cftbookname}{BOOK~}`

```
\newcommand*{\cftbookname}{}

```

`\cftbookbreak` Now for the ToC entry.

`\l@book` `\booknumberline` `\booknumberlinehook`

```
\newcommand*{\cftbookbreak}{\addpenalty{-\@highpenalty}%
  \addvspace{\cftbeforebookskip}}
\newcommand*{\l@book}[2]{%
  \ifnum\c@tocdepth >-3\relax
```

```
    \cftbookbreak
    \begingroup
      {%\leftskip \cftbookindent\relax
       \memRTLleftskip \cftbookindent\relax
%%        \rightskip \@tocrmarg
       \memRTLrightskip \@tocrmarg
%%        \parfillskip -\rightskip
       \parfillskip -\memRTLrightskip
       \parindent \cftbookindent\relax\@afterindenttrue
       \interlinepenalty\@M
       \leavevmode
       \settowidth{\@tempdima}{\cftbookfont\cftbookname}%
       \addtolength{\@tempdima}{\cftbooknumwidth}%
       \let\@cftbsnum \cftbookpresnum
       \let\@cftasnum \cftbookaftersnum
       \let\@cftasnumb \cftbookaftersnumb
%%        \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
       \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
       {\cftbookfont #1}%
       \cftbookfillnum{#2}}
      \nobreak
      \global\@nobreaktrue
      \everypar{\global\@nobreakfalse\everypar{}}%
    \endgroup
  \fi}
\newcommand*\booknumberlinehook[1]{}
\newcommand{\booknumberline}[1]{%
  \booknumberlinehook{#1}%
  \hb@xt@\@tempdima{%
    \cftbookname\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}%%\space}

\newlength{\cftbeforebookskip}
  \setlength{\cftbeforebookskip}{2.25em \@plus\p@}
\newdimen\cftbookindent
  \setlength{\cftbookindent}{0em}
\newdimen\cftbooknumwidth
  \setlength{\cftbooknumwidth}{1.5em}
\newcommand*{\cftbookfont}{\large\bfseries}
\newcommand*{\cftbookpresnum}{}
\newcommand*{\cftbookaftersnum}{}
\newcommand*{\cftbookaftersnumb}{}
\newcommand*{\cftbookleader}{%
  \large\bfseries\cftdotfill{\cftbookdotsep}}
\newcommand*{\cftbookdotsep}{\cftnodots}
\newcommand*{\cftbookpagefont}{\large\bfseries}
\newcommand{\cftbookafterpnum}{}
\newcommand{\cftbookfillnum}[1]{%
  {\cftbookleader}%
```

\cftbeforebookskip
\cftbookindent
\cftbooknumwidth
\cftbookfont
\cftbookpresnum
\cftbookaftersnum
\cftbookaftersnumb
\cftbookleader
\cftbookdotsep
\cftbookpagefont
\cftbookafterpnum
\cftbookfillnum
\cftbookformatpnum
\cftbookformatpnumhook

```
%%%%  {\hb@xt@\@pnumwidth{\hss {\cftbookpagefont #1}}}%
  \cftbookformatpnum{#1}%
  \cftbookafterpnum\par}
\newcommand{\cftbookformatpnumhook}[1]{}
\newcommand{\cftbookformatpnum}[1]{%
  \cftbookformatpnumhook{#1}%
  \hb@xt@\@pnumwidth{\hss {\cftbookpagefont #1}}}

```

`\cftpartname` If you want a part entry in the ToC like:
PART I Title ...
then `\renewcommand*{\cftpartname}{PART~}`

```
\newcommand*{\cftpartname}{}

```

`\cftpartbreak` `\l@part` Can't use `\newlistentry` for `\l@part` because of the initial penalty and the final `\nobreak` code.

```
\newcommand*{\cftpartbreak}{\addpenalty{-\@highpenalty}%
  \addvspace{\cftbeforepartskip}}
\newcommand*{\l@part}[2]{%
  \ifnum \c@tocdepth >-2\relax
    \cftpartbreak
    \begingroup
      {%\leftskip \cftpartindent\relax
       \memRTLleftskip \cftpartindent\relax
%%%       \rightskip \@tocrmarg
       \memRTLrightskip \@tocrmarg
%%%       \parfillskip -\rightskip
       \parfillskip -\memRTLrightskip
       \parindent \cftpartindent\relax\@afterindenttrue
       \interlinepenalty\@M
       \leavevmode
       \settowidth{\@tempdima}{\cftpartfont\cftpartname}%
       \addtolength{\@tempdima}{\cftpartnumwidth}%
       \let\@cftbsnum \cftpartpresnum
       \let\@cftasnum \cftpartaftersnum
       \let\@cftasnumb \cftpartaftersnumb
%%%       \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
       \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
       {\cftpartfont #1}%
       \cftpartfillnum{#2}}
      \nobreak
        \global\@nobreaktrue
        \everypar{\global\@nobreakfalse\everypar{}}%
    \endgroup
  \fi}

```

`\toclevel@part` Needed if the hyperref package is used.

```
\newcommand*{\toclevel@part}{-1}
```

\partnumberline
\partnumberlinehook

\partnumberline is a special version of \numberline output by \part. Its default definition is identical to \numberline.

```
\newcommand*\partnumberlinehook[1]{}
\newcommand{\partnumberline}[1]{%
  \partnumberlinehook{#1}%
  \hb@xt@\@tempdima{%
    \cftpartname\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}%%\space}
```

\cftbeforepartskip
\cftpartnumwidth
\cftpartfont
\cftpartpresnum
\cftpartaftersnum
\cftpartaftersnumb
\cftpartleader
\cftpartdotsep
\cftpartpagefont
\cftpartafterpnum
\cftpartindent
\cftpartformatpnum
\cftpartformatpnumhook
\cftpartfillnum

These are the user commands to control the typesetting of Part entries. They are initialised to give the standard appearance.

```
  \newlength{\cftbeforepartskip}
    \setlength{\cftbeforepartskip}{2.25em \@plus\p@}
  \newlength{\cftpartindent}
    \setlength{\cftpartindent}{0em}
  \newlength{\cftpartnumwidth}
    \setlength{\cftpartnumwidth}{1.5em}
  \newcommand{\cftpartfont}{\large\bfseries}
  \newcommand{\cftpartpresnum}{}
  \newcommand{\cftpartaftersnum}{}
  \newcommand{\cftpartaftersnumb}{}
  \newcommand{\cftpartleader}{%
              \large\bfseries\cftdotfill{\cftpartdotsep}}
  \newcommand{\cftpartdotsep}{\cftnodots}
  \newcommand{\cftpartpagefont}{\large\bfseries}
  \newcommand{\cftpartafterpnum}{}
  \newcommand{\cftpartformatpnumhook}[1]{}
  \newcommand*{\cftpartformatpnum}[1]{%
    \cftpartformatpnumhook{#1}%
    \hb@xt@\@pnumwidth{\hss {\cftpartpagefont #1}}}
  \newcommand{\cftpartfillnum}[1]{%
    {\cftpartleader}%
    {\cftpartformatpnum{#1}}%
     \cftpartafterpnum\par}
```

\cftchaptername

If you want a chapter entry in the ToC like:
Chapter 1. A title ...
then \renewcommand*{\cftchaptername}{\chaptername~}
\renewcommand*{\cftchapteraftersnum}{.}

```
\newcommand*{\cftchaptername}{}
```

\l@chapapp

Generic chapter/appendix ToC entry typesetting.

```
\newcommand*{\l@chapapp}[3]{%
  \ifnum \c@tocdepth >\m@ne
```

```
8508     \cftchapterbreak
8509     \vskip \cftbeforechapterskip
8510     {%\leftskip \cftchapterindent\relax
8511      \memRTLleftskip \cftchapterindent\relax
8512 %%%     \rightskip \@tocrmarg
8513      \memRTLrightskip \@tocrmarg
8514 %%%     \parfillskip -\rightskip
8515      \parfillskip -\memRTLrightskip
8516      \parindent \cftchapterindent\relax
8517      \@afterindenttrue
8518      \interlinepenalty\@M
8519      \leavevmode
8520      \let\@cftbsnum \cftchapterpresnum
8521      \let\@cftasnum \cftchapteraftersnum
8522      \let\@cftasnumb \cftchapteraftersnumb
8523      \def\@chapapp@head{#3}%
8524      \settowidth{\@tempdima}{\cftchapterfont\@chapapp@head}%
8525      \addtolength{\@tempdima}{\cftchapternumwidth}%
8526 %%%     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
8527      \advance\memRTLleftskip \@tempdima \null\nobreak\hskip -\memRTLleftskip
8528      {\cftchapterfont #1}\nobreak
8529      \cftchapterfillnum{#2}}
8530   \fi}
8531
```

`\l@chapter` `\l@chapter{⟨title⟩}{⟨page⟩}` typesets the ToC entry for a `chapter` heading. It is a parameterised copy of the default `\l@chapter` (see `classes.dtx` for the original definition). Can't use `\newlistentry` for this because of the intial penalty.

```
8532 \newcommand*{\l@chapter}[2]{%
8533   \l@chapapp{#1}{#2}{\cftchaptername}}
8534
```

`\toclevel@chapter` Needed if the hyperref package is used.

```
8535 \newcommand*{\toclevel@chapter}{0}
8536
```

`\cftappendixname` If you want a appendixed chapter entry in the ToC like:
Appendix A A title ...
then `\renewcommand*{\cftappendixname}{\appendixname~}`

```
8537 \newcommand*{\cftappendixname}{}
8538
```

`\l@appendix` Similar to `\l@chapter` but for an appendix.

`\toclevel@appendix`

```
8539 \newcommand*{\l@appendix}[2]{%
8540   \l@chapapp{#1}{#2}{\cftappendixname}}
8541 \newcommand{\toclevel@appendix}{0}
8542
```

\chapternumberline
\chapternumberlinehook

\chapternumberline is a special version of \numberline output by \chapter. It's default definition is identical to \numberline.

```
\newcommand*\chapternumberlinehook[1]{}
\newcommand{\chapternumberline}[1]{%
  \chapternumberlinehook{#1}%
  \hb@xt@\@tempdima{\@chapapp@head\@cftbsnum #1\@cftasnum\hfil}%
  \@cftasnumb}
```

\cftbeforechapterskip
\cftchapterindent
\cftchapternumwidth
\cftchapterfont
\cftchapterpresnum
\cftchapteraftersnum
\cftchapteraftersnumb
\cftchapterleader
\cftchapterdotsep
\cftchapterpagefont
\cftchapterafterpnum
\cftchapterformatpnum
\cftchapterformatpnumhook
\cftchapterfillnum

These are the user commands to control the typesetting of Chapter entries. They are initialised to give the standard appearance.

```
  \newlength{\cftbeforechapterskip}
    \setlength{\cftbeforechapterskip}{1.0em \@plus\p@}
  \newlength{\cftchapterindent}
    \setlength{\cftchapterindent}{0em}
  \newlength{\cftchapternumwidth}
    \setlength{\cftchapternumwidth}{1.5em}
  \newcommand{\cftchapterfont}{\bfseries}
  \newcommand{\cftchapterpresnum}{}
  \newcommand{\cftchapteraftersnum}{}
  \newcommand{\cftchapteraftersnumb}{}
  \newcommand{\cftchapterleader}{%
              \bfseries\cftdotfill{\cftchapterdotsep}}
  \newcommand{\cftchapterdotsep}{\cftnodots}
  \newcommand{\cftchapterpagefont}{\bfseries}
  \newcommand{\cftchapterafterpnum}{}
  \newcommand{\cftchapterformatpnumhook}[1]{}
  \newcommand*{\cftchapterformatpnum}[1]{%
    \cftchapterformatpnumhook{#1}%
    \hb@xt@\@pnumwidth{\hfil\cftchapterpagefont #1}}
  \newcommand*{\cftchapterfillnum}[1]{%
    {\cftchapterleader}\nobreak
    \cftchapterformatpnum{#1}%
    \cftchapterafterpnum\par}
```

\cftchapterbreak

Another parameter for \l@chapter.

```
  \newcommand{\cftchapterbreak}{\addpenalty{-\@highpenalty}}
```

\l@section
\l@subsection
\l@subsubsection
\l@paragraph
\l@subparagraph

\l@section{⟨*title*⟩}{⟨*page*⟩} typesets the ToC entry for a \section heading. Similarly \l@subsection and \l@subsubsection for \subsection and \subsubsection entries, and so on.

```
\newlistentry[chapter]{section}{toc}{0}
  \cftsetindents{section}{1.5em}{2.3em}
\newlistentry[section]{subsection}{toc}{1}
  \cftsetindents{subsection}{3.8em}{3.2em}
\newlistentry[subsection]{subsubsection}{toc}{2}
  \cftsetindents{subsubsection}{7.0em}{4.1em}
\newlistentry[subsubsection]{paragraph}{toc}{3}
```

```
  \cftsetindents{paragraph}{10.0em}{5.0em}
\newlistentry[paragraph]{subparagraph}{toc}{4}
  \cftsetindents{subparagraph}{12.0em}{6.0em}
```

The typesetting for Figures and Tables is defined later.

22.4 Support for the subfigure package

The code for supporting the subfigure package is, in all essentials, the same as that for the figure and table captions; only the names are changed. However, the code need only be executed if the subfigure package is actually loaded.

`\@cftl@subfigtab` This command redefines the `\l@subfigure` and `\l@subtable` commands.

```
\newcommand*{\@cftl@subfigtab}{
```

`\l@subfigure` `\l@subtable` `\l@subfigure{`⟨*title*⟩`}{`⟨*page*⟩`}` typesets the LoF entry for a subfigure caption heading, and `\l@subtable` does the same for subtables.

```
\newlistentry[figure]{subfigure}{lof}{1}
  \cftsetindents{subfigure}{2.3em}{2.5em}
\newlistentry[table]{subtable}{lot}{1}
  \cftsetindents{subtable}{2.3em}{2.5em}}
\renewcommand*{\@cftl@subfigtab}{}
```

Call the subfigure package setup code only if the subfigure package has been used.

```
\AtBeginDocument{\@ifpackageloaded{subfigure}{\@cftl@subfigtab}{}}
```

22.5 Switching page numbering

It can, at times, be useful to be able to have ToC entries that have no printed page numbers.

`\cftpagenumbersoff` The user level command for switching off page numbers is `\cftpagenumbersoff{`⟨*entry*⟩`}` where ⟨*entry*⟩ is the name of the entry. The macro redefines the `\cftXfillnum` command so that there is no leader and the page number is ignored.

```
\DeclareRobustCommand{\cftpagenumbersoff}[1]{%
  \@namedef{cft#1fillnum}##1{%
    \cftparfillskip\@nameuse{cft#1afterpnum}\par}}
```

`\cftpagenumberson` `\cftpagenumberson{`⟨*entry*⟩`}` is the user level command for reversing the corresponding `\cftpagenumbersoff`. The macro defines the `\cftXfillnum` command to correspond to the default definition.

```
\DeclareRobustCommand{\cftpagenumberson}[1]{%
  \@namedef{cft#1fillnum}##1{%
```

```
    \@nameuse{cft#1leader}\nobreak
    \@nameuse{cft#1formatpnum}{##1}%
    \@nameuse{cft#1afterpnum}\par}}
```

22.6 Chapter precis

`\chapterprecis` The command `\chapterprecis{`⟨*text*⟩`}` typesets ⟨*text*⟩ at the point where it is called, and also adds ⟨*text*⟩ to the `.toc` file. It is expects to be called immediately after a `\chapter` command.

```
\newcommand{\chapterprecis}[1]{%
  \chapterprecishere{#1}
  \chapterprecistoc{#1}}
```

`\chapterprecishere` `\chapterprecishere{`⟨*text*⟩`}` typesets ⟨*text*⟩. It expects to be called immediately after a `\chapter` command.

```
\newcommand{\chapterprecishere}[1]{%
  \prechapterprecis #1\postchapterprecis}
```

`\prechapterprecis` `\prechapterprecisshift` `\precisfont` `\postchapterprecis` The `\pre...` and `\post...` macros put code before and after `\chapterprecishere` text. By default `\prechapterprecis` adds some (negative) space (defined by `\prechapterprecisshift` whose value depends on whether or not the `article` option is used, as discovered by Lars Madsen) and starts a quote environment using the `\precisfont`. `\postchapterprecis` ends the quote environment.

```
\newdimen\prechapterprecisshift
\ifartopt
  \prechapterprecisshift=0pt
\else
  \prechapterprecisshift=-2\baselineskip
\fi
\newcommand*{\precisfont}{\normalfont\itshape}
\newcommand{\prechapterprecis}{%
  \vspace*{\prechapterprecisshift}%
  \begin{quote}\precisfont}
\newcommand*{\postchapterprecis}{\end{quote}}
```

`\precistocfont` Font for typesetting chapter precis in the ToC.

```
\newcommand{\precistocfont}{\normalfont\itshape}
```

`\chapterprecistoc` `\precistoctext` `\chapterprecistoc{`⟨*text*⟩`}` effectively adds ⟨*text*⟩ to the `.toc` file. The ⟨*text*⟩ will be typeset within the same margins as the the title text of a `\chapter` heading, using the `\precistocfont` font.

```
\newcommand{\chapterprecistoc}[1]{%
  \addtocontents{toc}{\precistoctext{#1}}}
\DeclareRobustCommand{\precistoctext}[1]{%
```

Start a group to localize changes to the paragraphing. Set the left margin to the chapter indent plus the chapter number width.

```
{%\nopagebreak\leftskip \cftchapterindent\relax
 \nopagebreak\memRTLleftskip \cftchapterindent\relax
%%%  \advance\leftskip \cftchapternumwidth\relax
 \advance\memRTLleftskip \cftchapternumwidth\relax
```

Set the right hand margin to `\@tocrmarg`.

```
%%%  \rightskip \@tocrmarg\relax
 \memRTLrightskip \@tocrmarg\relax
```

Typeset ⟨*text*⟩ using an italic font, then ensure that the paragraph is finished (to use the local skips). Finally close the group and we are done.

```
 \precistocfont #1\par}}
```

22.7 Adding things to the ToC

`\cftlocalchange` `\cftmakelocalchange{`⟨*file*⟩`}{`⟨*pnumwidth*⟩`}{`⟨*tocrmarg*⟩`}` makes an entry into ⟨*file*⟩ to change the `\@pnumwidth` and the `\@tocrmarg` values.

```
\newcommand{\cftlocalchange}[3]{%
 \addtocontents{#1}{\protect\setpnumwidth{#2} \protect\setrmarg{#3}}}
```

`\cftaddtitleline` `\cftaddtitleline{`⟨*file*⟩`}{`⟨*kind*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}` adds a `\contentsline` entry to ⟨*file*⟩ with the given information.

```
\newcommand{\cftaddtitleline}[4]{%
 \addtocontents{#1}{\protect\contentsline{#2}{#3}{#4}}}
```

`\cftaddnumtitleline` `\cftaddnumtitleline{`⟨*file*⟩`}{`⟨*kind*⟩`}{`⟨*num*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}` adds a `\contentsline` entry to ⟨*file*⟩ with the given information.

```
\newcommand{\cftaddnumtitleline}[5]{%
 \addtocontents{#1}%
   {\protect\contentsline{#2}{\protect\numberline{#3}%
                              {\protect\ignorespaces #4}}{#5}}}
```

`\cftinsertcode` `\cftinserthook` `\cftinsert` This is a generalisation of a suggestion by Lars Madsen (private email, 2007/12/14).

`\cftinsertcode{`⟨*id*⟩`}{`⟨*code*⟩`}` creates a hook to be executed in a 'List of'. Since we use `\@nameuse` it does not matter if the hook does not exist

`\cftinserthook{`⟨*list*⟩`}{`⟨*id*⟩`}` inserts the hook ⟨*id*⟩ into the ⟨*list*⟩ 'List of' file (in the form of `\cftinsert{`⟨*id*⟩`}`).

Use like this:

```
\cftinsertcode{A}{%
   \renewcommand*{\cftchapterfont}{\normalfont\scshape}
   ...
   }
\cftinsertcode{F}{...}
\cftinsertcode{G}{...}
   ...
```

```
\frontmatter
\tableofcontents
\cftinserthook{lof}{G}
\listoffigures
\chapter{...}
...
\mainmatter
\cftinserthook{lof}{F}
\cftinserthook{toc}{A}
\chapter{...}
...
```

```
\newcommand*{\cftinsert}[1]{\@nameuse{cftinsert#1}}
\newcommand{\cftinsertcode}[2]{\@namedef{cftinsert#1}{#2}}
\newcommand*{\cftinserthook}[2]{%
  \addtocontents{#1}{\protect\cftinsert\protect{#2\protect}}}

```

22.8 ToC and divisional numbering

Commands are provided, based on the tocvsec2 package, for changing the section numbering level and the ToC entry level.

`\@setclcnt` Helper macro to set a sectioning-related counter. Use as `\@setclcnt{`⟨*sec*⟩`}{`⟨*counter*⟩`}` to set *counter* to the level of ⟨*sec*⟩.

```
\newcommand*{\@setclcnt}[2]{%
  \def\@setclcntok{0}% = false
  \nametest{#1}{none}%
  \ifsamename
    \setcounter{#2}{-10}%
    \def\@setclcntok{1}% = true
  \fi
  \nametest{#1}{book}%
  \ifsamename
    \setcounter{#2}{-2}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{part}%
  \ifsamename
    \setcounter{#2}{-1}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{chapter}%
  \ifsamename
    \setcounter{#2}{0}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{section}%
  \ifsamename
```

```
    \setcounter{#2}{1}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{subsection}%
  \ifsamename
    \setcounter{#2}{2}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{subsubsection}%
  \ifsamename
    \setcounter{#2}{3}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{paragraph}%
  \ifsamename
    \setcounter{#2}{4}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{subparagraph}%
  \ifsamename
    \setcounter{#2}{5}%
    \def\@setclcntok{1}%
  \fi
  \nametest{#1}{all}%
  \ifsamename
    \setcounter{#2}{50}%
    \def\@setclcntok{1}%
  \fi
%%  \if@tempswa\else
    \ifnum \@setclcntok = 0\relax
    \@memerror{%
      Unknown document division name (#1)
    }{%
     I'll ignore it.
     Type \space <return> and I'll continue.\MessageBreak
     If you haven't mistyped the name then use
     \protect\setcounter\space instead.}%
  \fi}

```

\settocdepth \settocdepth{⟨*sec*⟩} is the user command for setting *tocdepth* in the .toc file to the value corresponding to ⟨*sec*⟩. .

```
\newcommand*{\settocdepth}[1]{%
  \def\@chtodok{0}%   false
  \nametest{#1}{none}%
  \ifsamename
    \addtocontents{toc}{\changetocdepth{-10}}%
    \def\@chtodok{1}%   true
  \fi
  \nametest{#1}{book}%
```

```
\ifsamename
  \addtocontents{toc}{\changetocdepth{-2}}%
  \def\@chtodok{1}%
\fi
\nametest{#1}{part}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{-1}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{chapter}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{0}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{section}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{1}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{subsection}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{2}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{subsubsection}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{3}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{paragraph}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{4}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{subparagraph}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{5}}%
  \def\@chtocdok{1}%
\fi
\nametest{#1}{all}%
\ifsamename
  \addtocontents{toc}{\changetocdepth{50}}%
  \def\@chtocdok{1}%
\fi
%  \if@tempswa
\ifnum\@chtocdok=1\relax
```

The next bit is from Heiko Oberdiek (CTT *Re: Memoir, \settocdepth and pdflatex => problem with PDF bookmarks*, 2006/07/21) as hyperref needs \tocdepth set in the body not just in the ToC.

```
    \@ifundefined{toclevel@#1}{%
      \@memwarn{Unknown toclevel for #1}%
    }{%
      \setcounter{tocdepth}{\@nameuse{toclevel@#1}}%
    }
  \else
    \@memerror{%
      Unknown document division name (#1)
    }{%
     I'll ignore it.
     Type \space <return> and I'll continue.}%
  \fi}
```

`\toclevel@none` `\toclevel@all` Couple of extras for hypperef to cater for `all` and `none` as 'division levels'.

```
\newcommand*{\toclevel@none}{-10}
\newcommand*{\toclevel@all}{50}
```

`\changetocdepth` Changes the `tocdepth` counter. Make it robust as it will be written to the *.toc file.

```
\DeclareRobustCommand{\changetocdepth}[1]{\setcounter{tocdepth}{#1}}
```

`\maxtocdepth` `\maxtocdepth{`⟨*sec*⟩`}` can be used to initialise *tocdepth* to the value corresponding to ⟨*sec*⟩. This can only be used between the end of the preamble and the `\tableofcontents` command.

```
\newcommand{\maxtocdepth}[1]{%
  \@setclcnt{#1}{tocdepth}}
```

`\maxsecnumdepth` `\maxsecnumdepth{`⟨*sec*⟩`}` can be used to initialise *secnumdepth* to the value corresponding to ⟨*sec*⟩.

```
\newcounter{maxsecnumdepth}
\newcommand{\maxsecnumdepth}[1]{%
  \@setclcnt{#1}{secnumdepth}\@setclcnt{#1}{maxsecnumdepth}}
```

`\setsecnumdepth` `\setsecnumdepth{`⟨*sec*⟩`}` is the user command for setting *secnumdepth* to the value for ⟨*sec*⟩. In the preamble it sets both the *secnumdepth* and *maxsecnumdepth* to ⟨*sec*⟩ while in the body it only sets *secnumdepth*.

```
\newcommand{\setsecnumdepth}[1]{%
  \ifx\@nodocument\relax%      after the preamble
    \@setclcnt{#1}{secnumdepth}%
  \else
    \@setclcnt{#1}{secnumdepth}%
    \@setclcnt{#1}{maxsecnumdepth}%
  \fi}
\setsecnumdepth{section}
```

23 Bibliography

\bibindent The open bibliography uses an indentation of \bibindent.

```
\newdimen\bibindent
  \setlength\bibindent{1.5em}
```

\bibitemsep The vertical separation between items in the bibliography list.

```
\newlength{\bibitemsep}
  \setlength{\bibitemsep}{\itemsep}
```

\biblistextra A hook into the bibitemlist.

```
\newcommand{\biblistextra}{\itemsep=\bibitemsep}

```

bibitemlist The thebibliography environment starts a new document division. Internally it tweaks some typesetting aspects; principally it uses \sloppy because good linebreaking is hard in a bibliography, and \sfcode‘\.=1000\relax causes a full stop not to produce an end-of-sentence space. The implementation of the environment is based on the generic list environment, and uses the \c@enumiv count for the labels. The following code is extracted from the book class, plus some additions.

```
\newenvironment{bibitemlist}[1]{%
  \typeout{bibitemlist}
  \list{\@biblabel{\@arabic\c@enumiv}}%
       {\settowidth\labelwidth{\@biblabel{#1}}%
        \leftmargin\labelwidth
        \advance\leftmargin\labelsep
        \@openbib@code
        \usecounter{enumiv}%
        \let\p@enumiv\@empty
        \renewcommand\theenumiv{\@arabic\c@enumiv}%
        \biblistextra}%
  \sloppy
  \clubpenalty4000
  \@clubpenalty \clubpenalty
  \widowpenalty4000%
  \sfcode‘\.\@m}%
  {\def\@noitemerr
    {\@latex@warning{Empty ‘thebibliography’ environment}}%
    \endlist}

```

\newblock The default is for \newblock to provide a small space.

```
\newcommand{\newblock}{\hskip .11em\@plus.33em\@minus.07em}
```

\@openbib@code This a an empty hook. It will be modified if the openbib option is used.

```
\let\@openbib@code\@empty
```

\setbiblabel
\@biblabel

This is the user command for setting the label for a \bibitem. The following sets the default definition.

```
\newcommand*{\setbiblabel}[1]{%
  \renewcommand*{\@biblabel}[1]{#1}}
\setbiblabel{[#1]\hfill}
```

\@memb@bchap
\@memb@bsec

In a CTT thread *memoir, natbib, and chapterbib* in January 2003 there was a discussion on how to get all three to work together. Donald Arseneau suggested that \bibsection be used as a 'standard' way of controlling the sectioning command of a bibliography. Here is an implementation of that idea. \@memb@bchap is a copy of memoir's original code for the start of the thebibliography environment which used a \chapter* title. \@memb@bsec is the much simpler version for a \section title.

```
\newcommand{\@memb@bchap}{%
  \chapter*{\bibname}%
  \bibmark
  \ifnobibintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\bibname}%
  \fi
  \prebibhook}
\newcommand{\@memb@bsec}{\section{\bibname}\prebibhook}
```

\bibsection

Normally treat the bibliography heading as a chapter.

```
\newcommand{\bibsection}{\@memb@bchap}
```

thebibliography

The definition of the thebibliography environment in this class is not quite the same as in the standard classes!

```
\newenvironment{thebibliography}[1]{%
  \bibsection
  \begin{bibitemlist}{#1}}{\end{bibitemlist}\postbibhook}
```

\ifnobibintoc
\bibintoc
\nobibintoc

Flag to control whether or not to add the bibliography title to the ToC, and declarations to set the flag. Default is to put the title into the ToC.

```
\newif\ifnobibintoc
\newcommand*{\bibintoc}{\nobibintocfalse}
\newcommand*{\nobibintoc}{\nobibintoctrue}
\bibintoc
```

\prebibhook
\postbibhook

These two macros are called just before starting the bib items and just after finishing them. By default they do nothing but can be changed by the user to give, say, some introductory information.

```
\newcommand{\prebibhook}{}
\newcommand{\postbibhook}{}
```

\@cite The output of the \cite command is produced by this macro. The default is used. The cite package is a good way of changing this.

```
% \renewcommand*{\@cite}[1]{[#1]}
```

23.1 Use with the natbib and chapterbib packages

The natbib package [Dal99] uses its own definition for the thebibliography environment, which knows nothing about adding the Bibliography to the ToC. The following makes appropriate changes to natbib code to support the class. The chapterbib package by Donald Arseneau also may make changes to the thebibliography environment — perhaps memoir's version or natbib's version. As packages get loaded after classes, I have to try and patch any non-memoir modifications at begin document time. The fixes have to be firstly for natbib and only after those can chapterbib be considered.

\bibsection \endthebibliography Natbib provides \bibsection for titling the bibliography. I also have to extend the end of the thebibliography environment to cater for \postbibhook.

```
\AtBeginDocument{%
  \@ifpackageloaded{natbib}{% natbib is loaded
```

As natbib is used, change \endthebibliography to the class definition.

```
    \addtodef{\endthebibliography}{}{\vskip-\lastskip\postbibhook}
    \@ifpackagewith{natbib}{sectionbib}{% with sectionbib option
```

natbib's sectionbib option is used,

```
      \renewcommand{\bibsection}{\@memb@bsec}}%
```

The sectionbib option is not used, so we have a chapter title.

```
      {\renewcommand{\bibsection}{\@memb@bchap}}}%
```

Finished with any natbib related changes.

```
  {}
```

Now deal with chapterbib if necessary.

```
  \@ifpackagewith{chapterbib}{sectionbib}{%
```

chapterbib is used with its sectionbib option. This is the only case to worry about. Kill chapterbib's \sectionbib macro which it calls at begin document to do its patch, then make sure the class definition is used.

```
    \renewcommand{\sectionbib}[2]{}
    \renewcommand{\bibsection}{\@memb@bsec}}{}
```

And we've finished with this bunch of \AtBeginDocument code.

```
}

```

24 The index

I allow for a single column index as well as the default double column.

\ifonecolindex TRUE for a one column index.

```
\newif\ifonecolindex
  \onecolindexfalse

```

\onecolindex
\twocolindex

```
\newcommand*{\onecolindex}{\onecolindextrue}
\newcommand*{\twocolindex}{\onecolindexfalse}

```

theindex The environment theindex can be used for indices. It makes an index with two columns, with each entry a separate paragraph. At the user level the commands \item, \subitem and \subsubitem are used to produce index entries of various levels. When a new letter of the alphabet is encountered an amount of \indexspace white space can be added.

```
\newenvironment{theindex}{%
  \clearforchapter
  \if@twocolumn
    \@restonecolfalse
  \else
    \@restonecoltrue
  \fi
  \ifonecolindex
    \onecolumn
    \chapter*{\indexname}
    \preindexhook
  \else
    \setlength{\columnseprule}{\indexrule}%
    \setlength{\columnsep}{\indexcolsep}%
    \twocolumn[\@makeschapterhead{\indexname}
                \preindexhook]%
  \fi
  \indexmark
  \ifnoindexintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\indexname}%
  \fi
  \thispagestyle{indextitlepagestyle}\parindent\z@
  \parskip\z@ \@plus .3\p@\relax
  \let\item\@idxitem}%
  {\if@restonecol\onecolumn\else\twocolumn\fi}

```

\ps@indextitlepagestyle

```
\aliaspagestyle{indextitlepagestyle}{chapter}

```

\ifnoindexintoc
\indexintoc
\noindexintoc

Flag to control whether or not to add the index title to the ToC, and declarations to set the flag. Default is to put the title into the ToC.

```
\newif\ifnoindexintoc
\newcommand*{\indexintoc}{\noindexintocfalse}
\newcommand*{\noindexintoc}{\noindexintoctrue}
\indexintoc
```

\indexcolsep
\indexrule

These two lengths control the column separation and the thickness of the inter-column rule.

```
\newlength{\indexcolsep} \setlength{\indexcolsep}{35pt}
\newlength{\indexrule}   \setlength{\indexrule}{0pt}
```

\preindexhook

A macro that is called between the index heading and the start of the two columns. The user can modify it to add something.

```
\newcommand{\preindexhook}{}
```

\l@index

Format the index entry in the table of contents.

```
\newcommand{\l@index}{\@dottedtocline{1}{0em}{0pt}}
```

\@idxitem
\subitem
\subsubitem

Thsee macros are used to format the entries in the index.

```
\newcommand{\@idxitem}  {\par\hangindent 40\p@}
\newcommand{\subitem}   {\par\hangindent 40\p@ \hspace*{20\p@}}
\newcommand{\subsubitem}{\par\hangindent 40\p@ \hspace*{30\p@}}
```

\indexspace

The amount of white space that is inserted between 'letter blocks' in the index.

```
\newcommand{\indexspace}{\par \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
```

\makeindex
\index
\specialindex
\makememindexhook

This is a modified version of the kernel \makeindex to allow for multiple indexes. It also defines \index (which is a modified version of the standard \index) and \specialindex.

\makememindexhook is a null op but can be redefined to add extra code into \makeindex. For example, to incorporate ledmac's \edindex into the scheme:

```
\renewcommand*{\makememindexhook}{%
   \def\edindex{\@bsphack%
     \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}}
```

```
\newcommand*{\makememindexhook}{}
\providecommand*{\makeindex}{}
\renewcommand*{\makeindex}[1][\jobname]{%
 \if@filesw
   \def\index{\@bsphack%
     \@ifnextchar [{\@index}{\@index[\jobname]}}
   \def\specialindex{\@bsphack\@spindex}%
   \makememindexhook
```

```
    \expandafter\newwrite\csname #1@idxfile\endcsname
    \expandafter\immediate\openout \csname #1@idxfile\endcsname #1.idx\relax
    \typeout{Writing index file #1.idx }%
  \fi}
```

Initially define, but emasculate, `\index` and `\specialindex` which are defined properly by the user calling `\makeindex`.

```
\renewcommand{\index}[2][\jobname]{\@bsphack\@esphack}
\newcommand{\specialindex}[3]{\@bsphack\@esphack}
```

`\printindex` The command to read an `ind` file.

```
\newcommand{\printindex}[1][\jobname]{\@input@{#1.ind}}
```

`\ifreportnoidxfile` `\reportnoidxfile` `\ignorenoidxfile` `\ifshowindexmark` `\showindexmarks` `\hideindexmarks` Two booleans to control reporting on unknown `idx` files and displaying indexed items in the margin.

```
\newif\ifreportnoidxfile
\newcommand*{\reportnoidxfile}{\reportnoidxfiletrue}
\newcommand*{\ignorenoidxfile}{\reportnoidxfilefalse}
  \ignorenoidxfile
\newif\ifshowindexmark
\newcommand*{\showindexmarks}{\showindexmarktrue}
\newcommand*{\hideindexmarks}{\showindexmarkfalse}
  \hideindexmarks
```

`\@index` `\@index[file]` is first stage of `\index`, handling the `idx` file.

```
\def\@index[#1]{%
  \@ifundefined{#1@idxfile}%
  {\ifreportnoidxfile
     \@memwarn{Undefined index file #1}%
    \fi
    \begingroup
    \@sanitize
    \@nowrindex}%
  {\def\@idxfile{#1}%
   \begingroup
   \@sanitize
   \@wrindexm@m}}
```

`\@nowrindex` Called when there is no `idx` file to throw away the indexed item.

```
\newcommand{\@nowrindex}[1]{%
  \ifshowindexmark\@showidx{#1}\fi\endgroup\@esphack}
```

`\@wrindexm@m` `\@@wrindexhyp` The next stage in index processing is `\@wrindexm@m{item}`, which writes the `idx` file name and indexed item to the `aux` file. The `\@@wrindexhyp` macro provides hyperlinks in case the hyperref package is used.

```
\newcommand{\@wrindexm@m}[1]{\@@wrindexhyp#1||\\}
\def\@@wrindexhyp#1|#2|#3\\{%
  \ifshowindexmark\@showidx{#1}\fi
  \ifx\\#2\\%
    \protected@write\@auxout{}%
      {\string\@@wrindexm@m{\@idxfile}{#1|hyperpage}{\thepage}}%
  \else
    \def\Hy@temp@A{#2}%
    \ifx\Hy@temp@A\HyInd@ParenLeft
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}{#1|#2hyperpage}{\thepage}}%
    \else
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}{#1|#2}{\thepage}}%
    \fi
  \fi
  \endgroup
  \@esphack}
```

`\hyperpage` `\hyperlink` These are defined in the hyperref package but we need them. Other hyperstuff is only used if the package itself is used.

```
\newcommand{\hyperpage}[1]{#1}
\newcommand{\hyperlink}[2]{#2}

```

`\@@wrindexm@m` The macro `\@@wrindexm@m{file}{item}{page}` has been written into the `aux` file. It is normally defined so that it calls `\@@@wrindexm@m{item}{page}` to finally write to the `idx` file.

```
\newcommand{\@@wrindexm@m}[1]{\begingroup
  \def\@idxfile{\@nameuse{#1@idxfile}}
  \@sanitize
  \@@@wrindexm@m}

```

`\@@@wrindexm@m` `\@@@wrindexm@m{`⟨*item*⟩`}{`⟨*page*⟩`}` writes the `\indexentry` to the `idx` file. This file is read both at the beginning and end of a document so `\@@@wrindexm@m` must be disabled for one of these reads otherwise the index entries will be doubled up. Initially I disabled it after the first read. It eventually dawned on me that this meant two LaTeX runs to update the `idx`. Enabling it for the second read means that the `idx` file is updated at the end of each run.

```
\newcommand{\@@@wrindexm@m}[2]{\endgroup}
\AtBeginDocument{%
  \def\@@@wrindexm@m#1#2{%
    \if@filesw
      \immediate\write \@idxfile{\string\indexentry{#1}{#2}}%
    \fi
    \endgroup}%
}
```

\@spindex Now do similar things for \specialindex.

```
\newcommand{\@spindex}[2]{%
  \@ifundefined{#1@idxfile}%
  {\ifreportnoidxfile
     \@memwarn{Undefined index file #1}%
    \fi
    \begingroup
    \@sanitize
    \@nowrindex}%
  {\def\@idxfile{#1}%
   \def\@sptheidx{#2}%
   \begingroup
   \@sanitize
   \@wrspindex}}

```

\@wrspindex
\@@wrspindexhyp

```
\newcommand{\@wrspindex}[1]{\@@wrspindexhyp#1||\\}
\def\@@wrspindexhyp#1|#2|#3\\{%
  \ifshowindexmark\@showidx{#1}\fi
  \ifx\\#2\\%
    \protected@write\@auxout{}%
      {\string\@@wrindexm@m{\@idxfile}%
        {#1|hyperspindexpage(\thepage)}%
      {\@nameuse{the\@sptheidx}}}%
  \else
    \def\Hy@temp@A{#2}%
    \ifx\Hy@temp@A\HyInd@ParenLeft
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}%
         {#1|#2hyperspindexpage(\thepage)}%
         {\@nameuse{the\@sptheidx}}}%
    \else
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}{#1|#2}%
        {\@nameuse{the\@sptheidx}}}%
    \fi
  \fi
  \endgroup
  \@esphack}

```

\hyperspindexpage

```
\def\hyperspindexpage(#1)#2{\hyperlink{page.#1}{#2}}

```

\ifmemhyperindex
\memhyperindextrue
\memhyperindexfalse

Flag to turn off \hyperindexfalse hyperrefing the index (apparently xindy can't cope with this. Email from Frederic Connes, 2005/07/13:

```
...You use |hyperspindex(\thepage) which xindy doesn't
recognise as a valid markup-locref. And I don't know how
```

```
to add it because xindy only accepts one argument in
markup-locref (if the number is not a page number, it will
still point to a page with that number) so replacing it
with |hyperpage won't work. ...
```

The default is \memhyperindextrue. Setting \memhyperindexfalse prohibits a hyper index — setting whatever option in hyperref to disable hyper indexing will (probably) not work as far as xindy is concerned.

```
\newif\ifmemhyperindex
  \memhyperindextrue

```

\ifm@mxindy
\m@mxindytrue
\m@mxindyfalse
\xindyindex
\@@wrspindexhyp

Use \xindyindex when you are going to use the xindy program rather than makeindex. hyperrefed entries won't work with xindy. The code was supplied by Frederic Connes[21] in an email to me on 2006/01/08.

```
\newif\ifm@mxindy
\m@mxindyfalse
\newcommand*{\xindyindex}{\m@mxindytrue}
\def\@@wrspindexhyp#1|#2|#3\\{%
  \ifshowindexmark\@showidx{#1}\fi
  \ifx\\#2\\%
    \protected@write\@auxout{}%
      {\string\@@wrindexm@m{\@idxfile}%
        \ifm@mxindy{#1}\else{#1|hyperspindexpage(\thepage)}\fi
      {\@nameuse{the\@sptheidx}}}%
  \else
    \def\Hy@temp@A{#2}%
    \ifx\Hy@temp@A\HyInd@ParenLeft
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}%
         \ifm@mxindy{#1|#2}\else{#1|#2hyperspindexpage(\thepage)}\fi
      {\@nameuse{the\@sptheidx}}}%
    \else
      \protected@write\@auxout{}%
        {\string\@@wrindexm@m{\@idxfile}{#1|#2}%
      {\@nameuse{the\@sptheidx}}}%
    \fi
  \fi
  \endgroup
  \@esphack}

```

\@@wrindexhyp
\@@wrspindexhyp

If the hyperref package is not being used, or \ifmemhyperindex is false, there is no need to clutter up the index files.

```
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}{}{\memhyperindexfalse}%
```

[21] frederic@connes.org

If the hyperref package is not being used, or a hyperindex is not required, simplify!

```
\ifmemhyperindex\else
  \def\@@wrindexhyp#1|\|\{%
    \ifshowindexmark\@showidx{#1}\fi
    \protected@write\@auxout{}%
      {\string\@@wrindexm@m{\@idxfile}{#1}{\thepage}}%
    \endgroup
    \@esphack}%
  \def\@@wrspindexhyp#1|\|\{%
    \ifshowindexmark\@showidx{#1}\fi
    \protected@write\@auxout{}%
      {\string\@@wrindexm@m{\@idxfile}{#1}{\@nameuse{the\@sptheidx}}}%
    \endgroup
    \@esphack}%
  \fi
}

```

\see These definitions are taken from the makeidx package.

\seename
\seealso
\alsoname

```
\newcommand*{\see}[2]{\emph{\seename} #1}
\newcommand*{\seename}{see}
\newcommand*{\seealso}[2]{\emph{\alsoname} #1}
\newcommand*{\alsoname}{see also}

```

\citeindexfile For the natbib package, and possibly other packages that do some special indexing.

```
\newcommand{\citeindexfile}{\jobname}
\AtBeginDocument{\@ifpackageloaded{natbib}{%
  \def\NAT@index{\index[\citeindexfile]{\NAT@idxtxt}}}{}}

```

The next part of the code is essentially the showidx package. I tried putting index entries into marginpars but too many on a page led to the 'too many floats' problem.

\indexmarkstyle
\@indexbox

```
\newtoks\indexmarkstyle
  \indexmarkstyle{\normalfont\footnotesize\ttfamily}
\newinsert\@indexbox
  \dimen\@indexbox\maxdimen

\begingroup
  \catcode`\@\active
  \expandafter\gdef\csname\string @sanitizeat\endcsname
  {\def @{\char`\@}}
\endgroup

```

\@showidx
```
\newcommand{\@showidx}[1]{%
  \insert\@indexbox{%
    \@sanitizeat
    \the\indexmarkstyle
    \hsize\marginparwidth
    \hangindent\marginparsep \parindent\z@
    \everypar{}\let\par\@@par \parfillskip\@flushglue
    \lineskip\normallineskip
    \baselineskip .8\normalbaselineskip\sloppy
    \raggedright \leavevmode
    \vrule \@height .7\normalbaselineskip \@width \z@\relax
      #1\relax
    \vrule \@height \z@ \@depth .3\normalbaselineskip \@width \z@\relax
  }%
  \ifhmode\penalty\@M \hskip\z@skip\fi}

```

\@leftidx
\@rightidx
```
\newcommand{\@leftidx}{\hskip-\marginparsep \hskip-\marginparwidth}
\newcommand{\@rightidx}{\hskip\columnwidth \hskip\marginparsep}

```

\@mkidx
```
\newcommand{\@mkidx}{\vbox to \z@{%
  \rlap{%
    \if@twocolumn
      \if@firstcolumn \@leftidx \else \@rightidx \fi
    \else
      \if@twoside
        \ifodd\c@page \@rightidx \else \@leftidx \fi
      \else
        \@rightidx
      \fi
    \fi
    \box\@indexbox
  }%
  \vss}}

```

25 Page bottom

\raggedbottom
\flushbottom

These kernel macros need changing because of the new \@indexbox marginal insert.

\@texttop
```
\renewcommand{\raggedbottom}{%
  \def\@textbottom{\vskip\z@ plus.0001fil}%
  \let\@texttop\@mkidx}
\renewcommand{\flushbottom}{%
```

```
  \let\@textbottom\relax
  \let\@texttop\@mkidx}
\let\@texttop\@mkidx
```

25.1 Widows and sloppybottom

There was a discussion *widow handling?* on CTT in May 2006. `ivowel@gmail.com` wrote

> in experimenting with raggedbottom, widowpenalty, and clubpenalty, I think that I have not found a solution that strikes me as particularly desirable. I think what I would really like is that widows (i.e., left-over single lines that begin on the following page) are resolved not by pushing one extra line from the same paragraph also onto the next page, but by stretching the `\textheight` to allow this one extra at the bottom of the same page.

Donald Arseneau, as he so often does, came up with a solution he termed `\sloppybottom`. Here is a generalised version.

`\sloppybottom` `\sloppybottom` allows an extra line on a page to save a widow. You must increase the `\topskip` (by 60 percent is reasonable) and this will push the text lower on the page. Run `\checkandfixthelayout` after the change. For example:

```
\setlength{\topskip}{1.6\topskip}
\checkandfixthelayout
\sloppybottom
...
```

```
\newcommand*{\sloppybottom}{%
  \def\@textbottom{\vskip \z@ \@plus.0001fil \@minus .95\topskip}%
  \topskip=1\topskip \@plus 0.625\topskip \@minus .95\topskip
  \def\@texttop{\vskip \z@ \@plus -0.625\topskip \@minus -0.95\topskip}}
```

26 Glossaries

Standard LaTeX provides little and insufficient support for glossaries, just a `\makeglossary` and `\glossary` commands. This class does somewhat better. The code follows along the lines of that for indexes.

`\ifonecolglossary` TRUE for a one column glossary, otherwise its two column.
`\onecolglossarytrue`
`\onecolglossaryfalse`

```
\newif\ifonecolglossary
  \onecolglossarytrue
```

\onecolglossary
\twocolglossary
```
\newcommand*{\onecolglossary}{\onecolglossarytrue}
\newcommand*{\twocolglossary}{\onecolglossaryfalse}
```

theglossary The environment theglossary is used for glossaries. It makes a glossary with one or two columns, headed by a chapter-like title.

```
\newenvironment{theglossary}{%
  \if@twocolumn
    \@restonecolfalse
  \else
    \@restonecoltrue
  \fi
  \ifonecolglossary
    \onecolumn
    \chapter*{\glossaryname}
    \preglossaryhook
  \else
    \setlength{\columnseprule}{\glossaryrule}
    \setlength{\columnsep}{\glossarycolsep}
    \twocolumn[\@makeschapterhead{\glossaryname}
                \preglossaryhook]%
  \fi
  \glossarymark
  \ifnoglossaryintoc\else
    \phantomsection
    \addcontentsline{toc}{chapter}{\glossaryname}
  \fi
  \thispagestyle{chapter}\parindent\z@
  \parskip\z@ \@plus .3\p@\relax
  \begintheglossaryhook}%
  {\atendtheglossaryhook\if@restonecol\onecolumn\else\twocolumn\fi}
```

\begintheglossaryhook
\atendtheglossaryhook
Two vacuous macros called as the last thing by \begin{theglossary} and the first thing by \end{theglossary} These could be used, for example, to insert another kind of environment.

```
\newcommand*{\begintheglossaryhook}{}
\newcommand*{\atendtheglossaryhook}{}
```

\preglossaryhook A vacuous macro called after the title is set and before the listing starts. The user can modify it to, for example, add some explanatory text.

```
\newcommand*{\preglossaryhook}{}
```

\ifnoglossaryintoc
\glossaryintoc
\noglossaryintoc
Flag and declarations to control whether or not the glossary title is added to the ToC. Default is to put the title in the ToC.

```
\newif\ifnoglossaryintoc
\newcommand*{\glossaryintoc}{\noglossaryintocfalse}
```

```
\newcommand*{\noglossaryintoc}{\noglossaryintoctrue}
  \glossaryintoc
```

`\glossarycolsep` `\glossaryrule` When the glossary is two column these lengths control the column separation and the width of a rule between the columns.

```
\newdimen\glossarycolsep \glossarycolsep=35\p@
\newdimen\glossaryrule \glossaryrule=0\p@
```

`\glossaryspace` Vertical space between 'letter blocks' in the glossary. Note that this is a macro, not a length.

```
\newcommand*{\glossaryspace}{%
    \par \vskip 1.0\onelineskip \@plus 5\p@ \@minus3\p@\relax}
```

`\makeglossary` The preamble command to set up a glossary. Output is to the `#1.glo` file. We need the `\providecommand` in case `\nofiles` has been called before the class. `\makeglossary` sets up several file-specific macros.

```
\providecommand*{\makeglossary}{}
\renewcommand*{\makeglossary}[1][\jobname]{%
  \makememglossaryhook
  \@namedef{memglsact#1}{@}%           actual
  \@namedef{memglsnx#1}{}%             no ref
  \@namedef{memglsn#1}{\thepage}%      num  by page
  \@namedef{memglsnf#1}{|memjustarg}%  no special number format
  \if@filesw \expandafter\newwrite\csname #1memglofile\endcsname
    \expandafter\immediate\openout \csname #1memglofile\endcsname #1.glo\relax
    \typeout{Writing glossary file #1.glo }%
  \fi}
```

`\makememglossaryhook` A vacuous macro called at the start of the `\makeglossary` code. It is redefinable to do something useful.

```
\newcommand*{\makememglossaryhook}{}
```

`\glossary` The user command for a raw glossary item. The full calling sequence looks like:
`\glossary[`⟨*file*⟩`](`⟨*key*⟩`){`⟨*term*⟩`}{`⟨*desc*⟩`}`
but most is accomplished by lower level macros. It calls `\@glossary` to handle the first optional argument.

```
  \def\glossary{\@bsphack%
    \@ifnextchar [{\@glossary}{\@glossary[\jobname]}}%
```

`\@glossary` `\@glossary[`⟨*file*⟩`]` handles the first of the `\glossary` optional arguments, even though the calling sequence looks like:
`\@glossary[`⟨*file*⟩`](`⟨*key*⟩`){`⟨*term*⟩`}{`⟨*desc*⟩`}`.
It calls `\@@glossary` to handle the second optional argument and the rest.

```
\def\@glossary[#1]{%
  \@ifnextchar ({\@@glossary[#1]}{\@@glossary[#1]()}}
```

\@@glossary The base macro for \glossary. Its real calling sequence is:
\@@glossary[⟨file⟩](⟨key⟩){⟨term⟩}{⟨desc⟩}.
Provided the glossary output file has been set up it calls \@wrglom@m to write the data, otherwise it throws away its arguments.

```
\def\@@glossary[#1](#2)#3#4{%
  \@ifundefined{#1memglofile}{%
    \begingroup
    \@sanitize
    \endgroup
    \@esphack%
   }{%
    \def\memglofile{#1}%
    \begingroup
    \@sanitize
```

Use the correct key, ⟨key⟩ if given, otherwise ⟨term⟩.

```
    \ifx\@empty#2\@empty
      \@wrglom@m{#3}{#3}{#4}%
    \else
      \@wrglom@m{#2}{#3}{#4}%
    \fi}}

```

\@wrglom@m \@wrglom@m{⟨key⟩}{⟨term⟩}{⟨desc⟩} writes its three arguments, plus the ⟨ref⟩ and the ⟨num⟩ to the aux file as the arguments to the \@@wrglom@m macro.

```
\newcommand{\@wrglom@m}[3]{%
  \protected@write\@auxout{}%
    {\string\@@wrglom@m{\memglofile}{#1}{#2}{#3}{\@nameuse{memglsnx\memglofile}}{\@nam
  \endgroup
  \@esphack}

```

\@@wrglom@m The calling sequence looks like:
\@@wrglom@m{⟨file⟩}{⟨key⟩}{⟨term⟩}{⟨desc⟩}{⟨ref⟩}{⟨num⟩}
but is actually really only
\@@wrglom@m{⟨file⟩}
It saves the output file identifier in \memglofile and ⟨file⟩ as \m@mgf, then calls \memwritetoglo to handle the remaining 'arguments'.

```
\newcommand{\@@wrglom@m}[1]{\begingroup
  \def\memglofile{\@nameuse{#1memglofile}}%
  \def\m@mgf{#1}%
  \@sanitize
  \memwritetoglo}

```

\memwritetoglo \memwritetoglo{⟨key⟩}{⟨name⟩}{⟨desc⟩}{⟨ref⟩}{⟨num⟩} will, at an
\@ctualm@mwritetoglo appropriate time write the raw glossary data to the appropriate glo file.
The aux file is read twice, once as one of the initial actions at the start of the document environment, and again at the end of the document environment. The

raw glossary data must only be written once from the `aux` to the `glo` file. Initially `\memwritetoglo` is defined to do nothing, then, after the `aux` file has been read at the beginning is redefined to do its writing.
The Makeindex program expects its input to be like:
`\glosaryentry{key@data}{num}` Very cunningly the raw data is fed to the `glo` file looking like:
`\glossaryentry{key@{\memgloterm{term} \memglodesc{desc} \memgloref{ref}}{num}`
which as far as Makeindex is concerned is simply:
`\glosaryentry{key@{data}}{num}`
(note where all the braces are).

```
\newcommand{\memwritetoglo}[5]{\endgroup}
\newcommand{\@ctualm@mwritetoglo}[5]{%
  \immediate\write \memglofile{\string\glossaryentry{#1\@nameuse{memglsact\m@mgf}
                    {\string\memgloterm{#2}}{\string\memglodesc{#3}}
                    {\string\memgloref{#4}}\@nameuse{memglsnf\m@mgf}}{#5}}%
  \endgroup}
\AtBeginDocument{%
  \let\memwritetoglo\@ctualm@mwritetoglo}
```

`\changeglossactual` `\changeglossref` `\changeglossnum` `\changeglossnumformat`

There are internal macros that are file-dependant, their names being constructed from the file name plus a fixed name. The `\makeglossary` command initializes these but the user may well need to change them. In the macros below, `...actual` refers to the actual character in a Makeindex configuration file (default `@`), `...ref` to the ⟨*ref*⟩ glossary argument, `...num` to the metanum argument (default `\thepage`) and `...numformat` to the Makeindex encapsulating command. Example uses are:

```
\changeglossactual[file]{?}
\changeglossref[file]{\thepage}
\changeglossnum[file]{\thesection}
\changeglossnumref[file]{|textbf}
```

```
\newcommand*{\changeglossactual}[2][\jobname]{%
  \@namedef{memglsact#1}{#2}}
\newcommand*{\changeglossref}[2][\jobname]{%
  \@namedef{memglsnx#1}{#2}}
\newcommand*{\changeglossnum}[2][\jobname]{%
  \@namedef{memglsn#1}{#2}}
\newcommand*{\changeglossnumformat}[2][\jobname]{%
  \@namedef{memglsnf#1}{#2}}
```

`\glossitem` The final sorted glossary data is in the form:
`\glossitem{`⟨*term*⟩`}{`⟨*desc*⟩`}{`⟨*ref*⟩`}{`⟨*num*⟩`}`
The definition here does little.

```
\newcommand{\glossitem}[4]{#1 #2 #3 #4\par}
```

\memgloterm
\memglodesc
\memgloref
\memglonum
These are wrappers round the ⟨*term*⟩, ⟨*desc*⟩, ⟨*ref*⟩ and ⟨*num*⟩ elements in the final data. They are vacuous to start with.

```
\newcommand*{\memgloterm}[1]{#1}
\newcommand*{\memglodesc}[1]{#1}
\newcommand*{\memgloref}[1]{#1}
\newcommand*{\memglonum}[1]{#1}
```

\printglossary
\printglossary[⟨*file*⟩] inputs the glossary gst file.

```
\newcommand*{\printglossary}[1][\jobname]{\@input@{#1.gls}}
```

27 Notes/Marginalia

27.1 A simple interface for specifying locations

The memoir class as of December 2009 supports \marginpar, \sidepar, and \sidebar for placing information in a margin. The means of specifying in which margin the information should be put is different for each of these. Here a consistent location interface is provided for all marginalia.

\m@msetm@argin
\m@mm@argin
A workhorse macro for setting a margin placement code. The argument may be one of: left, right, outer, or inner, with \m@mm@argin defined as the corresponding codes of 0, 1, 2, or 3. An unkown argument is coded as a negative number.

```
\newcommand*{\m@msetm@argin}[1]{%
  \def\@tempa{#1}\def\@tempb{left}%
  \ifx\@tempa\@tempb
    \def\m@mm@argin{0}%
  \else
    \def\@tempb{right}%
    \ifx\@tempa\@tempb
      \def\m@mm@argin{1}%
    \else
      \def\@tempb{outer}%
      \ifx\@tempa\@tempb
        \def\m@mm@argin{2}%
      \else
        \def\@tempb{inner}%
        \ifx\@tempa\@tempb
          \def\m@mm@argin{3}%
        \else
          \def\m@mm@argin{-1}%
        \fi
      \fi
    \fi
  \fi}
```

\ifmemtortm
\memtortmtrue
\memtortmfalse

A flag set TRUE if the marginalia is to be moved to the right of the text and FALSE if it is to be moved to the left of the text.

```
% MEM-TO-RighT-Margin
\newif\ifmemtortm
  \memtortmtrue
```

\m@mwhich@margin

\m@mwhich@margin{⟨*code*⟩} sets \ifmemtortm TRUE or FALSE depending on the value of ⟨*code*⟩ (0 to 3), and other factors.

Two column document If the first column, to the left, otherwise to the right, irrespective the document being one- or two-side.

One sided document If code = 0 (left), to the left, otherwise (code = 1 (right), 2 (outer), 3 (inner)) to the right.

Two sided document depends on whether a recto or verso page:

Recto (odd) page right if code = 1 (right) or 2 (outer), otherwise left.

Verso (even) page left if code = 0 (left) or 2 (outer), otherwise right.

```
\newcommand*{\m@mwhich@margin}[1]{%
  \memtortmtrue
  \if@twocolumn
    \if@firstcolumn% left
      \memtortmfalse
    \else%           right
      \memtortmtrue
    \fi
  \else
    \if@twoside
      \checkoddpage
      \ifcase #1\relax% 0 left
        \memtortmfalse
      \or%                    1 right
        \memtortmtrue
      \or%                    2 outer
        \ifoddpage
          \memtortmtrue
        \else
          \memtortmfalse
        \fi
      \or%                 3 inner
        \ifoddpage
          \memtortmfalse
        \else
          \memtortmtrue
        \fi
      \fi% end ifcase
    \else%           oneside
      \ifnum #1=\z@% 0   left, all else right
```

```
        \memtortmfalse
      \else
        \memtortmtrue
      \fi
    \fi% end if@twoside
  \fi}
```

27.2 Marginpars

A `\marginpar` is a kind of floating object — you can't control exactly where it will go. There is one problem with the kernel definition of `\marginpar` in that sometimes a `\marginpar` may end up on the wrong side of the page. The following is an attempt to fix that using the odd/even page check provided as part of the class.

`\@addmarginpar` The part of the code for `\marginpar` that deals with deciding which side of the page it should be printed on is `\@addmarginpar`, buried away in the kernel's `output` routine. A couple of minor changes are made to the kernel code. The first is at the beginning where I have added the `\checkoddpage` page checking code. In the new margin placement syntax, there is a new syntax: always left, in twoside mode. This cannot be implemented using the normal marginpar placement controls, so we have to add our own.

```
\def\@addmarginpar{%
    \checkoddpage
```

Continue with the kernel code. The twocolumn stuff in the new syntax scheme is the same as in the kernel.

```
    \@next\@marbox\@currlist{\@cons\@freelist\@marbox
    \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
    \if@twocolumn
        \if@firstcolumn \@tempcnta\m@ne \fi
    \else
```

Here we add the new stuff. We make use of the fact that we have hidden all the page checking in `\m@mwhich@margin`, and we only fire it of if we know `\marginparmargin` have been used (via `\ifm@msetmp`).

```
      \ifm@msetmp%
        % \@tempcnta > 0 => right side of page
        % \@tempcnta < 0 => left side of page
        \m@mwhich@margin{\m@mmpar@margin}% set left or right margin
        \ifmemtortm%
          \@tempcnta\@ne\relax%
         \else%
          \@tempcnta\m@ne\relax%
        \fi%
      \else%
        \if@mparswitch
```

The next line, reading
`\ifodd\c@page \else\@tempcnta\m@ne \fi`
is where the odd/even page checking is done in the kernel code. I replace it with my code, and then continue with the kernel.

```
          \ifoddpage \else \@tempcnta\m@ne \fi%
        \fi%
        \if@reversemargin \@tempcnta -\@tempcnta \fi%
      \fi%
    \fi%
    \ifnum\@tempcnta <\z@  \global\setbox\@marbox\box\@currbox \fi
    \@tempdima\@mparbottom
    \advance\@tempdima -\@pageht
    \advance\@tempdima\ht\@marbox
    \ifdim\@tempdima >\z@
```

The next line in the kernel reads:
`\@latex@warning@no@line {Marginpar on page \thepage\space moved}`
I have changed the warning message to give the user an indication of the severity of the move. Then follow the kernel on to the end.

```
      \@latex@warning@no@line {Marginpar on page
                               \thepage\space moved by \the\@tempdima}%
    \else
      \@tempdima\z@
    \fi
    \global\@mparbottom\@pageht
    \global\advance\@mparbottom\@tempdima
    \global\advance\@mparbottom\dp\@marbox
    \global\advance\@mparbottom\marginparpush
    \advance\@tempdima -\ht\@marbox
    \global\setbox \@marbox
                   \vbox {\vskip \@tempdima
                          \box \@marbox}%
    \global \ht\@marbox \z@
    \global \dp\@marbox \z@
    \kern -\@pagedp
    \nointerlineskip
    \hb@xt@\columnwidth
      {\ifnum \@tempcnta >\z@
          \hskip\columnwidth \hskip\marginparsep
       \else
          \hskip -\marginparsep \hskip -\marginparwidth
       \fi
       \box\@marbox \hss}%
    \nointerlineskip
    \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}%
}
```

27.2.1 Marginpar – margin interface

The 'standard' method of specifying the desired margin for a \marginpar consists of the two macros \normalmarginpar and \reversemarginpar, whose effects depend on whether the document is one- or two-sided, whether the page is recto or verso, and if it is set in one- or two-columns.

\ifm@msetmp TRUE if the macro \marginparmargin has been called.

m@msetmptrue
m@msetmpfalse
```
\newif\ifm@msetmp
  \m@msetmpfalse
```

\marginparmargin
\m@mmpar@margin The user command for specifying the desired margin for \marginpars. The code is stored as \m@mmpar@margin, and \setmpbools is called to convert to the regular \marginpar booleans.

```
\newcommand*{\marginparmargin}[1]{%
  \m@msetmptrue
  \m@msetm@argin{#1}%
  \ifnum\m@mm@argin<\z@
    \@memwarn{Bad \string\marginparmargin\space argument `#1'\MessageBreak
              set to `outer'}%
    \gdef\m@mmpar@margin{2}%   set as outer
  \else
    \global\let\m@mmpar@margin\m@mm@argin
  \fi
  \setmpbools}

```

\setmpbools Given \m@mmpar@margin, \setmpbools sets the corresponding \marginpar location booleans.

```
\newcommand*{\setmpbools}{%
  \if@twoside
    \@mparswitchtrue
  \else
    \@mparswitchfalse
  \fi
  \ifcase\m@mmpar@margin\relax% 0 left
    \@reversemargintrue%  \sideparswitchfalse \reversesidepartrue
  \or%                    1 right
    \@reversemarginfalse% \sideparswitchfalse \reversesideparfalse
  \or%                    2 outer
    \@reversemarginfalse% \sideparswitchtrue \reversesideparfalse
  \or%                    3 inner
    \@reversemargintrue% \sideparswitchtrue \reversesidepartrue
  \fi}

\newcommand*{\m@msetmpcodes}{%
  \if@mparswitch%  2 sided
    \if@reversemargin% inner
```

```
      \def\m@mmpar@margin{3}%
    \else% outer
      \def\m@mmpar@margin{2}%
    \fi
  \else% 1 sided
    \if@reversemargin% left
      \def\m@mmpar@margin{0}%
    \else% right
      \def\m@mmpar@margin{1}%
    \fi
  \fi}
```

27.3 A fixed marginpar

Introduce a non-floating marginpar.

`\parnopar` From *The TEXbook* Exercise 14.15. It creates an 'invisible' end/start paragraph, and may be used for getting TEX to try a pagebreak.
Roman Eisle (email 2008/09/05) found a problem with the original definition as its effects continued on:
`\newcommand{\parnopar}{\parfillskip=0pt\par\parskip=0pt\noindent}`
and he supplied the fix used below.

```
\newcommand{\parnopar}{\parfillskip=0pt\par\parskip=0pt\noindent
                       \parfillskip\@flushglue}
```

`\ifreversesidepar` `\ifsideparswitch` Analogues of `\marginpar` controls.[22]

```
\newif\ifreversesidepar
%  \reversesideparfalse
  \reversesidepartrue
\newif\ifsideparswitch
  \sideparswitchfalse
\if@twoside \sideparswitchtrue \fi
```

`\ifm@msetsp` `\m@msetsptrue` `\m@msetspfalse`

```
%% true if \sideparmargin used
\newif\ifm@msetsp
  \m@msetspfalse
```

`\sideparmargin` `\m@mspar@margin` User command for specifing the margin for `\sidepar`s. The numeric code is stored in `\m@mspar@margin`.

```
\newcommand*{\sideparmargin}[1]{%
  \m@msetsptrue
```

[22] Note that for some now forgotten reason the default placement of the `\sidepar` is reversed in comparison to say `\marginpar`. As not to change existing documents, we have left the default like this.

```
  \m@msetm@argin{#1}%
  \ifnum\m@mm@argin<\z@
    \@memwarn{Bad \string\sideparmargin\space argument `#1'\MessageBreak
              set to `outer'}%
    \gdef\m@mspar@margin{2}%   set as outer
  \else
    \global\let\m@mspar@margin\m@mm@argin
  \fi}

```

`\m@msidepar@left` `\m@msidepar@right` Move a `\sidepar` into the left/right margin.

```
\newcommand*{\m@msidepar@left}{%
  \@tempdimc\marginparwidth
  \advance\@tempdimc\marginparsep
  \kern-\@tempdimc}
\newcommand*{\m@msidepar@right}{%
  \@tempdimc\columnwidth
  \advance\@tempdimc\marginparsep
  \kern\@tempdimc}

```

`\setspbools` Given the `\m@mspar@margin` code, calculates the corresponding `\sidepar` location booleans.

```
\newcommand*{\setspbools}{%
    \ifcase\m@mspar@margin\relax% 0 left
      \sideparswitchfalse \reversesidepartrue
    \or%                    1 right
      \sideparswitchfalse \reversesideparfalse
    \or%                    2 outer
      \sideparswitchtrue \reversesideparfalse
    \or%                    3 inner
      \sideparswitchtrue \reversesidepartrue
    \fi}

```

`\setspcode` Given the `\sidepar` location booleans, sets up the corresponding `\m@mspar@margin` numeric codes.

```
\newcommand*{\setspcode}{%
  \ifsideparswitch
    \ifreversesidepar
      \def\m@mspar@margin{3}% inner
    \else
      \def\m@mspar@margin{2}% outer
    \fi
  \else
    \ifreversesidepar
      \def\m@mspar@margin{0}% left
    \else
      \def\m@mspar@margin{1}% right
```

```
    \fi
  \fi}
```

\sideparfont
\sideparform

The font to be used for \sidepars and the alignment for it. The \sideparform holds code that will set the text flush against the edge of the text.[23]

```
\newcommand*{\sideparfont}{\normalfont\normalsize}
\newcommand*{\sideparform}{\ifmemtortm\raggedright\else\raggedleft\fi}
```

\sidepar
\@sidepar
\sideparvshift

\sidepar[⟨*left*⟩]{⟨*right*⟩} (per \marginpar). Sidepars had a nasty habit of moving up or down depending on whether characters in the sidepar and the main text line have ascenders and/or descenders. The length \sideparvshift was provided to enable adjustments. This is Dan Luecking's modified version of my original code (CTT *Re: sidepars drift up a point*, 2006/04/11) which does a much better job, and the default for \sideparvshift is now 0pt.

```
\newcommand{\sidepar}{\@dblarg{\@sidepar}}
\long\def\@sidepar[#1]#2{\leavevmode\@bsphack\strut\vadjust{%
  \checkoddpage
  \ifm@msetsp% \sideparmargin used
%%%%    \setspbools
  \else% \sideparmargin not used, set the \m@mspar@margin code
    \setspcode
  \fi
  \rlap{\kern-\parindent
    \m@mwhich@margin{\m@mspar@margin}% set left or right margin
    \ifmemtortm
      \m@msidepar@right
    \else
      \m@msidepar@left
    \fi
  \setbox0=\vtop to 0pt{%
    \begin{minipage}[t]{\marginparwidth}%
      \sideparform\sideparfont%
      \ifmemtortm #2\else #1\fi
    \end{minipage}%
    \vss}%
  \vtop to 0pt{\kern\sideparvshift% default should be 0pt
    \kern-\dp\strutbox
    \kern-\ht0
    \box0 \vss}}%
  \@esphack}
```

Vertical shift for sidepar to align with text line

```
\newlength{\sideparvshift}
\setlength{\sideparvshift}{0pt}
%%%%  \setlength{\sideparvshift}{-2.08ex}% seems to work for all font sizes
```

[23] One might consider redefining this using the \RaggedRight and \RaggedLeft macros from the ragged2e package.

27.4 Sidebars

On 2002/10/22 Donald Arseneau posted the following code to CTT for adding sidebars to plain TeX.

```
\newinsert\sideins
\skip\sideins=0pt
\count\sideins=0
\dimen\sideins=2in

\def\sidebarvsep{25pt}
\def\sidebarhsep{15pt}

\let\mainpagecontents\pagecontents

\def\sidecontents{%
  \ifvoid\sideins\else
      {\advance\hsize\sidebarhsep
       \moveright\hsize \vtop to0pt{%
       \vskip-\sidebarvsep \vskip\topskip % offset by difference
       \unvbox\sideins \vss}%
    }\fi}
\def\pagecontents{\sidecontents\mainpagecontents}
\long\def\sidebar#1{
  \insert\sideins{%
    \splittopskip\sidebarvsep\relax
    \hsize 1.5in \rightskip=0pt plus 20pt \it
    \noindent \vbox to \sidebarvsep{}\ignorespaces #1%
    \ifhmode \unskip\strut\fi \par
   }%
}
```

The following started off (2002/03/19) as my attempt to rewrite the above into a form suitable for LaTeX, and this class in particular.
Donald Arseneau came up with some improvements to my sidebar code. The impetus for this came from a CTT thread, *whitespace after my command*, 2006/11/30 and earlier. In email to me on 2006/11/30 he said:

> For memoir `\sidebar`, it seems the rules are expanding to fill the space for the sidebar, so here is my suggested change to `\sidecontents`.
>
> It occurs to me that a separate `\sidetsep` (or `\sidetopsep`) would be more valuable than having an explicit 1`\onelineskip`.
>
> I grouped common code into two macros, and altered `\sidebarform` too.

Since then we have been going back and forth, with the result that practically all the complex code is Donald's.

\sideins Create a new insert called \sideins, which also creates a corresponding skip, count, dimen, and box.
\skip\sideins is the extra vertical space to allow on the page for the insert,
\count\sideins is the magnification factor for page breaking,
\dimen\sideins is the maximum insertion size (height) per page,
and the inserted material will be in \box\sideins when a page is output.

```
\newinsert\sideins
  \skip\sideins=0pt
  \count\sideins=0

```

\sidebartopsep Separation at the top of a sidebar.

```
\newlength{\sidebartopsep}
  \setlength{\sidebartopsep}{0pt}
```

\setsidebarheight The macro \setsidebarheight{⟨*length*⟩} sets the total height of sidebars on a page to ⟨*length*⟩. This is an interface for controlling \dimen\sideins properly, taking account of \topskip and the sidebar font size (and therefore should be invoked *after* declaring \sidebarfont).
Note that \sidebartopsep is *not* included as part of the allocated height.

```
\newcommand{\setsidebarheight}[1]{%
  \setlength{\dimen\sideins}{#1}%
  \advance\dimen\sideins-\topskip
  \advance\dimen\sideins\ht\strutbox}

```

\sidebarhsep The length \sidebarhsep is the gap between the typeblock and the sidebar.
\sidebarvsep \sidebarvsep is the vertical gap between sidebars on a page. The maximum
\sidebarwidth number of sidebar lines on a page is `1 + (\dimen\sideins - \sidebarvsep)`, assuming \dimen\sideins is defined in terms of \onelineskip. The width of the sidebar is \sidebarwidth.

```
\newlength{\sidebarhsep}
\newlength{\sidebarvsep}
\newlength{\sidebarwidth}
```

\sidebarfont The font for typesetting the contents of a sidebar.

```
\newcommand{\sidebarfont}{\normalfont\normalsize}
```

\setsidebars \setsidebars{⟨*hsep*⟩}{⟨*width*⟩}{⟨*vsep*⟩}{⟨*topsep*⟩}{⟨*font*⟩}{⟨*height*⟩} sets the main \sidebar parameters. A * argument means leave the setting as is.

```
\newcommand*{\setsidebars}[6]{%
  \nametest{#1}{*}\ifsamename\else
    \setlength{\sidebarhsep}{#1}\@memznegtest{\sidebarhsep}%
  \fi
  \nametest{#2}{*}\ifsamename\else
    \setlength{\sidebarwidth}{#2}\@memznegtest{\sidebarwidth}%
  \fi
  \nametest{#3}{*}\ifsamename\else
```

```
    \setlength{\sidebarvsep}{#3}\@memnegtest{\sidebarvsep}%
  \fi
  \nametest{#4}{*}\ifsamename\else
    \setlength{\sidebartopsep}{#4}%
  \fi
  \nametest{#5}{*}\ifsamename\else
    \def\sidebarfont{#5}%
  \fi
  \nametest{#6}{*}\ifsamename\else
    \setsidebarheight{#6}%
    \ifdim\dimen\sideins>\z@\else
%%%%      \@memerror{\protect\sidebarheight\space is zero or negative}{\@ehd}%
      \@memwarn{\protect\sidebarheight\space is zero or negative}%
    \fi
  \fi}
\setsidebars{\marginparsep}%          sidebarhsep
            {\marginparwidth}%        sidebarwidth
            {\onelineskip}%           sidebarvsep
            {0pt}%                    sidebartopsep
            {\normalsize\normalfont}% sidebarfont
            {\textheight}%            sidebarheight

```

\sidebarform

\sidebarform sets the 'raggedness' in a sidebar. The current definition is less ragged than \raggedright. Examples:

```
for flushleftright \renewcommand{\sidebarform}{}
for raggedleft     \renewcommand{\sidebarform}{\raggedleft}
```

```
\newcommand{\sidebarform}{%\rightskip=\z@ \@plus 2em}
  %\memRTLrightskip=\z@ \@plus 2em
  \ifmemtortm\raggedright\else\raggedleft\fi%
  % \ifmemtortm\memRTLrightskip=\z@ \@plus 2em\else\memRTLleftskip=\z@ \@plus 2em\fi
}

```

\ifsidebaroneside

Flag for oneside.

```
\newif\ifsidebaroneside
  \if@twoside\sidebaronesidefalse\else\sidebaronesidetrue\fi

```

\sidebarmargin
\m@msidebar@margin

\sidebarmargin{⟨*side*⟩} sets the side where sidebars will be printed. Valid arguments are: left, right, outer, or inner.

```
\newcommand*{\sidebarmargin}[1]{%
  \m@msetm@argin{#1}%
  \ifnum\m@mm@argin<\z@
    \@memwarn{Bad \string\sidebarmargin\space argument `#1'\MessageBreak
              set to `outer'}%
    \gdef\m@msidebar@margin{2}%   set as outer
```

```
  \else
    \global\let\m@msidebar@margin\m@mm@argin
  \fi}
%%%% default outer
\gdef\m@msidebar@margin{2}
```

`\m@sideb@left` `\m@sideb@right` Code used later to put sidebars into the desired margin.

```
\newcommand*{\m@sideb@left}{%
  \@tempdimc \sidebarwidth
  \advance\@tempdimc\sidebarhsep
  \kern-\@tempdimc}
\newcommand*{\m@sideb@right}{%
  \@tempdimc \columnwidth%  or \hsize
  \advance\@tempdimc\sidebarhsep
  \kern\@tempdimc}
```

`\sidecontents` This locates the horizontal position of a sidebar. On two column pages it is put in the margin next to the column where `\sidebar` is called, otherwise it is put in the outer (foredge) margin, except for the oneside option when it is put into the right hand margin. This can be separately controlled via `\ifsidebaroneside`. Emanuele Vicentini[24] suggested to code for the oneside option.

```
\newcommand{\sidecontents}{\hbox to \z@{%
  \m@mwhich@margin{\m@msidebar@margin}%
  \ifmemtortm
    \m@sideb@right
  \else
    \m@sideb@left
  \fi
```

DA and I have gone to and fro on the next bit of code trying to get the top alignment correct.

```
 \vtop to0pt{%
    \vskip\onelineskip
    \unvbox\sideins \vss}%
```

DA has the last word.

```
  \vtop to0pt{%
    \normalsize\normalfont\sidebarfont% select font so we know the strut size
    \vskip\topskip \vskip-\ht\strutbox
    \vskip\sidebartopsep% extra vertical shift
    \unvbox\sideins \vss}%
  \hss}}
```

`\sidebar` `\sidebar{`⟨*text*⟩`}` puts ⟨*text*⟩ into a sidebar.

[24] `emanuelevicentini@yahoo.it`, private email, 2004/04/05

Florence Henry (florence.henry@obspm.fr) in the thread *Of memoir, sidebar, and justification*, 2003/04/02, pointed out that itemize in a \sidebar could overflow. DA gave the solution hint.

DA says that having stacked insertions position properly is difficult. Here he uses the size of the strut to regulate both the height and the depth of each insertion (much as for footnotes). The following is his code and commentary.

```
\newcommand{\sidebar}[1]{%
  \insert\sideins{%
```

Begin the insertion with settings. The height of the strut box (dependant on the font) will determine the top alignment both initially and after a page break. The setting for maximum depth does not control anything; see the code further on instead.

```
    \hsize\sidebarwidth
    \@parboxrestore
```

Even though the margin placement is handled inside \sidecontents, there are some timming issues such that \ifmemtortm is executed too late for \sidebars internals to make use of it. Fixed by adding the margin check to \sidebar as well.

```
    \m@mwhich@margin{\m@msidebar@margin}%
    \sidebarform\sidebarfont
    \splittopskip=\ht\strutbox
    \splitmaxdepth=\dp\strutbox % doesn't do anything useful
```

Allow a free split at the top (so this whole insertion moves to the next page if it does not fit).

```
    \allowbreak
```

Control the (vertical) positioning of non-split entries. Footnotes (and previous versions of \sidebar) us a strut at the beginning of the text, but we will allow a baseline-skip to perform the same function. This method also introduces a \parskip when the paragraph begins, so we counteract that. (The problem with an initial strut is that it messes' up entries that begin with vertical space.)

```
    \prevdepth=\dp\strutbox    % supersedes a "top-strut"
    \vskip-\parskip
```

Now the user's sidebar entry:

```
    #1%
```

If the entry ended still in a paragraph, take the chance to insert a final strut; then end the paragraph.

```
    \ifvmode\else
      \unskip\@finalstrut\strutbox
    \fi\par
```

Explicitly force the effect of \maxdepth (\splitmaxdepth), but using the depth of the strut in the rôle of \maxdepth.

```
    \ifdim\prevdepth>\dp\strutbox \prevdepth=\dp\strutbox \fi
```

Now control how adjacent entries abut (whether or not a final strut got inserted) and how an entry fits at the bottom of the page. Use `\vskip`s to move from the text bottom to where a strut would bottom out, but insert an allowed breakpoint at the baseline position (so that the entry baseline may match the page's baseline). Finally insert the extra separation between entries.

```
    \ifdim\prevdepth>99\p@
      \nobreak
      \vskip-\prevdepth
      \allowbreak
      \vskip\dp\strutbox
    \fi
    \vskip\sidebarvsep}}
```

27.5 Footnotes

For my reference, in case I want to fiddle with it even more than I have already done so, here is the kernel code (from `ltfloat.dtx`) for footnotes. The insert for footnotes.

`\footins`

```
\newinsert\footins
  \skip\footins=\bigskipamount
  \count\footins=1000
  \dimen\footins=8in
```

Draws the rule separating footnotes from the main text. It is executed after a `\vspace` of `\skip\footins` and should take no vertical space.

`\footnoterule`

```
\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}

\@definecounter{footnote}
  \def\thefootnote{\@arabic\c@footnote}
\@definecounter{mpfootnote}
  \def\thempfootnote{\itshape\@alph\c@mpfootnote}
```

Generates the footnote marker from `\@thefnmark`.

`\@makefnmark`

```
\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}

\DeclareRobustCommand*\textsuperscript[1]{%
   \@textsuperscript{\selectfont#1}}
\def\@textsuperscript#1{%
   {\m@th\ensuremath{^{\mbox{\fontsize\sf@size\z@#1}}}}}

\newdimen\footnotesep
```

`\footnote` `\footnote[`⟨*num*⟩`]{`⟨*text*⟩`}` is what the user uses for footnoting. It defines

`\@thefnmark` and calls `\@footnotemark` and `\@footnotetext` to do the work.

```
\def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
  \protected@xdef\@thefnmark{\thempfn}%
  \@footnotemark\@footnotetext}}
```

`\@xfootnote` Handles the optional ⟨*num*⟩ argument to `\footnote`. It defines `\@thefnmark` and calls `\@footnotemark` and `\@footnotetext` to do the work.

```
\def\@xfootnote[#1]{%
  \begingroup
    \csname c@\@mpfn\endcsname #1\relax
    \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotemark\@footnotetext}
```

`\@footnotetext` `\@footnotetext{`⟨*text*⟩`}` sets up for typeseting the footnote (at the bottom of the page). It calls `\@makefntext{`⟨*text*⟩`}` to actually do the typesetting (`\@makefntext` has to be supplied by a class or package).

```
\long\def\@footnotetext#1{%
  \insert\footins{%
    \reset@font\footnotesize
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@footnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
    \color@endgroup}}
```

`\footnotemark` `\footnotemark[`⟨*num*⟩`]` the user uses this to produce just a footnote mark in the text.

```
\def\footnotemark{%
   \@ifnextchar[\@xfootnotemark
     {\stepcounter{footnote}%
      \protected@xdef\@thefnmark{\thefootnote}%
      \@footnotemark}}
```

`\@xfootnotemark` Handles the optional ⟨*num*⟩ argument to `\footnotemark`.

```
\def\@xfootnotemark[#1]{%
   \begingroup
     \c@footnote #1\relax
```

```
    \unrestored@protected@xdef\@thefnmark{\thefootnote}%
  \endgroup
  \@footnotemark}
```

`\@footnotemark` Typesets the mark in the main text, via `\@makefnmark`.

```
\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
  \@makefnmark
  \ifhmode\spacefactor\@x@sf\fi
  \relax}
```

`\footnotetext` `\footnotetext[`⟨*num*⟩`]{`⟨*text*⟩`}` is user view for creating a footnote without a marker in the main text.

```
\def\footnotetext{%
  \@ifnextchar [\@xfootnotenext
    {\protected@xdef\@thefnmark{\thempfn}%
    \@footnotetext}}
```

`\@xfootnotetext` Handles the optional ⟨*num*⟩ argument to `\footnotetext`.

```
\def\@xfootnotenext[#1]{%
  \begingroup
    \csname c@\@mpfn\endcsname #1\relax
    \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotetext}
```

`\@mpfn`
`\thempfn`

```
\def\@mpfn{footnote}
\def\thempfn{\thefootnote}
```

And from `ltboxes.dtx` for the `minipage` environment.

`\@mpfn` When setting up the environment, it includes:
`\thempfn`
`\@footnotetext`

```
\def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
\let\@footnotetext\@mpfootnotetext
```

`\@mpfootnotetext` Later it defines `\@mpfootnotetext{`⟨*text*⟩`}` as the minipage version of `\@footnotetext`.

```
\long\def\@mpfootnotetext#1{%
  \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins
```

```
\reset@font\footnotesize
\hsize\columnwidth
\@parboxrestore
\protected@edef\@currentlabel{%
  \csname p@mpfootnote\endcsname\@thefnmark}%
\color@begingroup
  \@makefntext{%
    \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
\color@endgroup}}
```

That completes the kernel extracts.

`\footnoterule` Usually, footnotes are separated from the main body of the text by a small rule. This rule is drawn by the macro `\footnoterule`. We have to make sure that the rule takes no vertical space (see `plain.tex`) so we compensate for the natural heigth of the rule of 0.4pt by adding the right amount of vertical skip.
To prevent the rule from colliding with the footnote we first add a little negative vertical skip, then we put the rule and make sure we end up at the same point where we begun this operation.

```
\renewcommand{\footnoterule}{%
  \kern-3\p@
  \hrule width .4\columnwidth
  \kern 2.6\p@}
```

And just to make sure that we are using memoir's font-dependent skips:

```
\skip\footins=\bigskipamount
```

`\c@footnote` Footnotes are numbered per chapter. The counter is predefined.

```
\@addtoreset{footnote}{chapter}
```

The footnote mechanism of LaTeX calls the macro `\@makefntext` to produce the actual footnote. The macro gets the text of the footnote as its argument and should use `\@thefnmark` as the mark of the footnote. The macro `\@makefntext` is called when effectively inside a `\parbox` of width `\columnwidth` (i.e., with `\hsize` = `\columnwidth`).
An example of what can be achieved is given by the following piece of TeX code.

```
\long\def\@makefntext#1{%
   \@setpar{\@@par
            \@tempdima = \hsize
            \advance\@tempdima-10pt
            \parshape \@ne 10pt \@tempdima}%
   \par
   \parindent 1em\noindent
   \hbox to \z@{\hss\@makefnmark}#1}
```

The effect of this definition is that all lines of the footnote are indented by 10pt, while the first line of a new paragraph is indented by 1em. To change these

dimensions, just substitute the desired value for '10pt' (in both places) or '1em'. The mark is flushright against the footnote.
The standard classes use a simpler macro, in which the footnote text is set like an ordinary text paragraph, with no indentation except on the first line of a paragraph, and the first line of the footnote. Thus, all the macro must do is set `\parindent` to the appropriate value for succeeding paragraphs and put the proper indentation before the mark.

```
\newcommand{\@makefntext}[1]{%
    \parindent 1em%
    \noindent
    \hb@xt@1.8em{\hss\@makefnmark}#1}
```

This class provides a configurable specification.
Normally, if two or more footnotes are sequentially applied:
`...text\footnote{first}\footnote{second}...`
the markers in the text run together (e.g., ...text[12]). The class provides a separator between the markers (e.g., ...text[1,2]). The underlying ideas for this are from the ledmac package [Wil03], which in turn got them from the footmisc package [Fai03].

`\multfootsep` The separator between multiple footnote markers.

```
\newcommand*{\multfootsep}{\textsuperscript{\normalfont,}}
```

`\multiplefootnotemarker` `\m@mmf@prepare` is a pair of self-cancelling kerns.
`\m@mmf@prepare`

```
\newcommand*{\multiplefootnotemarker}{3sp}
\newcommand*{\m@mmf@prepare}{%
  \kern-\multiplefootnotemarker
  \kern\multiplefootnotemarker\relax}
```

`\m@mmf@check` If `\m@mmf@check` recognises the last kern as `\multiplefootnotemarker` it typesets `\multfootsep`.

```
\newcommand*{\m@mmf@check}{%
  \ifdim\lastkern=\multiplefootnotemarker\relax
    \edef\@x@sf{\the\spacefactor}%
    \unkern
    \multfootsep
    \spacefactor\@x@sf\relax
  \fi}

```

`\@footnotetext` We have to modify the kernel's `\@footnotetext` and `\@footnotemark` to
`\m@mold@footnotetext` implement the separator.
`\@footnotemark`

```
\renewcommand{\@footnotetext}[1]{\insert\footins{%
  \def\baselinestretch{\m@m@singlespace}%  <- v1.61803 addition
  \reset@font%            <- v1.6180 addition
  \foottextfont
  \@preamfntext
```

```
  \hsize\columnwidth
  \protected@edef\@currentlabel{%
    \csname p@footnote\endcsname\@thefnmark}%
  \color@begingroup
    \@makefntext{%
      \rule\z@\footnotesep\ignorespaces{\foottextfont #1}%
      \@finalstrut\strutbox}%
  \color@endgroup}\m@mmf@prepare}
\let\m@mold@footnotetext\@footnotetext
\renewcommand*{\@footnotemark}{%
  \leavevmode
  \ifhmode
    \edef\@x@sf{\the\spacefactor}%
    \m@mmf@check
    \nobreak
  \fi
  \@makefnmark
  \m@mmf@prepare
  \ifhmode\spacefactor\@x@sf\fi
  \relax}

```

`\footmarkwidth` `\footmarksep` `\footparindent` The mark is typset right justified in a box with width `\footmarkwidth`. Second and later lines of the text are offset `\footmarksep` from the end of the box. Paragraphs in footnotes are indented by `\parindent`.

```
\newlength{\footmarkwidth}
\newlength{\footmarksep}
\newlength{\footparindent}
```

`\footmarkstyle` `\footscript` The marker is typeset according to `\footscript{`⟨*marker*⟩`}`. This can be specified by the user via `\footmarkstyle`.

```
\newcommand*{\footmarkstyle}[1]{\def\footscript##1{#1}}
```

`\makefootmarkhook` A vacuous macro that the user can redefine to do something useful?

```
\newcommand{\makefootmarkhook}{}

```

`\footfootmark` This macro typesets the footnote marker.

```
\newcommand{\footfootmark}{%
  \ifdim\footmarkwidth < \z@
```

Negative width, mark is in the margin.

```
    \llap{\hb@xt@ -\footmarkwidth{%
            \hss\normalfont\footscript{\@thefnmark}}%
          \hspace*{-\footmarkwidth}}%
  \else
    \ifdim\footmarkwidth = \z@
```

Zero width, mark is at (inside) the margin.

```
      {\normalfont\footscript{\@thefnmark}}%
    \else
```

Positive width.

```
      \hb@xt@\footmarkwidth{\hss\normalfont\footscript{\@thefnmark}}%
    \fi
  \fi}
```

\makefootmark The class version of \@makefntext.

\@makefntext

```
\newcommand{\makefootmark}[1]{%
  \leavevmode
  \parindent \footparindent\noindent
  \leftskip\footmarksep\relax
  \advance\leftskip \footmarkwidth \null\nobreak\hskip -\leftskip\relax
  \makefootmarkhook\relax
  \footfootmark #1}
```

In the thread *memoir and footnote.sty trouble*, 2003/04/02, Patrik Nyman (patrik.nyman@orient.su.se) noted that in order to stop footnote.sty (from mdwtools) barfing my original
\newcommand{\@makefntext}[1]{\makefootmark{#1}}
had to be changed.

```
\newcommand{\@makefntext}[1]{\makefootmark #1}
```

All that now remains for the footer part is to set the defaults.

```
\footmarkstyle{\textsuperscript{#1}}
\setlength{\footmarkwidth}{1.8em}
\setlength{\footmarksep}{-1.8em}
\setlength{\footparindent}{1em}
```

\@makefnmark The footnote markers that are printed in the text to point to the footnotes should be produced by the macro \@makefnmark. We use the default definition for it.

```
%\renewcommand\@makefnmark{\hbox{\@textsuperscript{%
%                                \normalfont\@thefnmark}}}
```

\footref Sometimes it is desireable to reference a footnote more than once. If a footnote is labelled (e.g., \footnote{text\label{fn}}) then the macro \footref{⟨*fn*⟩} will print the footnote marker. The code is essentially a much simplified version of \footnotemark from ltfloat.dtx.

```
\newcommand{\footref}[1]{%
  \begingroup
    \unrestored@protected@xdef\@thefnmark{\ref{#1}}%
  \endgroup
  \@footnotemark}
```

The following code for footnotes that can include verbatims is based on Jeremy Gibbons *'Footnotes with verbatim material'* in the column *'Hey — it works'* (edited by Jeremy Gibbons), TeX and TUG NEWS, vol 2 no. 4, p 9, October 1993. I have updated the basic code so that it works for LaTeX2e.

\verbfootnote
\@xverbfootnote

\verbfootnote is just like \footnote except that it can contain verbatim material.

```
\def\verbfootnote{\@ifnextchar[\@xverbfootnote{\stepcounter\@mpfn
  \protected@xdef\@thefnmark{\thempfn}%
  \@footnotemark\@verbfootnotetext}}

\def\@xverbfootnote[#1]{%
  \begingroup
    \csname c@\@mpfn\endcsname #1\relax
    \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotemark\@verbfootnotetext}

```

\@verbfootnotetext

This is the secret ingredient. It is based on the \footnote macro in *The TEXbook* page 363, which somehow manages to read an argument with verbatims. As Knuth says that it is subtle and involves trickery, don't expect me to even try to explain anything.
I'm not at all sure about the color bits, though!

```
\long\def\@verbfootnotetext{%
  \insert\footins\bgroup
    \footnotesize
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
    \edef\@currentlabel{\csname p@footnote\endcsname\@thefnmark}%
%%%%    \color@begingroup
    \@makefntext{\rule{\z@}{\footnotesep}\ignorespaces}%
    \futurelet\next\fo@t
}
```

\fo@t
\f@@t
\f@t
\@foot

Guess what these do.

```
\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{%
  \color@begingroup
  #1\@foot
  \color@endgroup}
\def\@foot{%
  \strut\egroup
%%%%  \color@endgroup
  \m@mmf@prepare
  }

```

\@verbmpfootnotetext

Footnotes in minipages are a little different, so another version of \...footnotetext.

```
\long\def\@verbmpfootnotetext{%
  \global\setbox\@mpfootins\vbox{%
    \reset@font\footnotesize
    \unvbox\@mpfootins
    \bgroup
    \hsize\columnwidth
    \@parboxrestore
    \edef\@currentlabel{\csname p@mpfootnote\endcsname\@thefnmark}%
    \color@begingroup
    \@makefntext{\rule{\z@}{\footnotesep}\ignorespaces}%
    }
    \futurelet\next\fo@t
}

```

\@minipagerestore To get the \verbfootnote to work in a minipage we have to use \@verbmpfootnotetext instead of \@verbfootnotetext. There is a nice hook in the kernel minipage code for this.
As of version 1.61803 the following code is placed earlier.
\def\@minipagerestore{\let\@verbfootnotetext\@verbmpfootnotetext}

27.6 Major extensions for footnotes

```
%%%%%%%%%%%%% major extension to footnoting

```

This all requires modifications to the output routine's \@makecol, which will be done later.

\setfootnoterule Per Starbäck found (CTT *feetbelowfloat in memoir*, 2006/11/24) that \feetbelowfloat did not affect anything when \raggedbottom is in effect.
\setfootnoterule[⟨*vfill*⟩]{⟨*uplift*⟩}{⟨*width*⟩}{⟨*thickness*⟩} defines the \footnoterule. Memoir's default setting is
\setfootnoterule{3pt}{0.4\columnwidth}{\normalrulethickness}
To force footnotes to the bottom after a \raggedbottom use
\setfootnoterule[\vfill]{3pt}{0.4\columnwidth}{\normalrulethickness}

```
\newcommand*{\setfootnoterule}[4][]{%
  \def\footnoterule{\kern -#2\relax #1\relax
    \hrule width #3\relax
    \kern #2\kern-#4}}
\setfootnoterule{3pt}{0.4\columnwidth}{\normalrulethickness}

```

All sorts of extra kinds of footnotes.

\m@mdoextrafeet These macros are hooks into \@makecol.
\extrafeetins
\extrafeetinshook

```
\newcommand{\m@mdoextrafeet}{\extrafeetins}
\newcommand*{\extrafeetins}{%
  \setbox\@outputbox \vbox{%
    \boxmaxdepth \@maxdepth
```

```
9773     \unvbox\@outputbox
9774     \ifvoid\footinsv@r\else\@footstartv@r\@footgroupv@r\fi
9775     \extrafeetinshook}}
9776 \newcommand{\extrafeetinshook}{}
9777
```

`\m@mdodoreinextrafeet` `\extrafeetreinshook` Hooks into `\@reinserts` for multiple footnote kinds.

```
9778 \newcommand{\m@mdodoreinextrafeet}{%
9779   \ifvoid\footinsv@r\else\insert\footinsv@r{\unvbox\footinsv@r}\fi
9780   \extrafeetreinshook}
9781 \newcommand{\extrafeetreinshook}{}
9782
```

`\foottextfont` `\footinsdim` `\@preamfntext` General macros.

```
9783 \newcommand{\foottextfont}{\footnotesize}
9784 \newlength{\footinsdim}
9785   \setlength{\footinsdim}{8in}   % standard for \dimen\footins
9786 \newcommand{\@preamfntext}{%
9787   \interlinepenalty\interfootnotelinepenalty
9788   \floatingpenalty \@MM
9789   \splittopskip=\footnotesep
9790   \splitmaxdepth=\dp\strutbox
9791   \@parboxrestore}
9792
```

We are going to change some of the kernel footnote macros

`\@mpfootnotetext` `\m@mold@mpfootnotetext`

```
9793 \renewcommand{\@mpfootnotetext}[1]{%
9794   \global\setbox\@mpfootins\vbox{%
9795     \unvbox\@mpfootins
9796     \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
9797     \foottextfont \hsize\columnwidth \@parboxrestore
9798     \protected@edef\@currentlabel{%
9799       \csname p@mpfootnote\endcsname\@thefnmark}%
9800   \color@begingroup
9801   \reset@font%           <- v1.6180 addition
9802     \@makefntext{%
9803       \rule\z@\footnotesep\ignorespaces{\foottextfont #1}%
9804       \@finalstrut\strutbox}%
9805   \color@endgroup}}
9806
```

Save our revised version of `\@mpfootnotetext`

```
9807 \let\m@mold@mpfootnotetext\@mpfootnotetext
9808
```

We also to patch the minipage environment. We can use `\@minipagerestore` for the `begin` part (done earlier), but have to modify `\endminipage`

```
\m@mdoextrafeetmini
 \extrafeetminihook
  \@minipagerestore
\newcommand{\m@mdoextrafeetmini}{%
  \extrafeetminihook}
\newcommand{\extrafeetminihook}{}
%%%%\renewcommand{\@minipagerestore}{\m@mdoextrafeetmini}

   \extrafeetendmini
\extrafeetendminihook
\newcommand{\extrafeetendmini}{%
  \ifvoid\@mpfootinsv@r\else
    \vskip\skip\@mpfootins
    \normalcolor\footnoterule\mp@footgroupv@r
  \fi
  \extrafeetendminihook}
\newcommand{\extrafeetendminihook}{}
```

`\m@mdoextrafeetendmini` This is our patched version of `\endminipage`.

```
        \endminipage
\newcommand{\m@mdoextrafeetendmini}{\extrafeetendmini}
\def\endminipage{%
  \par
  \unskip
  \ifvoid\@mpfootins\else
    \vskip\skip\@mpfootins
    \normalcolor
    \footnoterule
    \unvbox\@mpfootins
  \fi
  \m@mdoextrafeetendmini
  \@minipagefalse
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

`\plainfootnotes` The declaration for standard footnotes; easy, just use the saved versions of `\@footnotetext` and `\@mpfootnotetext`.

```
\newcommand{\plainfootnotes}{%
  \let\@footnotetext\m@mold@footnotetext
  \let\@mpfootnotetext\m@mold@mpfootnotetext}
```

Now for lots of somewhat tedious code, interspersed with complex 'Dirty Tricks' type things.
Robert Schlicht [25] [RS] has provided valuable help in tuning up the code.

[25] Private email, 2004/03/12, (`w.m.l@gmx.net`)

\newfootnoteseries

\newfootnoteseries{⟨*series*⟩} creates the set of macros required for footnote code (see the kernel code presented earlier). The created macros have ⟨*series*⟩ appended to their name.

```
\newcommand{\newfootnoteseries}[1]{%
  \expandafter\newinsert\csname footins#1\endcsname%  -> \footins#1
  \expandafter\skip\csname footins#1\endcsname \bigskipamount%
%%%                                      - > \skip\footins#1 % [RS]
  \newcounter{footnote#1}%                          -> \c@footnote#1
  \@nameuse{c@footnote#1} \z@%                      -> \c@footnote#1=0
  \global\@namelet{p@footnote#1} \@empty%           -> \p@footnote#1
  \@namedef{thefootnote#1}{\arabic{footnote#1}}% -> \thefootnote#1
  \@namedef{foottextfont#1}{\foottextfont}%         -> \foottextfont#1
  \m@makefootnote{#1}%                              -> \footnote#1
  \m@make@xfootnote{#1}%                            -> \@xfootnote#1
  \m@make@footnotetext{#1}%                         -> \@footnotetext#1
  \m@makefootnotemark{#1}%                          -> \footnotemark#1
  \m@make@xfootnotemark{#1}%                        -> \@xfootnotemark#1
  \m@make@footnotemark{#1}%                         -> \@footnotemark#1
  \m@makefootnotetext{#1}%                          -> \footnotetext#1
  \m@make@xfootnotenext{#1}%                        -> \@xfootnotenext#1
  \m@make@mpfn{#1}%                                 -> \@mpfn#1
  \m@makethempfn{#1}%                               -> \thempfn#1
  \m@make@makefnmark{#1}%                           -> \@makefnmark#1
  \m@makefootref{#1}%                               -> \footref#1
  \m@makefootfootmark{#1}%                          -> \footfootmark#1
  \m@makemakefootmark{#1}%                          -> \makefootmark#1
  \m@makefootmarkstyle{#1}%                         -> \footmarkstyle#1
  \@namedef{@makefntext#1}##1{\@nameuse{makefootmark#1} ##1}%
  \m@make@footstart{#1}%                            -> \@footstart#1
  \m@make@footgroup{#1}%                            -> \@footgroup#1
  \expandafter\newinsert\csname @mpfootins#1\endcsname% -> \@mpfootins#1
  \newcounter{mpfootnote#1}%                        -> \c@mpfootnote#1
  \global\@namelet{p@mpfootnote#1}\@empty
  \@namedef{thempfootnote#1}{\itshape\alph{mpfootnote#1}}%
  \m@make@mpfootnotetext{#1}%                       -> \@mpfootnotetext#1
```

Reset the counter per chapter, except for articles.

```
  \ifartopt\else%  [RS]
    \expandafter\@cons\csname cl@chapter\endcsname {{footnote#1}}%
  \fi
```

Add the footnote to the (re)insert hooks.

```
  \g@addto@macro{\extrafeetinshook}{%
    \ifvoid\@nameuse{footins#1}\else
      \@nameuse{@footstart#1}\@nameuse{@footgroup#1}\fi}
  \g@addto@macro{\extrafeetreinshook}{%
    \ifvoid\@nameuse{footins#1}\else
      \insert\@nameuse{footins#1}{\unvbox\@nameuse{footins#1}}\fi}
  \g@addto@macro{\extrafeetendminihook}{%
    \ifvoid\@nameuse{@mpfootins#1}\else
```

```
      \vskip\skip\@mpfootins
      \normalcolor\footnoterule\@nameuse{mp@footgroup#1}\fi}
  \g@addto@macro{\extrafeetminihook}{%
    \@namedef{@mpfn#1}{mpfootnote#1}
    \@namedef{thempfn#1}{\@nameuse{thempfootnote#1}}
    \csname c@mpfootnote#1\endcsname\z@
     \expandafter\let\expandafter\@t@mp \csname @mpfootnotetext#1\endcsname
     \expandafter\let \csname @footnotetext#1\endcsname \@t@mp}
  \g@addto@macro{\@mem@extranofeet}{%  % [RS]
    \ifvoid\@nameuse{footins#1}\else\@mem@nofootfalse\fi}
  \plainfootstyle{#1}%
}

```

`\m@makefootnote` `\m@makefootnote{`⟨*series*⟩`}` creates `\footnote<series>`

```
\newcommand{\m@makefootnote}[1]{
  \@namedef{footnote#1}{\@ifnextchar[
    {\@nameuse{@xfootnote#1}}{%\advance \@nameuse{c@\@mpfn#1} by \@ne
                              \stepcounter{\@mpfn#1}%
      \@name@p@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
      \@nameuse{@footnotemark#1}\@nameuse{@footnotetext#1}}}}

```

`\m@make@xfootnote` `\m@make@xfootnote{`⟨*series*⟩`}` creates `\@xfootnote<series>`

```
\newcommand{\m@make@xfootnote}[1]{
  \@namedef{@xfootnote#1}[##1]{%
  \begingroup
    \csname c@\@mpfn#1\endcsname ##1\relax
    \@name@unresp@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
  \endgroup
  \@nameuse{@footnotemark#1}\@nameuse{@footnotetext#1}}}

```

`\m@make@footnotetext` `\m@make@footnotetext{`⟨*series*⟩`}` creates `\@footnotetext<series>`

```
\newcommand{\m@make@footnotetext}[1]{%
  \@namelongdef{@footnotetext#1}##1{%
  \insert\@nameuse{footins#1}{%
  \def\baselinestretch{\m@m@singlespace}%  <- v1.61803 addition
  \reset@font\@nameuse{foottextfont#1}%
  \@preamfntext
  \hsize\columnwidth
  \protected@edef\@currentlabel{%
    \csname p@footnote#1\endcsname\@nameuse{@thefnmark#1}}%
  \color@begingroup
    \@nameuse{@makefntext#1}{%
      \rule\z@\footnotesep\ignorespaces{\@nameuse{foottextfont#1}##1}% <- v1.6180339a
      \@finalstrut\strutbox}%
  \color@endgroup}%
  \m@mmf@prepare}}

```

\m@make@mpfootnotetext \m@make@mpfootnotetext{⟨*series*⟩} creates \@mpfootnotetext<series>

```
\newcommand{\m@make@mpfootnotetext}[1]{%
  \@namelongdef{@mpfootnotetext#1}##1{%
    \global\setbox\@nameuse{@mpfootins#1}\vbox{%
      \unvbox\@nameuse{@mpfootins#1}%
      \def\baselinestretch{\m@m@singlespace}%  <- v1.61803 addition
      \reset@font\@nameuse{foottextfont#1}%
      \hsize\columnwidth \@parboxrestore
      \protected@edef\@currentlabel{%
        \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
    \color@begingroup
      \@nameuse{@makefntext#1}{%
        \rule\z@\footnotesep\ignorespaces{\@nameuse{foottextfont#1}##1}% <- v1.6180339
        \@finalstrut\strutbox}%
    \color@endgroup}%
    \m@mmf@prepare}}
```

\m@makefootnotemark \m@makefootnotemark{⟨*series*⟩} creates \footnotemark<series>

```
\newcommand{\m@makefootnotemark}[1]{
\@namedef{footnotemark#1}{%
  \@ifnextchar[ {\@nameuse{@xfootnotemark#1}}
    {%\advance\@nameuse{c@footnote#1} by \@ne%
     \stepcounter{footnote#1}%
     \@name@p@xdef{@thefnmark#1}{\@nameuse{thefootnote#1}}%
     \@nameuse{@footnotemark#1}}}}
```

\m@make@xfootnotemark \m@make@xfootnotemark{⟨*series*⟩} creates \@xfootnotemark<series>

```
\newcommand{\m@make@xfootnotemark}[1]{%
  \@namedef{@xfootnotemark#1}[##1]{%
  \begingroup
    \@nameuse{c@footnote#1} ##1\relax
    \@name@unresp@xdef{@thefnmark#1}{\@nameuse{thefootnote#1}}%
  \endgroup
  \@nameuse{@footnotemark#1}}}
```

\m@make@footnotemark \m@make@footnotemark{⟨*series*⟩} creates \@footnotemark<series>

```
\newcommand{\m@make@footnotemark}[1]{%
\@namedef{@footnotemark#1}{%
  \leavevmode
  \ifhmode
    \edef\@x@sf{\the\spacefactor}%
    \m@mmf@check
    \nobreak
  \fi
  \@nameuse{@makefnmark#1}%
  \m@mmf@prepare
```

```
9971   \ifhmode\spacefactor\@x@sf\fi
9972   \relax}}
9973
```

\m@makefootmarkstyle \m@makefootmarkstyle{⟨series⟩} creates \footmarkstyle<series>

```
9974 \newcommand{\m@makefootmarkstyle}[1]{%
9975  \@namedef{footmarkstyle#1}##1{%
9976     \@namedef{footscript#1}####1{##1}}}
9977
```

\m@makefootnotetext \m@makefootnotetext{⟨series⟩} creates \footnotetext<series>

```
9978 \newcommand{\m@makefootnotetext}[1]{%
9979 \@namedef{footnotetext#1}{%
9980   \@ifnextchar[  {\@nameuse{@xfootnotenext#1}}%
9981   {\@name@p@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
9982   \@nameuse{@footnotetext#1}}}}
9983
```

\m@make@xfootnotenext \m@make@xfootnotenext{⟨series⟩} creates \@xfootnotenext<series>

```
9984 \newcommand{\m@make@xfootnotenext}[1]{
9985 \@namedef{@xfootnotenext#1}[##1]{%
9986   \begingroup
9987     \csname c@\@mpfn#1\endcsname ##1\relax
9988     \@name@unresp@xdef{@thefnmark#1}{\@nameuse{thempfn#1}}%
9989   \endgroup
9990   \@nameuse{@footnotetext#1}}}
9991
```

\m@make@mpfn \m@make@mpfn{⟨series⟩} creates \@mpfn<series>

```
9992 \newcommand{\m@make@mpfn}[1]{%
9993   \@namedef{@mpfn#1}{\@nameuse{footnote#1}}}
9994
```

\m@makethempfn \m@makethempfn{⟨series⟩} creates \thempfn<series>

```
9995 \newcommand{\m@makethempfn}[1]{%
9996   \@namedef{thempfn#1}{\@nameuse{thefootnote#1}}}
9997
```

\m@make@makefnmark \m@make@makefnmark{⟨series⟩} creates \@makefnmark<series>

```
9998 \newcommand{\m@make@makefnmark}[1]{%
9999   \@namedef{@makefnmark#1}{%
10000    \hbox{\@textsuperscript{\normalfont\@nameuse{@thefnmark#1}}}}}
10001
```

\m@makefootref \m@makefootref{⟨series⟩} creates \footref<series>

```
10002 \newcommand{\m@makefootref}[1]{%
10003   \@namedef{footref#1}##1{%
10004     \begingroup
10005       \@name@unresp@xdef{@thefnmark#1}{\ref{##1}}%
```

```
10006     \endgroup
10007     \@nameuse{@footnotemark#1}}}
10008
```

\m@makefootfootmark \m@makefootfootmark{⟨series⟩} creates \footfootmark<series>

```
10009 \newcommand{\m@makefootfootmark}[1]{%
10010   \@namedef{footfootmark#1}{%
10011     \ifdim\footmarkwidth < \z@
10012       \llap{\hb@xt@ -\footmarkwidth{%
10013             \hss\normalfont\@nameuse{footscript#1}%
10014             {\@nameuse{@thefnmark#1}}}%
10015             \hspace*{-\footmarkwidth}}%
10016     \else
10017       \ifdim\footmarkwidth = \z@
10018         {\normalfont\@nameuse{footscript#1}{\@nameuse{@thefnmark#1}}}%
10019       \else
10020         \hb@xt@\footmarkwidth{%
10021             \hss\normalfont\@nameuse{footscript#1}%
10022             {\@nameuse{@thefnmark#1}}}%
10023       \fi
10024     \fi}}
10025
```

\m@makemakefootmark \m@makemakefootmark{⟨series⟩} creates \makefootmark<series>

```
10026 \newcommand{\m@makemakefootmark}[1]{%
10027   \@namedef{makefootmark#1}##1{%
10028     \leavevmode
10029     \parindent \footparindent\noindent
10030     \leftskip\footmarksep\relax
10031     \advance\leftskip \footmarkwidth
10032     \null\nobreak\hskip -\leftskip\relax
10033     \makefootmarkhook\relax
10034     \@nameuse{footfootmark#1}##1}}
10035
```

\m@make@footgroup \m@make@footgroup{⟨series⟩} creates \@footgroup<series>

```
10036 \newcommand{\m@make@footgroup}[1]{%
10037   \@namedef{@footgroup#1}{\unvbox\@nameuse{footins#1}}}
10038
```

\m@makemp@footgroup \m@makemp@footgroup{⟨series⟩} creates \mp@footgroup<series>

```
10039 \newcommand{\m@makemp@footgroup}[1]{%
10040   \@namedef{mp@footgroup#1}{\unvbox\@nameuse{@mpfootins#1}}}
10041
```

\m@make@footstart \m@make@footstart{⟨series⟩} creates \@footstart<series>

```
10042 \newcommand{\m@make@footstart}[1]{%
10043   \@namedef{@footstart#1}{%
10044     \vskip\bigskipamount
```

```
    \leftskip=\z@
    \rightskip=\z@
    \footnoterule}}
```

`\plainfootstyle` `\plainfootstyle{`⟨*series*⟩`}` specifies a plain (normal) footnote style for ⟨*series*⟩.

```
\newcommand{\plainfootstyle}[1]{%
  \m@make@footnotetext{#1}%
  \m@make@footgroup{#1}%
  \m@make@footstart{#1}%
  \m@make@mpfootnotetext{#1}%
  \m@makemp@footgroup{#1}%
  \@nameuse{footmarkstyle#1}{\textsuperscript{##1}}
  \expandafter\dimen\csname footins#1\endcsname=\footinsdim
  \expandafter\count\csname footins#1\endcsname=1000\relax}
```

Now the basic code for footnote declarations (we will be using `footnotev@r` instead of the regular `footnote`).

`\footinsv@r` Define the new `\footinsv@r` insert.

```
\newinsert\footinsv@r
  \skip\footinsv@r\bigskipamount
  \count\footinsv@r=1000 % no magnifcation
  \dimen\footinsv@r=\footinsdim
```

Create a new `v@r` footnote series.

```
\m@make@footstart{v@r}
```

`\@footgroupv@r` Initialise the `footgroup` for the series to do nothing.

```
\newcommand{\@footgroupv@r}{}
```

And also for minipages:

`\@mpfootinsv@r`
`\mp@footgroupv@r`

```
\newinsert\@mpfootinsv@r
\newcommand{\mp@footgroupv@r}{}
```

ledmac package style footnotes (see `ledmac.dtx` for more detailed explanations). The two- and three-column notes use a macro `\m@mrigidbalance` to split text into a number of columns. This is based on *The TEXbook*, page 397.

`\m@m@k`
`\m@m@h`

```
\newcount\m@m@k \newdimen\m@m@h
```

\m@mrigidbalance
\m@mdosplits
\m@msplitoff

\m@mrigidbalance{⟨*box*⟩}{⟨*num*⟩}{⟨*length*⟩} splits a ⟨*box*⟩ (of text) into ⟨*num*⟩ columns with ⟨*length*⟩ space between the top baseline and the top of the \vbox.

```
\newcommand*{\m@mrigidbalance}[3]{\setbox0=\box#1 \m@m@k=#2 \m@m@h=#3
  \@@line{\splittopskip=\m@m@h \vbadness=\@M \hfilneg
  \valign{##\vfill\cr\m@mdosplits}}}
\newcommand*{\m@mdosplits}{\ifnum\m@m@k>0 \noalign{\hfil}\m@msplitoff
  \global\advance\m@m@k-1\cr\m@mdosplits\fi}
\newcommand*{\m@msplitoff}{\dimen0=\ht0
  \divide\dimen0 by\m@m@k \advance\dimen0 by\m@m@h
  \setbox2 \vsplit0 to \dimen0
  \unvbox2 }
```

27.6.1 Two column footnotes

\twocolumnfootnotes

Declaration for two column footnotes. This causes the standard \footnote and friends to internally use the v@r series, which here produces two column footnotes.

```
\newcommand{\twocolumnfootnotes}{%
  \@namedef{foottextfontv@r}{\foottextfont}%  % [RS]
  \let\@footnotetext\@twocolfootnotetext
  \dimen\footinsv@r=2\footinsdim
  \count\footinsv@r=500\relax
  \m@make@twocol@footgroup{v@r}%
  \let\@footgroupv@r\@twocol@footgroupv@r
  \let\@mpfootnotetext\@mptwocolfootnotetext
  \m@make@mptwocol@footgroup{v@r}%
  \let\mp@footgroupv@r\@mptwocol@footgroupv@r}
```

\@twocolfootnotetext

\@twocolfootnotetext{⟨*text*⟩} is the two column version of \@footnotetext for the v@r series.

```
\newcommand{\@twocolfootnotetext}[1]{\insert\footinsv@r{%
  \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
  \reset@font\foottextfont
  \@preamfntext
  \protected@edef\@currentlabel{%
    \csname p@footnote\endcsname\@thefnmark}%
  \color@begingroup
    \@twocolfootfmt{#1}%
  \color@endgroup}%
  \m@mmf@prepare}
```

\@preamtwofmt

Give each column 0.45 of the textwidth.

```
\newcommand{\@preamtwofmt}{%
  \hsize .45\hsize
  \parindent=\z@
  \tolerance=5000\relax
```

```
10106   \raggedright
10107   \leavevmode}
10108
```

\@twocolfootfmt \@twocolfootfmt

```
10109 \newcommand{\@twocolfootfmt}[1]{%
10110   \@preamtwofmt
10111   {\footfootmark\strut {\foottextfont #1}\strut\par}\allowbreak}
10112
```

\@mptwocolfootnotetext \@mptwocolfootnotetext{⟨*text*⟩} is the two column version of \@mpfootnotetext for the v@r series for minipages.

```
10113 \newcommand{\@mptwocolfootnotetext}[1]{%
10114   \global\setbox\@mpfootinsv@r\vbox{%
10115     \unvbox\@mpfootinsv@r
10116     \def\baselinestretch{\m@m@singlespace}%    <- v1.61803 addition
10117     \reset@font\foottextfont
10118     \hsize\columnwidth \@parboxrestore
10119     \protected@edef\@currentlabel{%
10120       \csname p@mpfootnote\endcsname\@thefnmark}%
10121   \color@begingroup
10122     \@twocolfootfmt{#1}%
10123   \color@endgroup}%
10124   \m@mmf@prepare}
10125
```

\twocolumnfootstyle \twocolumnfootstyle{⟨*series*⟩} specifies a two column footnote style for ⟨*series*⟩.

```
10126 \newcommand{\twocolumnfootstyle}[1]{%
10127   \m@make@twocolfootnotetext{#1}%
10128   \m@make@mptwocolfootnotetext{#1}%
10129   \m@make@twocolfootfmt{#1}%
10130   \m@make@twocol@footgroup{#1}%
10131   \m@make@mptwocol@footgroup{#1}%
10132   \m@make@footstart{#1}%
10133   \@namelongdef{@footnotetext#1}##1{%
10134      \@nameuse{@twocolfootnotetext#1}{##1}}%
10135   \@namelongdef{@mpfootnotetext#1}##1{%
10136      \@nameuse{@mptwocolfootnotetext#1}{##1}}%
10137   \@namedef{@footgroup#1}{\@nameuse{@twocol@footgroup#1}}%
10138   \@namedef{mp@footgroup#1}{\@nameuse{@mptwocol@footgroup#1}}%
10139   \expandafter\dimen\csname footins#1\endcsname=2\footinsdim
10140   \expandafter\count\csname footins#1\endcsname=500\relax}
10141
```

\m@make@twocolfootnotetext \m@make@twocolfootnotetext{⟨*series*⟩} creates \@twocolfootnotetext⟨*series*⟩

```
10142 \newcommand{\m@make@twocolfootnotetext}[1]{%
10143   \@namelongdef{@twocolfootnotetext#1}##1{%
10144     \insert\@nameuse{footins#1}{%
```

```
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
    \reset@font\@nameuse{foottextfont#1}%
    \@preamfntext
    \protected@edef\@currentlabel{%
      \csname p@footnote#1\endcsname \@nameuse{@thefnmark#1}}%
    \color@begingroup
      \@nameuse{@twocolfootfmt#1}{##1}%
    \color@endgroup}%
    \m@mmf@prepare}}

```

\m@make@mptwocolfootnotetext

\m@make@mptwocolfootnotetext{⟨*series*⟩} creates \@mptwocolfootnotetext⟨*series*⟩

```
\newcommand{\m@make@mptwocolfootnotetext}[1]{%
\@namelongdef{@mptwocolfootnotetext#1}##1{%
  \global\setbox\@nameuse{@mpfootins#1}\vbox{%
    \unvbox\@nameuse{@mpfootins#1}
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
    \reset@font\@nameuse{foottextfont#1}%
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
  \color@begingroup
    \@nameuse{@twocolfootfmt#1}{##1}%
  \color@endgroup}\m@mmf@prepare}}

```

\m@make@twocolfootfmt

\m@make@twocolfootfmt{⟨*series*⟩} creates \@twocolfootfmt⟨*series*⟩

```
\newcommand{\m@make@twocolfootfmt}[1]{%
  \@namedef{@twocolfootfmt#1}##1{%
    \@preamtwofmt
    {\@nameuse{footfootmark#1}\strut
     {\@nameuse{foottextfont#1}##1}\strut\par}\allowbreak}}

```

\m@make@twocol@footgroup

\m@make@twocol@footgroup{⟨*series*⟩} creates \@twocol@footgroup⟨*series*⟩

```
\newcommand{\m@make@twocol@footgroup}[1]{%
  \@namedef{@twocol@footgroup#1}{{%
    \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
     \m@mrigidbalance{\@nameuse{footins#1}}{\tw@}{\splittopskip}}}}

```

\m@make@mptwocol@footgroup

\m@make@mptwocol@footgroup{⟨*series*⟩} creates \@mptwocol@footgroup⟨*series*⟩

```
\newcommand{\m@make@mptwocol@footgroup}[1]{%
\@namedef{@mptwocol@footgroup#1}{{%
  \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
  \m@mrigidbalance{\@nameuse{@mpfootins#1}}{\tw@}{\splittopskip}}}}

```

27.6.2 Three column footnotes

\threecolumnfootnotes Declaration for three column footnotres.

```
\newcommand{\threecolumnfootnotes}{%
  \@namedef{foottextfontv@r}{\foottextfont}%  % [RS]
  \let\@footnotetext\@threecolfootnotetext
  \dimen\footinsv@r=3\footinsdim
  \count\footinsv@r=333\relax
  \m@make@threecol@footgroup{v@r}%
  \let\@footgroupv@r\@threecol@footgroupv@r
  \let\@mpfootnotetext\@mpthreecolfootnotetext
  \m@make@mpthreecol@footgroup{v@r}%
  \let\mp@footgroupv@r\@mpthreecol@footgroupv@r}

```

\@threecolfootnotetext \@threecolfootnotetext{⟨*text*⟩} is the three column version of \@footnotetext

```
\newcommand{\@threecolfootnotetext}[1]{\insert\footinsv@r{%
  \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
  \reset@font\foottextfont
  \@preamfntext
  \protected@edef\@currentlabel{%
    \csname p@footnote\endcsname\@thefnmark}%
  \color@begingroup
    \@threecolfootfmt{#1}%
  \color@endgroup}\m@mmf@prepare}

```

\@preamthreefmt Give each column 0.3 of the text width.

```
\newcommand{\@preamthreefmt}{%
  \hsize .3\hsize
  \parindent=\z@
  \tolerance=5000\relax
  \raggedright
  \leavevmode}

```

\@threecolfootfmt

```
\newcommand{\@threecolfootfmt}[1]{%
  \@preamthreefmt
  {\footfootmark\strut {\foottextfont #1}\strut\par}\allowbreak}

```

\@mpthreecolfootnotetext \@mpthreecolfootnotetext{⟨*text*⟩} is the three column version of \@mpfootnotetext

```
\newcommand{\@mpthreecolfootnotetext}[1]{%
  \global\setbox\@mpfootinsv@r\vbox{%
    \unvbox\@mpfootinsv@r
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
```

```
    \reset@font\foottextfont
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@mpfootnote\endcsname\@thefnmark}%
  \color@begingroup
    \@threecolfootfmt{#1}%
  \color@endgroup}\m@mmf@prepare}
```

`\threecolumnfootstyle` `\threecolumnfootstyle{`⟨*series*⟩`}` specifies three column footnote style for ⟨*series*⟩

```
\newcommand{\threecolumnfootstyle}[1]{%
  \m@make@threecolfootnotetext{#1}%
  \m@make@mpthreecolfootnotetext{#1}%
  \m@make@threecolfootfmt{#1}%
  \m@make@threecol@footgroup{#1}%
  \m@make@mpthreecol@footgroup{#1}%
  \m@make@footstart{#1}%
  \@namelongdef{@footnotetext#1}##1{%
    \@nameuse{@threecolfootnotetext#1}{##1}}%
  \@namelongdef{@mpfootnotetext#1}##1{%
    \@nameuse{@mpthreecolfootnotetext#1}{##1}}%
  \@namedef{@footgroup#1}{\@nameuse{@threecol@footgroup#1}}%
  \@namedef{mp@footgroup#1}{\@nameuse{@mpthreecol@footgroup#1}}%
  \expandafter\dimen\csname footins#1\endcsname=3\footinsdim
  \expandafter\count\csname footins#1\endcsname=333\relax}
```

`\m@make@threecolfootnotetext` `\m@make@threecolfootnotetext{`⟨*series*⟩`}` creates `\@threecolfootnotetext`⟨*series*⟩

```
\newcommand{\m@make@threecolfootnotetext}[1]{%
\@namelongdef{@threecolfootnotetext#1}##1{%
  \insert\@nameuse{footins#1}{%
  \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
  \reset@font\@nameuse{foottextfont#1}%
  \@preamfntext
  \protected@edef\@currentlabel{%
    \csname p@footnote#1\endcsname \@nameuse{@thefnmark#1}}%
  \color@begingroup
    \@nameuse{@threecolfootfmt#1}{##1}%
  \color@endgroup}\m@mmf@prepare}}
```

`\m@make@mpthreecolfootnotetext` `\m@make@mpthreecolfootnotetext{`⟨*series*⟩`}` creates `\@mpthreecolfootnotetext`⟨*series*⟩

```
\newcommand{\m@make@mpthreecolfootnotetext}[1]{%
\@namelongdef{@mpthreecolfootnotetext#1}##1{%
  \global\setbox\@nameuse{@mpfootins#1}\vbox{%
    \unvbox\@nameuse{@mpfootins#1}
```

```
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
    \reset@font\@nameuse{foottextfont#1}%
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
  \color@begingroup
    \@nameuse{@threecolfootfmt#1}{##1}%
  \color@endgroup}\m@mmf@prepare}}

```

\m@make@threecolfootfmt `\m@make@threecolfootfmt{⟨series⟩}` creates `\@threecolfootfmt⟨series⟩`

```
\newcommand{\m@make@threecolfootfmt}[1]{%
\@namelongdef{@threecolfootfmt#1}##1{%
  \@preamthreefmt
  {\@nameuse{footfootmark#1}\strut
   {\@nameuse{foottextfont#1}##1}\strut\par}\allowbreak}}

```

m@make@threecol@footgroup `\m@make@threecol@footgroup{⟨series⟩}` creates `\@threecol@footgroup⟨series⟩`

```
\newcommand{\m@make@threecol@footgroup}[1]{%
\@namedef{@threecol@footgroup#1}{{%
  \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
  \m@mrigidbalance{\@nameuse{footins#1}}{\thr@@}{\splittopskip}}}}

```

make@mpthreecol@footgroup `\m@make@mpthreecol@footgroup{⟨series⟩}` creates `\@mpthreecol@footgroup⟨series⟩`

```
\newcommand{\m@make@mpthreecol@footgroup}[1]{%
\@namedef{@mpthreecol@footgroup#1}{{%
  \@nameuse{foottextfont#1} \splittopskip=\ht\strutbox
  \m@mrigidbalance{\@nameuse{@mpfootins#1}}{\thr@@}{\splittopskip}}}}

```

27.6.3 Paragraphed footnotes

Paragraphed footnotes are based on *The TEXbook*, page 398ff, in the Dirty Tricks appendix. It does a lot of box manipulations.

\m@munvxh `\m@munvxh{⟨vbox⟩}`: unvbox, extract the last line, and unhbox it.

```
\newcommand{\m@munvxh}[1]{%
  \setbox0=\vbox{\unvbox#1%
    \global\setbox1=\lastbox}%
  \unhbox1
  \unskip
  \unskip
  \unpenalty
  \hskip\m@mipn@skip}

```

\m@mungebox

```
\newcommand{\m@mungebox}{%
  \setbox0=\hbox{\m@munvxh0}%
  \dp0=\z@
  \ht0=\footfudgefactor\wd0
  \box0
  \penalty0}
```

\m@mipn@skip
\m@minterparanoteglue

These are 'inter-para-note-skip' and 'inter-parafootnote' glue, for paragraphed footnotes.

```
\newskip\m@mipn@skip
\newcommand*{\m@minterparanoteglue}[1]{%
  {\foottextfont\global\m@mipn@skip=#1\relax}}
\m@minterparanoteglue{1em plus.4em minus.4em}
```

\m@mmakehboxofhboxes

Make an hbox of hboxes.

```
\newcommand*{\m@mmakehboxofhboxes}{\setbox0=\hbox{}%
  \loop
    \unpenalty
    \setbox2=\lastbox
  \ifhbox2
    \setbox0=\hbox{\box2\unhbox0}
  \repeat}
```

\m@mremovehboxes

```
\newcommand*{\m@mremovehboxes}{\setbox0=\lastbox
  \ifhbox0{\m@mremovehboxes}\unhbox0 \fi}
```

\footfudgefiddle

TeX uses `\footfufgefactor` to estimate the space required for paragraphed footnotes. If it underestimates then the notes approach, or cover, the footer. `\footfudgefiddle` can be changed (upward) from its default to improve matters.

```
\newcommand*{\footfudgefiddle}{64}
```

\paragraphfootnotes

Declaration for paragraphed footnotes.

```
\newcommand{\paragraphfootnotes}{%
  \@namedef{foottextfontv@r}{\foottextfont}%  % [RS]
  \let\@footnotetext\@parafootnotetext
  \dimen\footinsv@r=\footinsdim
  \count\footinsv@r=1000\relax
  \m@make@para@footgroup{v@r}%
  \let\@footgroupv@r\@para@footgroupv@r
  \let\@mpfootnotetext\@mpparafootnotetext
  \m@make@mppara@footgroup{v@r}%
  \let\mp@footgroupv@r\@mppara@footgroupv@r
```

```
  {\foottextfont
   \dimen0=\baselineskip
   \multiply\dimen0 by 1024
   \divide\dimen0 by \hsize \multiply\dimen0 by \footfudgefiddle
   \xdef\footfudgefactor{\expandafter\strip@pt\dimen0 }}}

```

`\@parafootnotetext` `\@parafootnotetext{`⟨*text*⟩`}` is a paragraph version of `\@footnotetext`

```
\newcommand{\@parafootnotetext}[1]{\insert\footinsv@r{
  \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
  \reset@font\foottextfont
  \@preamfntext
  \protected@edef\@currentlabel{%
    \csname p@footnote\endcsname\@thefnmark}%
  \setbox0=\vbox{\hsize=\maxdimen
    \color@begingroup
      \noindent \@parafootfmt{#1}%
    \color@endgroup}%
  \m@mungebox}\m@mmf@prepare}

```

`\@parafootfmt` `\@parafootfmt{`⟨*text*⟩`}` is a paragraph version of `\@footfmt`

```
\newcommand{\@parafootfmt}[1]{%
  \parindent=\z@
  \parfillskip=0pt \@plus 1fil
  {\footfootmark\strut {\foottextfont #1}\penalty-10}}

```

`\@mpparafootnotetext` `\@mpparafootnotetext{`⟨*text*⟩`}` is a paragraph version of `\@mpfootnotetext`

```
\newcommand{\@mpparafootnotetext}[1]{%
  \global\setbox\@mpfootinsv@r\vbox{%
    \unvbox\@mpfootinsv@r
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
    \reset@font\foottextfont
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@mpfootnote\endcsname\@thefnmark}%
    \setbox0=\vbox{\hsize=\maxdimen
      \color@begingroup
      \noindent \@parafootfmt{#1}%
      \color@endgroup}%
  \m@mungebox}\m@mmf@prepare}

```

`\paragraphfootstyle` `\paragraphfootstyle{`⟨*series*⟩`}` specifies paragraphed footnote style for ⟨*series*⟩.

```
\newcommand{\paragraphfootstyle}[1]{%
  \m@make@parafootnotetext{#1}%
  \m@make@mpparafootnotetext{#1}%
```

```
\m@make@parafootfmt{#1}%
\m@make@para@footgroup{#1}%
\m@make@mppara@footgroup{#1}%
\m@make@para@footstart{#1}%
\@namelongdef{@footnotetext#1}##1{%
  \@nameuse{@parafootnotetext#1}{##1}}%
\@namelongdef{@mpfootnotetext#1}##1{%
  \@nameuse{@mpparafootnotetext#1}{##1}}%
\@namedef{@footgroup#1}{\@nameuse{@para@footgroup#1}}%
\@namedef{mp@footgroup#1}{\@nameuse{@mppara@footgroup#1}}%
\@namedef{@footstart#1}{\@nameuse{@para@footstart#1}}%
\expandafter\dimen\csname footins#1\endcsname=\footinsdim
\expandafter\count\csname footins#1\endcsname=1000\relax
{\@nameuse{foottextfont#1}%
 \dimen0=\baselineskip
 \multiply\dimen0 by 1024
 \divide\dimen0 by \hsize \multiply\dimen0 by 64
 \xdef\footfudgefactor{\expandafter\strip@pt\dimen0 }}}

```

`\m@make@parafootnotetext` `\m@make@parafootnotetext{`⟨*series*⟩`}` creates `\@parafootnotetext`⟨*series*⟩

```
\newcommand{\m@make@parafootnotetext}[1]{%
\@namelongdef{@parafootnotetext#1}##1{%
  \insert\@nameuse{footins#1}{
  \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
  \reset@font\@nameuse{foottextfont#1}%
  \@preamfntext
  \protected@edef\@currentlabel{%
    \csname p@footnote#1\endcsname \@nameuse{@thefnmark#1}}%
  \setbox0=\vbox{\hsize=\maxdimen
    \color@begingroup
      \noindent \@nameuse{@parafootfmt#1}{##1}%
    \color@endgroup}%
  \m@mungebox}\m@mmf@prepare}}

```

`\m@make@mpparafootnotetext` `\m@make@mpparafootnotetext{`⟨*series*⟩`}` creates `\@mpparafootnotetext`⟨*series*⟩

```
\newcommand{\m@make@mpparafootnotetext}[1]{%
\@namelongdef{@mpparafootnotetext#1}##1{%
  \global\setbox\@nameuse{@mpfootins#1}\vbox{%
    \unvbox\@nameuse{@mpfootins#1}
    \def\baselinestretch{\m@m@singlespace}%   <- v1.61803 addition
    \reset@font\@nameuse{foottextfont#1}%
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
      \csname p@mpfootnote#1\endcsname\@nameuse{@thefnmark#1}}%
    \setbox0=\vbox{\hsize=\maxdimen
      \color@begingroup
        \noindent \@nameuse{@parafootfmt#1}{##1}%
      \color@endgroup}%
```

```
10415     \m@mungebox}\m@mmf@prepare}}
10416
```

`\m@make@parafootfmt` `\m@make@parafootfmt{`⟨*series*⟩`}` creates `\@parafootfmt`⟨*series*⟩

```
10417 \newcommand{\m@make@parafootfmt}[1]{%
10418 \@namelongdef{@parafootfmt#1}##1{%
10419   \parindent=\z@
10420   \parfillskip=0pt \@plus 1fil
10421   {\@nameuse{footfootmark#1}\strut
10422    {\@nameuse{foottextfont#1}##1}\penalty-10}}}
10423
```

`\m@make@para@footgroup` `{`⟨*series*⟩`}` creates `\@para@footgroup`⟨*series*⟩

```
10424 \newcommand{\m@make@para@footgroup}[1]{%
10425 \@namedef{@para@footgroup#1}{%
10426   \unvbox\@nameuse{footins#1}
10427   \m@mmakehboxofhboxes
10428   \setbox0=\hbox{\unhbox0 \m@mremovehboxes}%
10429   \@nameuse{foottextfont#1}%
10430   \noindent\unhbox0\par}}
10431
```

`\m@make@mppara@footgroup` `\m@make@mppara@footgroup{`⟨*series*⟩`}` creates `\@mppara@footgroup`⟨*series*⟩

```
10432 \newcommand{\m@make@mppara@footgroup}[1]{%
10433 \@namedef{@mppara@footgroup#1}{%
10434   \unvbox\@nameuse{@mpfootins#1}
10435   \m@mmakehboxofhboxes
10436   \setbox0=\hbox{\unhbox0 \m@mremovehboxes}%
10437   \@nameuse{foottextfont#1}%
10438   \noindent\unhbox0\par}}
10439
```

`\m@make@para@footstart` `\m@make@para@footstart{`⟨*series*⟩`}` creates `\@para@footstart`⟨*series*⟩

```
10440 \newcommand{\m@make@para@footstart}[1]{%
10441 \@namedef{@para@footstart#1}{%
10442   \vskip\bigskipamount
10443   \leftskip=\z@
10444   \rightskip=\z@
10445   \parindent=\z@
10446   \vskip\skip\@nameuse{footins#1}%
10447   \footnoterule}}
10448
```

27.7 Nasty insert bits

`\sidebar` is implemented as a new kind of `\insert` and the extended footnotes are also implemented via new kinds of `\insert`s. New `\insert`s mean that the kernel's `\@makecol` must be revised to cater for them. As this is part of the output process it is nasty to do.

Before v1.61803 \@makecol got (re)defined at several points in the code. Here is what I hope is the final version od \@makecol and associated code after having resolved and combined all the changes.

\if@mem@nofoot
\@mem@testifnofoot
\@mem@extranofoot

These are from [RS] as \@doclearpage should check for more than just \footins being void. Note that this covers all current class defined inserts, including sidebars.

```
\newif\if@mem@nofoot
\newcommand*{\@mem@testifnofoot}{%
  \@mem@nofoottrue
  \ifvoid\footins\else\@mem@nofootfalse\fi
  \ifvoid\footinsv@r\else\@mem@nofootfalse\fi
  \ifvoid\sideins\else\@mem@nofootfalse\fi
  \@mem@extranofeet}
\newcommand*{\@mem@extranofeet}{}

```

\memold@doclearpage
\mem@doclearpage
\@doclearpage

I thought that I could get away with using the kernel's \@doclearpage but [RS] discovered that I couldn't.

```
\let\memold@doclearpage\@doclearpage
\newcommand{\mem@doclearpage}{%
  \@mem@testifnofoot
  \if@mem@nofoot
    \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
    \setbox\@tempboxa\box\@cclv
    \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
    \global\let\@toplist\@empty
    \global\let\@botlist\@empty
    \global\@colroom\@colht
    \ifx \@currlist\@empty
    \else
      \@latexerr{Float(s) lost}\@ehb
      \global\let\@currlist\@empty
    \fi
    \@makefcolumn\@deferlist
    \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
    \if@twocolumn
      \if@firstcolumn
        \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
        \global\let\@dbltoplist\@empty
        \global\@colht\textheight
        \begingroup
          \@dblfloatplacement
          \@makefcolumn\@dbldeferlist
          \@whilesw\if@fcolmade \fi{\@outputpage
                                    \@makefcolumn\@dbldeferlist}%
        \endgroup
      \else
        \vbox{}\clearpage
```

```
      \fi
    \fi
  \else
    \setbox\@cclv\vbox{\box\@cclv\vfil}%
    \@makecol\@opcol
    \clearpage
  \fi}
```

Replace the kernel's `\@doclearpage`.

```
\gdef\@doclearpage{\mem@doclearpage}
```

`\m@m@makecolfloats` `\m@m@makecoltext` These two macros contain code common to versions of `\@makecol`

```
\newcommand*{\m@m@makecolfloats}{%
  \xdef\@freelist{\@freelist\@midlist}%
  \global\let\@midlist\@empty
  \@combinefloats}
\newcommand*{\m@m@makecoltext}{%
  \ifvbox\@kludgeins
    \@makespecialcolbox
  \else
    \setbox\@outputbox \vbox to\@colht{%
      \@texttop
      \dimen@ \dp\@outputbox
      \unvbox \@outputbox
      \vskip -\dimen@
      \@textbottom}%
  \fi}
```

`\m@m@makecolintro` A hook into the revised `\@makecol`

```
\newcommand*{\m@m@makecolintro}{}
```

`\m@mopfootnote` (footnote) code for possible use in `\@makecol`.

```
\newcommand*{\m@mopfootnote}{\setbox\@outputbox \vbox{%
  \boxmaxdepth\@maxdepth
  \@tempdima\dp\@cclv
  \unvbox\@cclv
  \vskip-\@tempdima
  \vskip \skip\footins
  \color@begingroup
    \normalcolor
    \footnoterule
    \unvbox \footins
  \color@endgroup}}
```

`\m@mopfootnotebf` (footnote) code in support of footnotes below floats. Problem with original code noted by Jørgen Larsen (`jl@ruc.dk`) on 2008/05/24.

```
\newcommand*{\m@mopfootnotebf}{%
  \setbox\@outputbox \vbox{%
  \boxmaxdepth\@maxdepth
  \unvbox\@outputbox
  \vskip\skip\footins
  \color@begingroup
    \normalcolor
    \footnoterule
    \unvbox \footins
  \color@endgroup}}

```

`\m@mopsidebar` (sidebar) code for possible use in `\@makecol`. From DA's latest sidebar fixes.

```
\newcommand*{\m@mopsidebar}{%
  \ifvoid\sideins\else
    \setbox\@outputbox \vbox{%
      \sidecontents
      \unvbox\@outputbox}
  \fi}

```

`\mem@makecol` DA's latest version of `\@makecol` (giving the standard footnote order (bottom floats after footnotes)) putting the sidebar insert after the others (from mempatch v4.9).

```
\gdef\mem@makecol{%
  \m@m@makecolintro
  \ifvoid\footins
    \setbox\@outputbox \box\@cclv
  \else
    \m@mopfootnote
  \fi
  \m@mdoextrafeet
  \m@m@makecolfloats
  \m@mopsidebar
  \m@m@makecoltext
  \global \maxdepth \@maxdepth}

```

`\mem@makecolbf` A version of `\@makecol` which puts footnotes at the bottom of the page (after any bottom floats).

```
\gdef\mem@makecolbf{%
  \m@m@makecolintro
  \setbox\@outputbox \box\@cclv
  \m@m@makecolfloats
  \ifvoid\footins\else
    \m@mopfootnotebf
  \fi
  \m@mdoextrafeet
  \m@mopsidebar
```

```
  \m@m@makecoltext
  \global\maxdepth \@maxdepth}

\mem@makecoldblf A version of \@makecol which is a placeholder to fix the doublefloat problem.

\gdef\mem@makecoldblf{%
  \m@m@makecolintro
  \setbox\@outputbox \box\@cclv
  \m@m@makecolfloats
  \m@mopsidebar%    <- added
  \ifvoid\footins
  \else
    \m@mopfootnote
  \fi
  \m@mdoextrafeet
  \m@m@makecoltext
  \global \maxdepth \@maxdepth}

\feetabovefloat declarations to put footnotes above bottom floats (standard LaTeX) or at the
\feetbelowfloat bottom of the page
      \@makecol
\newcommand{\feetabovefloat}{\gdef\@makecol{\mem@makecol}}
\newcommand{\feetbelowfloat}{\gdef\@makecol{\mem@makecolbf}}
\feetabovefloat

\@reinserts DA's final version from last patch.

\gdef\@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \m@mdodoreinextrafeet
  \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
  \ifvoid\sideins\else\insert\sideins{\unvbox\sideins}\fi}
```

27.8 Side footnotes

27.8.1 Extension to the regular footnote

The code here enables the regular \footnote to be set in the margin. It is based on Robin Fairbairns' footmisc package.

```
\ifm@mfnmargin Declarations for regular footnotes to be set as normal at the foot of the page, or
\footnotesatfoot in the margin.
\footnotesinmargin
\newif\ifm@mfnmargin
\newcommand*{\footnotesatfoot}{\m@mfnmarginfalse}
\newcommand*{\footnotesinmargin}{\m@mfnmargintrue}
\footnotesatfoot
```

\@footnotetext Change \@footnotetext to enable footnotes to be put into the margin by using \marginpar.

```
\renewcommand{\@footnotetext}[1]{%
  \ifm@mfnmargin%   use marginpar
```

Use \marginpar for placing the footnote.

```
    \marginpar{%
    \def\baselinestretch{\m@m@singlespace}%
    \reset@font
    \foottextfont
    \protected@edef\@currentlabel{%
      \csname p@footnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{\ignorespaces#1}%
    \color@endgroup}%
  \else% regular feet
```

This is the original code for \@footnotetext.

```
    \insert\footins{%
    \def\baselinestretch{\m@m@singlespace}%
    \reset@font
    \foottextfont
    \@preamfntext
    \hsize\columnwidth
    \protected@edef\@currentlabel{%
      \csname p@footnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces{\foottextfont #1}%
        \@finalstrut\strutbox}%
    \color@endgroup}%
  \fi%
  \m@mmf@prepare}

```

27.9 Bottom aligned side footnotes

The code for the bottom aligned marginal footnotes is based partly on the \sidebar code and code from David Salomon, 'Output Routines: Examples and Techniques. Part III: Insertions', *TUGboat*, 11, 4, pp 588–605, Nov 1990.

\sidefootmargin \m@msidefoot@margin \sidefootmargin is the user command for setting the side for side footnotes (stored as \m@msidefoot@margin). The default is outer.

```
\newcommand*{\sidefootmargin}[1]{%
  \m@msetm@argin{#1}%
  \ifnum\m@mm@argin<\z@
    \@memwarn{Bad \string\sidefootmargin\space argument `#1'\MessageBreak
              set to `outer'}%
    \gdef\m@msidefoot@margin{2}%   set as outer
  \else
```

```
    \global\let\m@msidefoot@margin\m@mm@argin
  \fi}
\sidefootmargin{outer}
```

`\sidefootins` As for any non-main text matter we need an insert.

```
\newinsert\sidefootins
  \skip\sidefootins=0pt
  \count\sidefootins=0\relax
```

`\sidefootadjust` `\sidefootheight` `\setsidefootheight` The length `\sidefootadjust` can be used for fine control over the vertivcal position of the base of the column of side footnotes. The length `\sidefootheight` is the height of the column and is best set by the `\setsidefootheight` macro, which takes the desired height as its argument and sets the height of the `\sidefootins` `\insert`. The height is initially set to the `\textheight`.

```
\newlength{\sidefootadjust}
  \setlength{\sidefootadjust}{0pt}
\newlength{\sidefootheight}
\newcommand*{\setsidefootheight}[1]{%
  \setlength{\dimen\sidefootins}{#1}%
  \advance\dimen\sidefootins -\topskip
  \advance\dimen\sidefootins \ht\strutbox
  \setlength{\sidefootheight}{\dimen\sidefootins}}
\setsidefootheight{\textheight}
```

`\sidefoothsep` `\sidefootwidth` `\sidefootvsep` The horizontal space between the text block and side footnotes (`\sidefoothsep`), the width of the column of notes (`\sidefootwidth`), and the vertical space between consecutive side footnotes (`\sidefootvsep`).

```
\newlength{\sidefoothsep}
\newlength{\sidefootvsep}
\newlength{\sidefootwidth}
```

`\setsidefeet` The macro `\setsidefeet{`⟨*hsep*⟩`}{`⟨*width*⟩`}{`⟨*vsep*⟩`}{`⟨*adj*⟩`}{`⟨*font*⟩`}{`⟨*height*⟩`}` sets the specifications for the side footnotes. An '*' means 'use the current value'.

```
\newcommand*{\setsidefeet}[6]{%
  \nametest{#1}{*}\ifsamename\else
    \setlength{\sidefoothsep}{#1}\@memznegtest{\sidefoothsep}%
  \fi
  \nametest{#2}{*}\ifsamename\else
    \setlength{\sidefootwidth}{#2}\@memznegtest{\sidefootwidth}%
  \fi
  \nametest{#3}{*}\ifsamename\else
    \setlength{\sidefootvsep}{#3}\@memznegtest{\sidefootvsep}%
  \fi
```

```
  \nametest{#4}{*}\ifsamename\else
    \setlength{\sidefootadjust}{#4}%
  \fi
  \nametest{#5}{*}\ifsamename\else
    \def\sidefoottextfont{#5}%
  \fi
  \nametest{#6}{*}\ifsamename\else
    \setsidefootheight{#6}%
    \ifdim\dimen\sidefootins>\z@\else
      \@mememerror{\protect\sidefootheight\space is zero or negative}{\@ehd}%
    \fi
  \fi}
\setsidefeet{\marginparsep}{\marginparwidth}%
            {\onelineskip}{0pt}%
            {\normalfont\footnotesize}{\textheight}%

```

`\sidefootform` Set the sidefootnotes raggedy right

```
\newcommand*{\sidefootform}{\rightskip=\z@ \@plus 2em}

```

`\m@sideft@left` `\m@sideft@right` Macros placing the sidefootnotes at the left and the right respectively.

```
\newcommand*{\m@sideft@left}{%
  \@tempdimc \sidefootwidth
  \advance\@tempdimc\sidefoothsep
  \kern-\@tempdimc}
\newcommand*{\m@sideft@right}{%
  \@tempdimc \columnwidth% or \hsize
  \advance\@tempdimc\sidefoothsep
  \kern\@tempdimc}

```

`\m@mdownsf` A length used in the vertical positioning of sidefootnotes. (Perhaps one of the `\@tempdim` lengths could be used instead?)

```
\newlength{\m@mdownsf}

```

`\sidefootcontents` The essence of the sidefootnote task. This first positions the column to the left or the right and then tries to make the bottom of the column align with the bottom of the textblock.

```
\newcommand*{\sidefootcontents}{\hbox to \z@{%
  \m@mwhich@margin{\m@msidefoot@margin}%
  \ifmemtortm
    \m@sideft@right
  \else
    \m@sideft@left
  \fi
```

Now the fun part. The general idea is to measure the height of the insert's contents, subtract this from the specified height of the insert, and move the

contents down by that amount. The code below seems to work except when sidefootnotes spill over to the following page; also, like `\sidebar`, they are out of vertical alignment when on a `\chapter` page. I haven't found a way to automatically adjust for these, which is why the `\sidefootadjust` length is there to enable manual adjustment.

```
\vtop to 0pt{%    original
  \normalsize\normalfont\sidefoottextfont
  \vskip\topskip \vskip-\ht\strutbox
  \vskip\sidefootadjust%    use this for minor vertical adjustment
  \m@mdownsf=\dimen\sidefootins
  \advance\m@mdownsf-\ht\sidefootins
  \advance\m@mdownsf-\dp\sidefootins
```

Here's where the fiddling occurs, arrived at by a mixture of theory and experiment.

```
  \ifdim\m@mdownsf>\sidefootvsep
    \advance\m@mdownsf\sidefootvsep
    \advance\m@mdownsf 0.5\ht\strutbox
  \fi
  \vskip\m@mdownsf%     --- basically works
  \unvbox\sidefootins%
  \vss}%
\hss}}

```

`\m@mopsidefoot` If there are any sidefeet then add them to the output.

```
\newcommand*{\m@mopsidefoot}{%
  \ifvoid\sidefootins\else
    \setbox\@outputbox \vbox{%
      \sidefootcontents
      \unvbox\@outputbox}
  \fi}

```

`\mem@makecol` `\mem@makecolbf` `\@reinserts` `\@mem@extranofeet` `@mem@testifnofoot` Revise these to cater for the new sidefoot insert.

```
\gdef\mem@makecol{
  \m@m@makecolintro
  \ifvoid\footins
    \setbox\@outputbox \box\@cclv
  \else
    \m@mopfootnote
  \fi
  \m@mdoextrafeet
  \m@m@makecolfloats
  \m@mopsidebar
  \m@mopsidefoot
  \m@m@makecoltext
  \global \maxdepth \@maxdepth}

```

```
\gdef\mem@makecolbf{
  \m@m@makecolintro
  \setbox\@outputbox \box\@cclv
  \m@m@makecolfloats
  \ifvoid\footins
  \else
    \m@mopfootnotebf
  \fi
  \m@mdoextrafeet
  \m@mopsidebar
  \m@mopsidefoot
  \m@m@makecoltext
  \global \maxdepth \@maxdepth}

\gdef\@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \m@mdodoreinextrafeet
  \ifvbox\@kludgeins\else\insert\@kludgeins{\unvbox\@kludgeins}\fi
  \ifvoid\sideins\else\insert\sideins{\unvbox\sideins}\fi
  \ifvoid\sidefootins\else\insert\sidefootins{\unvbox\sidefootins}\fi}

```

The easy way of extending the `\clearpage` code is by:

```
\renewcommand*{\@mem@extranofeet}{%
 \ifvoid\sidefootins\else\@mem@nofootfalse\fi}
```

but as so many other changes have to be made it's probably better to leave that alone and change `\@mem@testifnofoot` directly.

```
\renewcommand*{\@mem@testifnofoot}{%
  \@mem@nofoottrue
  \ifvoid\footins\else\@mem@nofootfalse\fi
  \ifvoid\footinsv@r\else\@mem@nofootfalse\fi
  \ifvoid\sideins\else\@mem@nofootfalse\fi
  \ifvoid\sidefootins\else\@mem@nofootfalse\fi
  \@mem@extranofeet}

```

27.9.1 The sidefootnote macros

The macros here are essentially a copy of the ones for `\footnote`. I have not done anything about sidefootnotes in minipages, as I can't imagine a use for them while simultaneously making the final appearance looking good.
The usual counter.

```
\newcounter{sidefootnote}
  \renewcommand{\thesidefootnote}{\@arabic\c@sidefootnote}
\@addtoreset{sidefootnote}{chapter}

```

By using:
\letcountercounter{sidefootnote}{footnote}
the same counter will be used for both footnotes and sidefootnotes. This can be reverted to separate counters by;
\unletcounter{sidefootnote}

\sidefoottextfont The font for the sidefootnotes.

```
%\newcommand*{\sidefoottextfont}{\normalfont\footnotesize}
```

The remainder of the code is a copied from that for \footnote but with the appropriate variables for \sidefootnotes

\@sidempfn
\thesidempfn

```
\newcommand*{\@sidempfn}{sidefootnote}
\newcommand*{\thesidempfn}{\thesidefootnote}
```

\@makesidefnmark

```
\newcommand*{\@makesidefnmark}{\hbox{\@textsuperscript{%
  \normalfont\@thesidefnmark}}}
```

\@preamsidefntext

```
\newcommand{\@preamsidefntext}{%
  \interlinepenalty\interfootnotelinepenalty
  \floatingpenalty \@MM
  \splittopskip=\footnotesep
  \splitmaxdepth=\dp\strutbox
  \@parboxrestore}
```

\@sidefootnotetext

```
\newcommand{\@sidefootnotetext}[1]{\insert\sidefootins{%
  \hsize\sidefootwidth
  \@parboxrestore
    \def\baselinestretch{\m@m@singlespace}%
  \sidefootform \normalsize\normalfont\sidefoottextfont
  \splittopskip=\ht\strutbox
  \splitmaxdepth=\dp\strutbox
  \allowbreak
  \prevdepth=\dp\strutbox
  \vskip-\parskip
  \protected@edef\@currentlabel{%
    \csname p@sidefootnote\endcsname\@thesidefnmark}%
\color@begingroup
\@makesidefntext{{\sidefoottextfont #1}}%
\color@endgroup
  \ifvmode\else
```

```
    \unskip\@finalstrut\strutbox
  \fi
  \par
  \ifdim\prevdepth>\dp\strutbox \prevdepth=\dp\strutbox\fi
  \ifdim\prevdepth>99\p@
    \nobreak
    \vskip-\prevdepth
    \allowbreak
    \vskip\dp\strutbox
  \fi
  \vskip\sidefootvsep}%
  \m@mmf@prepare}

\@sidefootnotemark
\newcommand*{\@sidefootnotemark}{%
  \leavevmode
  \ifhmode
    \edef\@x@sf{\the\spacefactor}%
    \m@mmf@check
    \nobreak
  \fi
  \@makesidefnmark
  \m@mmf@prepare
  \ifhmode\spacefactor\@x@sf\fi
  \relax}

  \sidefootnote
\newcommand*{\sidefootnote}{\@ifnextchar[
  \@xsidefootnote{\stepcounter\@sidempfn
  \protected@xdef\@thesidefnmark{\thesidempfn}%
  \@sidefootnotemark\@sidefootnotetext}}

  \@xsidefootnote
\def\@xsidefootnote[#1]{%
  \begingroup
    \csname c@\@sidempfn\endcsname #1\relax
    \unrestored@protected@xdef\@thesidefnmark{\thesidempfn}%
  \endgroup
  \@sidefootnotemark\@sidefootnotetext}

\sidefootnotemark
\newcommand{\sidefootnotemark}{%
  \@ifnextchar[
    \@xsidefootnotemark
      {\stepcounter{sidefootnote}%
       \protected@xdef\@thesidefnmark{\thesidefootnote}%
       \@sidefootnotemark}}
```

\@xsidefootnotemark

```
\def\@xsidefootnotemark[#1]{%
  \begingroup
    \c@sidefootnote #1\relax
    \unrestored@protected@xdef\@thesidefnmark{\thesidefootnote}%
  \endgroup
  \@sidefootnotemark}
```

\sidefootnotetext

```
\newcommand*{\sidefootnotetext}{%
  \@ifnextchar[
    \@xsidefootnotetext
      {\protected@xdef\@thesidefnmark{\thesidempfn}%
       \@sidefootnotetext}}
```

\@xsidefootnotetext

```
\def\@xsidefootnotetext[#1]{%
  \begingroup
    \csname c@\@sidempfn\endcsname #1\relax
    \unrestored@protected@xdef\@thesidefnmark{\thesidempfn}%
  \endgroup
  \@sidefootnotetext}
```

\sidefootmarkstyle
\sidefootscript
\makesidefootmarkhook
\sidefootfootmark

```
\newcommand*{\sidefootmarkstyle}[1]{\def\sidefootscript##1{#1}}
\newcommand*{\makesidefootmarkhook}{}
\newcommand*{\sidefootfootmark}{%
  \ifdim\sidefootmarkwidth < \z@
    \llap{\hb@xt@ -\sidefootmarkwidth{%
          \hss\normalfont\sidefootscript{\@thesidefnmark}}%
         \hspace*{-\sidefootmarkwidth}}%
  \else
    \ifdim\sidefootmarkwidth = \z@
      {\normalfont\sidefootscript{\@thesidefnmark}}%
    \else
      \hb@xt@\sidefootmarkwidth{\hss\normalfont\sidefootscript{\@thesidefnmark}}%
    \fi
  \fi}
```

\makesidefootmark

```
\newcommand{\makesidefootmark}[1]{%
  \leavevmode
  \parindent \sidefootparindent\noindent
  \leftskip\sidefootmarksep\relax
  \advance\leftskip \sidefootmarkwidth \null\nobreak\hskip -\leftskip\relax
  \makesidefootmarkhook\relax
```

```
  \sidefootfootmark #1}

\@makesidefntext
\newcommand{\@makesidefntext}[1]{\makesidefootmark #1}
```

Final layout.

```
\sidefootmarkwidth
  \sidefootmarksep
\sidefootparindent
\newlength{\sidefootmarkwidth}
  \setlength{\sidefootmarkwidth}{0em}
\newlength{\sidefootmarksep}
  \setlength{\sidefootmarksep}{0em}
\newlength{\sidefootparindent}
  \setlength{\sidefootparindent}{1em}
\sidefootmarkstyle{\textsuperscript{#1}}
```

27.10 End notes

This is from the pagenote package, with minor amendments.

\ifm@mpnpageopt
\ifm@mpncontopt

We need two flags for the options. \m@mpnpageoptfalse means that page numbers are not available and \m@mpncontopt means that note numbers are reset at each \chapter.

```
\newif\ifm@mpnpageopt
  \m@mpnpageoptfalse
\newif\ifm@mpncontopt
  \m@mpncontoptfalse
```

\c@pagenote
\thepagenote

We need a counter for the notes. This is the default definition.

```
\newcounter{pagenote}[chapter]
\renewcommand{\thepagenote}{\arabic{pagenote}}
\setcounter{pagenote}{0}
```

\notepageref
\continuousnotenums

Let the user change the default options. \notepageref makes page numbers available (the package page option) and \continuousnotenums means that notes will be numbered continuously throughout the document(the package continuous option).

```
\newcommand*{\notepageref}{\m@mpnpageopttrue}
\@onlypreamble\notepageref
\newcommand*{\continuousnotenums}{%
  \counterwithout{pagenote}{chapter}
  \renewcommand{\thepagenote}{\arabic{pagenote}}}
\@onlypreamble\continuousnotenums
```

\ifmempagenotes Need to check if notes are required.

```
\newif\ifmempagenotes
  \mempagenotesfalse
```

\makepagenote This sets up the note file. At the end it emasculates itself so it can only be used once.

```
\newcommand*{\makepagenote}{%
  \newwrite\@notefile
  \immediate\openout\@notefile=\jobname.ent
  \mempagenotestrue
```

\pagenote Make sure that this has a useful definition.

```
  \def\pagenote{\@bsphack\begingroup
    \@sanitize
    \m@m@wrpnote}%

  \typeout{Writing note file \jobname.ent}%
  \let\makepagenote\@empty}
```

\immediate@protected@write We might have to do some immediate writes. This is an immediate version of the kernel \protected@write.

```
\newcommand{\immediate@protected@write}[3]{%
  \begingroup
  #2%
  \let\protect\@unexpandable@protect
  \edef\reserved@a{\immediate\write#1{#3}}%
  \reserved@a
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}
```

\m@m@pnwrite If the page option is used we cannot use an immediate write because the page number is only known in the output routine.

```
\let\m@m@pnwrite\immediate@protected@write
\AtBeginDocument{%
  \ifm@mpnpageopt
    \let\m@m@pnwrite\protected@write
  \fi}
```

\pnchap Redefine these for different subheadings in the notes list.

\pnschap

```
\newcommand*{\pnchap}{\f@rtoc}
\newcommand*{\pnschap}{\f@rbdy}
```

\m@m@wrpnote This writes the note information to the note file. If the optional argument is empty it increments the note counter and calls \notenumintext to handle its appearance in the body text.

```
\newcommand{\m@m@wrpnote}[2][]{%
  \@ifmtarg{#1}{\refstepcounter{pagenote}%
                \notenumintext{\thepagenote}}{}%
```

Check if this is the first note in a division, and if so indicate this in the file.

```
  \ifm@mpn@new@chap
    \global\m@mpn@new@chapfalse
    \addtonotes{\string\pagenotesubhead{\@chapapp}{\thechapter}{\pnchap}}%
  \fi
  \ifm@mpn@new@schap
    \global\m@mpn@new@schapfalse
    \addtonotes{\string\pagenotesubhead{\@chapapp}{}{\pnschap}}%
  \fi
```

Finally, write the entry.

```
  \m@m@pnwrite\@notefile{}
    {\string\noteentry{\thepagenote}{#1}{#2}{\thepage}}%
  \endgroup
  \@esphack}

```

\pagenote The user command to generate a note. It is given substance by \makepagenote.

```
\def\pagenote{\@bsphack\begingroup \@sanitize\m@m@pagenote}
```

\m@m@pagenote

```
\newcommand{\m@m@pagenote}[2][]{\endgroup\@esphack}

```

\pagetofootnote Let the user change pagenotes to footnotes, or vice-versa. In either case the
\foottopagenote optional argument is ignored.
\memsavefootnote
\memsavepagenote

```
\newcommand*{\pagetofootnote}{%
  \let\memsavepagenote\pagenote
  \renewcommand{\pagenote}[2][]{\footnote{##2}}}
\newcommand*{\foottopagenote}{%
  \let\memsavefootnote\footnote
  \renewcommand*{\footnote}[2][]{\pagenote{##2}}}

```

\addtonotes \addtonotes{⟨*text*⟩} puts ⟨*text*⟩ into the notes file.

```
\newcommand{\addtonotes}[1]{\ifmempagenotes
 \IfFileExists{\jobname.ent}{\m@m@pnwrite\@notefile{}{#1}}{\mempnofilewarn}%
\fi}

```

\notenumintext \notenumintext{⟨*notenum*⟩} typesets ⟨*notenum*⟩ (in the body text).
\notenuminnotes

```
\newcommand{\notenumintext}[1]{%
  \textsuperscript{#1}}
```

\notenuminnotes{⟨notenum⟩} typesets ⟨notenum⟩ (as part of the note).

```
\newcommand{\notenuminnotes}[1]{%
  {\normalfont #1.}\space}
```

\noteentry \noteentry{⟨notenum⟩}{⟨id⟩}{⟨pagenum⟩}{⟨text⟩} typesets a note.

```
\newcommand{\noteentry}[4]{%
  \prenoteinnotes
  \noteidinnotes{#1}{#2}\pageinnotes{#4}\noteinnotes{#3}%
  \postnoteinnotes}

```

\idtextinnotes \idtextinnotes{⟨id text⟩} typesets the note's ⟨id text⟩.

```
\newcommand{\idtextinnotes}[1]{%
  [#1]\space}
```

\noteidinnotes \noteidinnotes{⟨notenum⟩}{⟨id⟩} is used to typeset the note identification (in the note listing). It is set so that it typesets the ⟨id⟩ if it is not empty, otherwise it sets the ⟨notenum⟩.

```
\newcommand{\noteidinnotes}[2]{%
  \@ifmtarg{#2}{%
    \notenuminnotes{#1}}{\idtextinnotes{#2}}}
```

\pageinnotes \pageinnotes⟨pagenum⟩ calls \printpageinnotes{⟨pagenum⟩} to typeset the \printpageinnotes originating page number (in the note), but only if the \notepageref declaration has been used (the page number is not trustworthy unless the \notepageref has been used).

```
\newcommand{\pageinnotes}[1]{%
  \ifm@mpnpageopt \printpageinnotes{#1}\fi}
\newcommand*{\printpageinnotes}[1]{%
  (\pagerefname\ #1)\space}
```

\noteinnotes \noteinnotes{⟨text⟩} is used to typeset the note's text (in the note list).

```
\newcommand{\noteinnotes}[1]{#1}

```

\prenoteinnotes These are called immediately before and after the note information is typeset.
\postnoteinnotes

```
\newcommand{\prenoteinnotes}{\par\noindent}
\newcommand{\postnoteinnotes}{\par}

```

\notesname Heading for note list.
\notedivision

```
\providecommand*{\notesname}{Notes}
\newcommand*{\notedivision}{\chapter{\notesname}}

```

\printnotes User commands to print the note file.
\printpagenotes*

```
\newcommand*{\printpagenotes}{\@ifstar{\@sprintpagenotes}{\@printpagenotes}}
```

\mempnofilewarn Warning when the notes file does not exist.

```
\newcommand*{\mempnofilewarn}{%
  \ClassWarning{memoir}{There is no .ent file}}
```

\@sprintpagenotes Macro implementing \printpagenotes*.

```
\newcommand*{\@sprintpagenotes}{%
  \ifmempagenotes
  \notedivision
\IfFileExists{\jobname.ent}{%
  \immediate\closeout\@notefile
  \input{\jobname.ent}%
  \immediate\openout\@notefile=\jobname.ent%
  }{%
  \mempnofilewarn
}%
\fi}
```

\@printpagenotes Macro implementing \printpagenotes.

```
\newcommand*{\@printpagenotes}{%
  \ifmempagenotes
    \notedivision
    \IfFileExists{\jobname.ent}{%
      \immediate\closeout\@notefile
      \input{\jobname.ent}%
     }{%
       \mempnofilewarn
     }
  \fi}
```

\pagenotesubhead The section heading before each set of notes.
\pagenotesubhead{⟨*chaptername*⟩}{⟨*number*⟩}{⟨*title*⟩}

```
\newcommand*{\pagenotesubhead}[3]{%
  \section*{#1 #2 #3}}
```

28 Change marks

When preparing a manuscript it normally goes through several iterations. The commands provided may be used to identify changes made to a document during its life cycle.
The code for this part of the class is based on the version controls in the iso class [Wil00b].

28.1 Print control

Members of the development group often need to see the changes between document versions, while the general public does not.

\ifchangemarks This controls the appearence of the version controls defined below.

```
\newif\ifchangemarks\changemarksfalse
```

The marks only work properly when the draft option is in effect. Also, the command \changemarkstrue must be put in the document preamble.

\changemarks More user friendly version of \changemarks(true/false).
\nochangemarks

```
\newcommand*{\changemarks}{\changemarkstrue}
\newcommand*{\nochangemarks}{\changemarksfalse}
```

\v@rid This acts as an alias for \marginpar when both \ifchangemarks is true and the draft option is in effect, otherwise it throws away its two arguments.

```
\newcommand{\v@rid}[2]{%
  \@bsphack
  \ifchangemarks
     \ifdraftdoc
       \marginpar[#1]{#2}%
  \fi\fi
  \@esphack}
```

28.2 Change marking

The following commands flag changes in the typeset document. Each of the commands takes one parameter which is intended to be a 'change number' or comment for tracking purposes. A symbol and the ⟨*change-id*⟩ is put into the margin near where the command is given. The marking commands should be attached to some word or punctuation mark in the text otherwise extraneous spaces may creep into the final document.

\added \added{⟨*change-id*⟩} Flags, with the symbol ⊕, that something has been added to the manuscript.

```
\newcommand{\added}[1]{%
  \@bsphack
  \ifchangemarks
    \v@rid{\small$\oplus$ #1}{\small$\oplus$ #1}%
  \fi
  \@esphack}
```

\deleted \deleted{⟨*change-id*⟩} Flags, with the symbol ≠, that something has been deleted from the manuscript.

```
\newcommand{\deleted}[1]{%
  \@bsphack
```

```
  \ifchangemarks
    \v@rid{\small$\neq$ #1}{\small$\neq$ #1}%
  \fi
  \@esphack}
```

`\changed` `\changed{⟨change-id⟩}` Flags, with the symbol ⇔, that something has been changed in the text.

```
\newcommand{\changed}[1]{%
  \@bsphack
  \ifchangemarks
    \v@rid{\small$\Leftrightarrow$ #1}{\small$\Leftrightarrow$ #1}%
  \fi
  \@esphack}
```

29 Trimming marks

The showtrims options prints trimming marks at the corners of the logical page. The code for this comes from ideas gleaned from Martin Schröder's everyshi package [Sch98] and Melchior Franz's crop package [Fra00]. The implementation and any errors are mine.

The implementation up to October 2002 was limited to putting a cross at the page corners. The manual directed users to the crop packge if they wanted anything more. In October 2002 the implementation was substantially extended. The background to this is below.

Before the release of version 1.7 of crop in May 2002 its author asked me to make some changes to memoir that would be helpful to him, and I did so. Later he told me that the changes were unnecessary and I reverted to the original memoir code. On 2002/10/06 Peter Heslin (`peter.heslin@ucd.ie`) started a thread on `comp.text.tex` titled 'Incompatibility of memoir.cls and crop.sty' in which he said that memoir and crop did not seem to work together. The following are some snippets from that thread, identified by the various proponents.

crop Ah, yes. It cannot work with all those `\settrims` etc. crop.sty does only respect `\stock{width,height}`, but none of the other memoir specific trim marks stuff ...

crop ... I'm not a memoir expert. crop.sty respects all the usual LaTeX paper dimensions and simply puts the marks around the page. It works with all standard classes and the KOMA classes. memoir seems to do things completely differently ...

Sorry, but I'm afraid I have to pass the problem to the memoir author. ...

memoir I will try and take a look at the problem but I'm not a crop expert — I've never used it. However it seems odd to me that crop doesn't work with memoir. Memoir provides a different interface for specifying the page layout but then translates everything to the standard length variables. ...

crop You've just taken code from it `...:->`

> crop doesn't have its own idea at all. It just uses that of DEK and LaTeX, which says, that the reference point of the output box in the output routine is assumed 1 inch to the left [later corrected to right] and 1 inch down from the upper left (virtual) paper corner.

crop With other words: crop changes `\hoffset` and `\voffset`, in order to center the logical (virtual) page on the physical sheet of paper, in the middle of the crop marks. The length macros `\evensidemargin` and `\oddsidemargin` are unchanged and refer still to the logical page (modulo 1 inch).

The memoir class, in contrast (mis)uses the LaTeX margins `\even(odd)sidemargin` for that purpose.

Thus, the meaning of `\even(odd)sidemargin` is different in both packages.

Given all the above I concluded that `crop` would remain as it was, and that in order to satisfy Peter Heslin I would have to extend the trimming marks. The following was originally limited to extensions asked for by Peter Heslin.

`\showtrimsoff` `\showtrimson` Switch trimming marksoff and on. Requested by James Hunt on CTT *Are crop marks needed on every page?*, February 2006.

```
\newcommand*{\showtrimsoff}{\showtrimsfalse}
\newcommand*{\showtrimson}{\showtrimstrue}
```

`\trimmark` This is a cross in a zero sized picture for marking the corner of a logical page. Up to version 1.0 the macro used `\setlength` for adjusting the `\unitlength` but Henrik Holm[26] discovered that if the `calc` package is used then LaTeX complains about a missing number.

```
\newcommand*{\trimmark}{%
  \begin{picture}(0,0)
    \unitlength 1cm
    \thinlines
    \put(-2,0){\line(1,0){4}}
    \put(0,-2){\line(0,1){4}}
  \end{picture}}
```

`\Ltrimpictl` `\Ltrimpictr` `\Ltrimpicbl` `\Ltrimpicbr` 'L' shaped trim marks for four corners.

```
\newcommand*{\Ltrimpictl}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \thinlines
    \put(-2,0){\line(-1,0){18}}
    \put(0,2){\line(0,1){18}}
  \end{picture}}
```

[26] Message to CTT on 2002/01/04 (`h.holm@spray.no`)

```
\newcommand*{\Ltrimpictr}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \thinlines
    \put(2,0){\line(1,0){18}}
    \put(0,2){\line(0,1){18}}
  \end{picture}}
\newcommand*{\Ltrimpicbl}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \thinlines
    \put(-2,0){\line(-1,0){18}}
    \put(0,-2){\line(0,-1){18}}
  \end{picture}}
\newcommand*{\Ltrimpicbr}{%
  \begin{picture}(0,0)
    \unitlength 1mm
    \thinlines
    \put(2,0){\line(1,0){18}}
    \put(0,-2){\line(0,-1){18}}
  \end{picture}}

```

`\Ftrimpicbl` Frame the page.

```
\newcommand*{\Ftrimpicbl}{%
  \begin{picture}(0,0)
    \unitlength 1pt
    \thinlines
    \put(0,0){\framebox(\strip@pt\paperwidth,\strip@pt\paperheight){}}
  \end{picture}}

```

`\tmarktl` `\tmarktr` `\tmarkbl` `\tmarkbr` The trimming marks used for corner display.

```
\newcommand*{\tmarktl}{\trimmark}
\newcommand*{\tmarktr}{\trimmark}
\newcommand*{\tmarkbl}{\trimmark}
\newcommand*{\tmarkbr}{\trimmark}

```

`\tmarktm` `\tmarkml` `\tmarkmr` `\tmarkbm` The trimming marks used for mid-side display.

```
\newcommand*{\tmarktm}{%
  \begin{picture}(0,0)%
    \unitlength 1mm
    \thinlines
    \put(0,2){\line(0,1){10}}
  \end{picture}}
\newcommand*{\tmarkml}{%
  \begin{picture}(0,0)%
    \unitlength 1mm
```

```
    \thinlines
    \put(-2,0){\line(-1,0){10}}
  \end{picture}}
\newcommand*{\tmarkmr}{%
  \begin{picture}(0,0)%
    \unitlength 1mm
    \thinlines
    \put(2,0){\line(1,0){10}}
  \end{picture}}
\newcommand*{\tmarkbm}{%
  \begin{picture}(0,0)%
    \unitlength 1mm
    \thinlines
    \put(0,-12){\line(0,1){10}}
  \end{picture}}
```

`\trimXmarks` `\trimLmarks` `\trimFrame` `\trimNone` These are declarations for the different kinds of trimming marks.

```
\newcommand*{\trimXmarks}{%
  \let\tmarktl\trimmark
  \let\tmarktr\trimmark
  \let\tmarkbl\trimmark
  \let\tmarkbr\trimmark}
\newcommand*{\trimLmarks}{%
  \let\tmarktl\Ltrimpictl
  \let\tmarktr\Ltrimpictr
  \let\tmarkbl\Ltrimpicbl
  \let\tmarkbr\Ltrimpicbr}
\newcommand*{\trimFrame}{%
  \let\tmarktl\null
  \let\tmarktr\null
  \let\tmarkbl\Ftrimpicbl
  \let\tmarkbr\null}
\newcommand*{\trimNone}{%
  \let\tmarktl\relax
  \let\tmarktr\relax
  \let\tmarkbl\relax
  \let\tmarkbr\relax
  \let\tmarktm\relax
  \let\tmarkml\relax
  \let\tmarkmr\relax
  \let\tmarkbm\relax}
```

`\trimmarks` This positions four marks (`\trimmark`) at the corners of a logical page. It is basically a `\vbox` with zero height and width.
Bastiaan Niels Veelo reported (2003/07/11) odd things (e.g., some macros ignored) when changing trim marks. Turns out to be related to the use of `\protect`.

```
\newcommand*{\trimmarks}{%
  \vbox to \z@{\vskip-1in \vskip\trimtop % top of logical page
    \hb@xt@\z@{\hskip-1in
      \ifodd\c@page
        \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
      \else
        \if@twoside
          \hskip\trimedge  % left of logical page
        \else
          \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
        \fi
      \fi
      \vbox to \paperheight{%
        \let\protect\relax        % <- v1.4 addition
        \hb@xt@\paperwidth{\tmarktl\hfil\tmarktm\hfil\tmarktr}%
        \vfil
        \hb@xt@\paperwidth{\tmarkml\hfil\tmarkmr}%
        \vfil
        \hb@xt@\paperwidth{\tmarkbl\hfil\tmarkbm\hfil\tmarkbr}}%
    \hss}%
  \vss}}

```

29.1 Quark marks

William Adams (2006/08/28) supplied the following code to use trim marks along the style of Quark Xpress.

`\registrationColour` `\quarkmarks` Trim marks on the style of Quark Express.

```
\newcommand*{\registrationColour}[1]{#1}
\newcommand*{\quarkmarks}{%
\renewcommand*{\tmarktl}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(-36,0){\line(1,0){24}}
    \put(0,12){\line(0,1){24}}
    \put(3,27){\ttfamily\fontsize{8bp}{10bp}\selectfont\jobname\ \
      \today\ \ \printtime\ \ Page \thepage}
  \end{picture}}}
\renewcommand*{\tmarktm}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(-24,24){\line(1,0){48}}
    \put(0,12){\line(0,1){24}}
    \put(0,24){\oval(12,12)}
  \end{picture}}}
\renewcommand*{\tmarktr}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
```

```
    \put(12,0){\line(1,0){24}}
    \put(0,12){\line(0,1){24}}
  \end{picture}}}
\renewcommand*{\tmarkmr}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(12,0){\line(1,0){24}}
    \put(24,-24){\line(0,1){48}}
    \put(24,0){\oval(12,12)}
  \end{picture}}}
\renewcommand*{\tmarkbr}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(12,0){\line(1,0){24}}
    \put(0,-36){\line(0,1){24}}
  \end{picture}}}
\renewcommand*{\tmarkbm}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(-24,-24){\line(1,0){48}}
    \put(0,-36){\line(0,1){24}}
    \put(0,-24){\oval(12,12)}
  \end{picture}}}
\renewcommand*{\tmarkbl}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(-36,0){\line(1,0){24}}
    \put(0,-36){\line(0,1){24}}
  \end{picture}}}
\renewcommand*{\tmarkml}{\registrationColour{%
  \begin{picture}(0,0)
    \setlength{\unitlength}{1bp}\thicklines
    \put(-36,0){\line(1,0){24}}
    \put(-24,-24){\line(0,1){48}}
    \put(-24,0){\oval(12,12)}
  \end{picture}}}
\renewcommand*{\trimmarks}{%
%%  \special{papersize=\the\stockwidth,\the\stockheight}
  {%
  \vbox to \z@{\vskip-1in \vskip\trimtop % top of logical page
    \hb@xt@\z@{\hskip-1in
      \ifodd\c@page
        \hskip\stockwidth \hskip-\trimedge \hskip-\paperwidth
      \else
        \if@twoside
          \hskip\trimedge % left of logical page
        \else
          \hskip\stockwidth \hsip-\trimedge \hskip-\paperwidth
        \fi
      \fi
```

```
      \vbox to \paperheight{%
        \let\protect\relax %      <- v1.4 addition
        \hb@xt@\paperwidth{\tmarktl\hfil\tmarktm\hfil\tmarktr}%
        \vfil
        \hb@xt@\paperwidth{\tmarkml\hfil\tmarkmr}%
        \vfil
        \hb@xt@\paperwidth{\tmarkbl\hfil\tmarkbm\hfil\tmarkbr}}%
      \hss}%
    \vss}}%
}}
```

Any marks are put onto the pages by adding to the `\shipout` routine.

`\mem@oldshipout` Keep a copy of the current version of `\shipout` in `\mem@oldshipout`.

```
\let\mem@oldshipout\shipout
```

`\mem@shipi` `\mem@shipii` Effectively these will add the `\trimmarks` to the box holding the contents of the page. Note that any `\makeindex` must come *before* pagesel (and selectp) package.

```
\newcommand*{\mem@shipi}{%
  \ifvoid\@cclv\expandafter\aftergroup\fi\mem@shipii}
\newcommand*\mem@shipii{%
  \ifvoid\@cclv
    \mem@oldshipout\box\@cclv
  \else
    \ifshowtrims
```

Heiko Oberdiek responded to a problem reported by Rolf Niepraschk. The earlier implementation might cause the output box to be shifted. Heikos explanation (in response to the line `\mem@oldshipout\vbox{\trimmarks\ifvbox\@cclv\unvbox\else\box\fi\@cclv}`) :

```
In vertical mode (\mem@oldshipout\vbox{...}) TeX puts
interline skip between two boxes (\trimmarks and \@cclv),
here \linekskip (1pt) is put inbetween causing the
shift downwards.
```

Heiko also provided the fix used below. Actually Heikos fix also fixes a problem that might cause glue settings to disappear.

```
    \mem@oldshipout\vbox{%
        \trimmarks
        \nointerlineskip
        \box\@cclv
     }%
    \else
      \mem@oldshipout\box\@cclv
    \fi
  \fi}
```

\shipout Our new version of \shipout, which is only needed for the showtrims option. This adds \mem@shipi to the page box which then calls the original version of \shipout.

```
\ifshowtrims
  \renewcommand*{\shipout}{\afterassignment\mem@shipi\setbox\@cclv=}
\fi
```

30 Verbatims, boxes, and files

All the code in this section was added for version 1.2 of the class.

30.1 Modified version of the verbatim package

Much of this is from the verbatim package code [SRR01]. Unless indicated otherwise, the code and commentary is from that package.

30.1.1 Preliminaries

\every@verbatim
\afterevery@verbatim The hook (i.e., token register) \every@verbatim is initialized to ⟨*empty*⟩. PW added the \afterevery@verbatim hook.

```
\newtoks\every@verbatim
  \every@verbatim={}
\newtoks\afterevery@verbatim
  \afterevery@verbatim={}
```

\@makeother \@makeother takes as argument a character and changes its category code to 12 (other).

```
\def\@makeother#1{\catcode`#112\relax}
```

\@vobeyspaces The macro \@vobeyspaces causes spaces in the input to be printed as spaces in the output.

```
\begingroup
 \catcode`\ =\active%
 \def\x{\def\@vobeyspaces{\catcode`\ \active\let \@xobeysp}}
 \expandafter\endgroup\x
```

\@xobeysp The macro \@xobeysp produces exactly one space in the output, protected against breaking just before it. (\@M is an abbreviation for the number 10000.)

```
\def\@xobeysp{\leavevmode\penalty\@M\ }
```

\verbatim@line We use a newly defined token register called \verbatim@line that will be used as the character buffer.

```
\newtoks\verbatim@line
```

PW. I have extended the original verbatim package code to handle TABs within verbatims. Normally TeX replaces a TAB by either a single space or ignores it altogether. For this purpose I have bits of code from the moreverb package [Fai98] for handling TABs.
Code and commentary from moreverb.
We define a few auxiliary macros and counters for expanding tabs.

```
\newcount\tab@position
```

`\@xobeytab` `\@xobeytab` puts enough spaces in to get to the next nominal tab stop

```
\def\@xobeytab{%
  \loop
    \toks@\expandafter{\the\toks@\@xobeysp}%
    \advance\tab@position-1
  \ifnum\tab@position>0 \repeat
}
```

`\@vobeytabs` `\@vobeytabs` initialises use of `\@xobeytab`. Needs to be executed within a group, as mustn't be allowed to leak out into the wide world.

```
\begingroup
  \catcode`\^^I=\active
  \gdef\@vobeytabs{\catcode`\^^I\active\let^^I\@xobeytab}%
\endgroup
```

`\verbatim@tabexpand` `\verbatim@tabexpand{`⟨*body of line*⟩`}\@nil` processes every character of a line by tail recursion, counting the characters and juggling things when a tab is encountered.

```
\def\verbatim@tabexpand#1{%
  \ifx#1\@nil
    \the\toks@
    \expandafter\par
  \else
    \ifx#1\@xobeytab
      \@xobeytab
    \else
```

We can safely put `\@xobeysp` into the token register, since it does precisely what we need

```
      \toks@\expandafter{\the\toks@#1}%
      \advance\tab@position\m@ne
    \fi
    \ifnum\tab@position=0 \tab@position\tab@size \fi
    \expandafter\verbatim@tabexpand
  \fi
}
```

End of code and commentary from moreverb.
PW. Some macros for turning tabbing on and off.

\tabson \tabsoff \@maybeobeytabs

\tabson turns tabbing on, and \tabsoff turns it off. Default is no tabbing.

```
\newif\ift@bs
\newcommand{\tabson}[1][4]{%
  \ifnum\@ne > #1\relax
    \tabsoff
  \else
    \t@bstrue
    \def\tab@size{#1\relax}%
    \def\@maybeobeytabs{\@vobeytabs}%
  \fi
}
\newcommand{\tabsoff}{%
  \t@bsfalse
  \def\tab@size{\z@}%
  \def\@maybeobeytabs{}%
}
\tabsoff
```

\tabverbatim@processline

Process a line with TABs (extracted from moreverb).

```
\def\tabverbatim@processline{\tab@position\tab@size
  \toks@{}%
  \expandafter\verbatim@tabexpand\the\verbatim@line\@nil}
```

\notabverbatim@processline

Processes a line ignoring TABs (this is the original verbatim package definition of \verbatim@processline).

```
\def\notabverbatim@processline{\the\verbatim@line\par}
```

We are now back to the verbatim code.
The following four macros are defined globally in a way suitable for the verbatim and verbatim* environments.

\verbatim@startline \verbatim@addtoline \verbatim@processline

\verbatim@startline initializes processing of a line by emptying the character buffer (\verbatim@line).

```
\def\verbatim@startline{\verbatim@line{}}
```

\verbatim@addtoline adds the tokens in its argument to our buffer register \verbatim@line without expanding them.

```
\def\verbatim@addtoline#1{%
  \verbatim@line\expandafter{\the\verbatim@line#1}}
```

Processing a line inside a verbatim or verbatim* environment means printing it. Ending the line means that we have to begin a new paragraph. We use \par for this purpose. Note that \par is redefined in \@verbatim to force TEX into horizontal mode and to insert an empty box so that empty lines in the input do appear in the output. (PW changed next line from \def\verbatim@processline{\the\verbatim@line\par}

```
\def\verbatim@processline{\notabverbatim@processline}
```

\verbatim@finish As a default, \verbatim@finish processes the remaining characters. When this macro is called we are facing the following problem: when the \end{verbatim} command is encountered \verbatim@processline is called to process the characters preceding the command on the same line. If there are none, an empty line would be output if we did not check for this case.
If the line is empty \the\verbatim@line expands to nothing. To test this we use a trick similar to that on p. 376 of the TEXbook, but with $...|$ instead of the ! tokens. These $ tokens can never have the same category code as a $ token that might possibly appear in the token register \verbatim@line, as such a token will always have been read with category code 12 (other). Note that \ifcat expands the following tokens so that \the\verbatim@line is replaced by the accumulated characters

```
\def\verbatim@finish{\ifcat$\the\verbatim@line$\else
  \verbatim@processline\fi}
```

30.1.2 The verbatim and verbatim* environments

\verbatim@font We start by defining the macro \verbatim@font that is to select the font and to set font-dependent parameters. Then we go through \verbatim@nolig@list to avoid certain ligatures. \verbatim@nolig@list is a macro defined in the LaTeX 2ε kernel to expand to

```
\do\`\do\<\do\>\do\,\do\'\do\-
```

All the characters in this list can be part of a ligature in some font or other.
PW. This is the original version which I'm going to replace.

```
\def\verbatim@font{\normalfont\ttfamily
                   \hyphenchar\font\m@ne
                   \let\do\do@noligs
                   \verbatim@nolig@list}
```

Actually the kernel defines the macro \@noligs which just runs the last two lines of the \verbatim@font above. As other package may add stuff to \@noligs, we will use that instead.

\setverbatimfont User level handle for changing the font used for verbatim text.
\m@mverbfont
\verbatim@font

```
\newcommand{\setverbatimfont}[1]{\def\m@mverbfont{#1}}
\setverbatimfont{\normalfont\ttfamily}

\def\verbatim@font{\m@mverbfont
                   \hyphenchar\font\m@ne
                   \@noligs}
```

\@verbatim The macro \@verbatim sets up things properly. First of all, the tokens of the \every@verbatim hook are inserted. Then a trivlist environment is started

and its first \item command inserted. Each line of the verbatim or verbatim* environment will be treated as a separate paragraph.

```
\def\@verbatim{\the\every@verbatim
  \trivlist \item \relax
```

The following extra vertical space is for compatibility with the LaTeXkernel: otherwise, using the verbatim package changes the vertical spacing of a verbatim environment nested within a quote environment.

```
  \if@minipage\else\vskip\parskip\fi
```

The paragraph parameters are set appropriately: the penalty at the beginning of the environment, left and right margins, paragraph indentation, the glue to fill the last line, and the vertical space between paragraphs. The latter space has to be zero since we do not want to add extra space between lines.

```
  \@beginparpenalty \predisplaypenalty
%%%  \leftskip\@totalleftmargin\rightskip\z@
  \memRTLleftskip\@totalleftmargin\memRTLrightskip\z@
  \parindent\z@\parfillskip\@flushglue\parskip\z@
```

There's one point to make here: the list environment uses TeX's \parshape primitive to get a special indentation for the first line of the list. If the list begins with a verbatim environment this \parshape is still in effect. Therefore we have to reset this internal parameter explicitly. We could do this by assigning 0 to \parshape. However, there is a simpler way to achieve this: we simply tell TeX to start a new paragraph. As is explained on p. 103 of the TeXbook, this resets \parshape to zero.

```
  \@@par
```

We now ensure that \par has the correct definition, namely to force TeX into horizontal mode and to include an empty box. This is to ensure that empty lines do appear in the output. Afterwards, we insert the \interlinepenalty since TeX does not add a penalty between paragraphs (here: lines) by its own initiative. Otherwise a verbatim environment could be broken across pages even if a \samepage declaration were present.
However, in a top-aligned minipage, this will result in an extra empty line added at the top. Therefore, a slightly more complicated construct is necessary. One of the important things here is the inclusion of \leavevmode as the first macro in the first line, for example, a blank verbatim line is the first thing in a list item.

```
  \def\par{%
    \if@tempswa
      \leavevmode\null\@@par\penalty\interlinepenalty
    \else
      \@tempswatrue
      \ifhmode\@@par\penalty\interlinepenalty\fi
    \fi}%
```

But to avoid an error message when the environment doesn't contain any text, we redefine \@noitemerr which will in this case be called by \endtrivlist.

```
  \def\@noitemerr{\@warning{No verbatim text}}%
```

Now we call \obeylines to make the end of line character active,

```
\obeylines
```

change the category code of all special characters, to 12 (other).

```
\let\do\@makeother \dospecials
```

and switch to the font to be used.

```
\verbatim@font
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of \unpenalty!

```
\everypar \expandafter{\the\everypar \unpenalty}%
```

PW added next code at end of \@verbatim.

```
\wrapright\the\afterevery@verbatim}
```

\verbatim
\verbatim*

Now we define the toplevel macros. \verbatim is slightly changed: after setting up things properly it calls \verbatim@start. This is done inside a group, so that \verbatim can be used directly, without \begin.
PW. The following is the original code, but I want a version of verbatim that recognises TABs.

```
   \begin{macrocode}
\def\verbatim{%
   \begingroup\@verbatim \frenchspacing\@vobeyspaces
   \verbatim@start}
   \end{macrocode}
   \cs{verbatim*} is defined accordingly.
   \begin{macrocode}
\@namedef{verbatim*}{\begingroup\@verbatim\verbatim@start}
\def\endverbatim{\endtrivlist\endgroup}
\expandafter\let\csname endverbatim*\endcsname=\endverbatim
   \end{macrocode}
```

PW. My code for these is a modified version of the original verbatim code.

```
\def\verbatim{\begingroup
  \ift@bs
    \def\verbatim@processline{\tabverbatim@processline}%
  \fi
  \@verbatim \frenchspacing\@vobeyspaces\@maybeobeytabs\verbatim@start}
\@namedef{verbatim*}{\begingroup
  \ift@bs
    \def\verbatim@processline{\tabverbatim@processline}%
  \fi
  \@verbatim\@maybeobeytabs\verbatim@start}
```

\endverbatim
\endverbatim*

To end the verbatim and verbatim* environments it is only necessary to finish the trivlist environment started in \@verbatim and close the corresponding group, and handle[27] following (non-) paragraph, by using \@doendpe.

[27] Noted by Zarko Cucej (zarko.cucej@uni-mb.si).

```
\def\endverbatim{\endtrivlist\endgroup\@doendpe}
\@namelet{endverbatim*}\endverbatim
```

For abnormal \parskips the NTG class included the following, but I'm not sure if it is relevant here, but if it is just how it should be included.

```
From NTG, where it is a \cs{def}
\providecommand*{\verbatim}{%
  \topsep=-0.5\parskip
  \@verbatim
  \frenchspacing\@vobeyspaces \@xverbatim}
```

30.1.3 The comment environment

The \comment macro is similar to \verbatim*. However, we do not need to switch fonts or set special formatting parameters such as \parindent or \parskip. We need only set the category code of all special characters to 12 (other) and that of ^^M (the end of line character) to 13 (active). The latter is needed for macro parameter delimiter matching in the internal macros defined below. In contrast to the default definitions used by the \verbatim and \verbatim* macros, we define \verbatim@addtoline to throw away its argument and \verbatim@processline, \verbatim@startline, and \verbatim@finish to act as no-ops. Then we call \verbatim@. But the first thing we do is to call \@bsphack so that this environment has no influence whatsoever upon the spacing.
PW: This is the original code for the comment environment, which I'm going to change.

```
\def\comment{\@bsphack
             \let\do\@makeother\dospecials\catcode`\^^M\active
             \let\verbatim@startline\relax
             \let\verbatim@addtoline\@gobble
             \let\verbatim@processline\relax
             \let\verbatim@finish\relax
             \verbatim@}
```

\endcomment is very simple: it only calls \@esphack to take care of the spacing. The \end macro closes the group and therefore takes care of restoring everything we changed.

```
\let\endcomment=\@esphack
```

PW: The remainder of this section is my code.

\setupcomment \setupcomment does all the \lets in the original \comment code.

```
\newcommand{\setupcomment}{%
```

```
\let\do\@makeother\dospecials\catcode`\^^M\active
\let\verbatim@startline\relax
\let\verbatim@addtoline\@gobble
\let\verbatim@processline\relax
\let\verbatim@finish\relax}
```

The macros below do no checking to see if something has (not) been defined previously. It's 'user beware' time.

`\newcomment` `\newcomment{`⟨*name*⟩`}` creates a new comment environment called ⟨*name*⟩. This is a generalisation of the original comment code.

```
\newcommand{\newcomment}[1]{%
  \expandafter\def\csname #1\endcsname{\@bsphack\setupcomment\verbatim@}%
  \expandafter\let\csname end#1\endcsname=\@esphack}
```

`\commentsoff` `\commentsoff{`⟨*name*⟩`}` switches off the ⟨*name*⟩ comment environment by defining the relevent macros to do nothing.

```
\newcommand{\commentsoff}[1]{%
  \expandafter\def\csname #1\endcsname{}%
  \expandafter\def\csname end#1\endcsname{}}
```

`\commentson` `\commentson{`⟨*name*⟩`}` switches on the ⟨*name*⟩ comment environment. It has to do the same things as `\newcomment` does, so let `\newcomment` do the work.

```
\newcommand{\commentson}[1]{\newcomment{#1}}

```

We had better supply the `comment` environment, as promised.

```
\newcomment{comment}

```

PW: That is the end of my changes and extensions to the original `comment` environment code.

30.1.4 The main loop

Here comes the tricky part: During the definition of the macros we need to use the special characters \, {, and } not only with their normal category codes, but also with category code 12 (other). We achieve this by the following trick: first we tell TEX that \, {, and } are the lowercase versions of !, [, and]. Then we replace every occurrence of \, {, and } that should be read with category code 12 by !, [, and], respectively, and give the whole list of tokens to `\lowercase`, knowing that category codes are not altered by this primitive!
But first we have ensure that !, [, and] themselves have the correct category code! To allow special settings of these codes we hide their setting in the macro `\vrb@catcodes`. If it is already defined our new definition is skipped.

```
\@ifundefined{vrb@catcodes}%
  {\def\vrb@catcodes{%
     \catcode`\!12\catcode`\[12\catcode`\]12}}{}
```

This trick allows us to use this code for applications where other category codes are in effect.

We start a group to keep the category code changes local.

```
\begingroup
 \vrb@catcodes
 \lccode`\!=`\\ \lccode`\[=`\{ \lccode`\]=`\}
```

We also need the end-of-line character `^^M`, as an active character. If we were to simply write `\catcode`\^^M=\active` then we would get an unwanted active end of line character at the end of every line of the following macro definitions. Therefore we use the same trick as above: we write a tilde `~` instead of `^^M` and pretend that the latter is the lowercase variant of the former. Thus we have to ensure now that the tilde character has category code 13 (active).

```
 \catcode`\~=\active \lccode`\~=`\^^M
```

The use of the `\lowercase` primitive leads to one problem: the uppercase character 'C' needs to be used in the code below and its case must be preserved. So we add the command:

```
 \lccode`\C=`\C
```

Now we start the token list passed to `\lowercase`. We use the following little trick (proposed by Bernd Raichle): The very first token in the token list we give to `\lowercase` is the `\endgroup` primitive. This means that it is processed by TEX immediately after `\lowercase` has finished its operation, thus ending the group started by `\begingroup` above. This avoids the global definition of all macros.

```
 \lowercase{\endgroup
```

`\verbatim@start` The purpose of `\verbatim@start` is to check whether there are any characters on the same line as the `\begin{verbatim}` and to pretend that they were on a line by themselves. On the other hand, if there are no characters remaining on the current line we shall just find an end of line character. `\verbatim@start` performs its task by first grabbing the following character (its argument). This argument is then compared to an active `^^M`, the end of line character.

```
    \def\verbatim@start#1{%
      \verbatim@startline
      \if\noexpand#1\noexpand~%
```

If this is true we transfer control to `\verbatim@` to process the next line. We use `\next` as the macro which will continue the work.

```
        \let\next\verbatim@
```

Otherwise, we define `\next` to expand to a call to `\verbatim@` followed by the character just read so that it is reinserted into the text. This means that those characters remaining on this line are handled as if they formed a line by themselves.

```
      \else \def\next{\verbatim@#1}\fi
```

Finally we call `\next`.

```
      \next}%
```

\verbatim@ The three macros \verbatim@, \verbatim@@, and \verbatim@@@ form the "main loop" of the verbatim environment. The purpose of \verbatim@ is to read exactly one line of input. \verbatim@@ and \verbatim@@@ work together to find out whether the four characters \end (all with category code 12 (other)) occur in that line. If so, \verbatim@@@ will call \verbatim@test to check whether this \end is part of \end{verbatim} and will terminate the environment if this is the case. Otherwise we continue as if nothing had happened. So let's have a look at the definition of \verbatim@:

```
\def\verbatim@#1~{\verbatim@@#1!end\@nil}%
```

Note that the ! character will have been replaced by a \ with category code 12 (other) by the \lowercase primitive governing this code before the definition of this macro actually takes place. That means that it takes the line, puts \end (four character tokens) and \@nil (one control sequence token) as a delimiter behind it, and then calls \verbatim@@.

\verbatim@@ \verbatim@@ takes everything up to the next occurrence of the four characters \end as its argument.

```
\def\verbatim@@#1!end{%
```

That means: if they do not occur in the original line, then argument #1 is the whole input line, and \@nil is the next token to be processed. However, if the four characters \end are part of the original line, then #1 consists of the characters in front of \end, and the next token is the following character (always remember that the line was lengthened by five tokens). Whatever #1 may be, it is verbatim text, so #1 is added to the line currently built.

```
    \verbatim@addtoline{#1}%
```

The next token in the input stream is of special interest to us. Therefore \futurelet defines \next to be equal to it before calling \verbatim@@@.

```
    \futurelet\next\verbatim@@@}%
```

\verbatim@@@ \verbatim@@@ will now read the rest of the tokens on the current line, up to the final \@nil token.

```
\def\verbatim@@@#1\@nil{%
```

If the first of the above two cases occurred, i.e. no \end characters were on that line, #1 is empty and \next is equal to \@nil. This is easily checked.

```
    \ifx\next\@nil
```

If so, this was a simple line. We finish it by processing the line we accumulated so far. Then we prepare to read the next line.

```
      \verbatim@processline
      \verbatim@startline
      \let\next\verbatim@
```

Otherwise we have to check what follows these \end tokens.

```
    \else
```

Before we continue, it's a good idea to stop for a moment and remember where we are: We have just read the four character tokens `\end` and must now check whether the name of the environment (surrounded by braces) follows. To this end we define a macro called `\@tempa` that reads exactly one character and decides what to do next. This macro should do the following: skip spaces until it encounters either a left brace or the end of the line. But it is important to remember which characters are skipped. The `\end`⟨*optional spaces*⟩`{` characters may be part of the verbatim text, i.e. these characters must be printed.
Assume for example that the current line contains

```
␣␣␣␣␣␣\end␣{AVeryLongEnvironmentName}
```

As we shall soon see, the scanning mechanism implemented here will not find out that this is text to be printed until it has read the right brace. Therefore we need a way to accumulate the characters read so that we can reinsert them if necessary. The token register `\@temptokena` is used for this purpose.
Before we do this we have to get rid of the superfluous `\end` tokens at the end of the line. To this end we define a temporary macro whose argument is delimited by `\end\@nil` (four character tokens and one control sequence token) to be used below on the rest of the line, after appending a `\@nil` token to it. (Note that this token can never appear in `#1`.) We use the following definition of `\@tempa` to get the rest of the line (after the first `\end`).

```
\def\@tempa##1!end\@nil{##1}%
```

We mentioned already that we use token register `\@temptokena` to remember the characters we skip, in case we need them again. We initialize this with the `\end` we have thrown away in the call to `\@tempa`.

```
\@temptokena{!end}%
```

We shall now call `\verbatim@test` to process the characters remaining on the current line. But wait a moment: we cannot simply call this macro since we have already read the whole line. Therefore we have to first expand the macro `\@tempa` to insert them again after the `\verbatim@test` token. A `^^M` character is appended to denote the end of the line. (Remember that this character comes disguised as a tilde.)

```
\def\next{\expandafter\verbatim@test\@tempa#1\@nil^^M}%
```

That's almost all, but we still have to now call `\next` to do the work.

```
    \fi \next}%
```

`\verbatim@test` We define `\verbatim@test` to investigate every token in turn.

```
\def\verbatim@test#1{%
```

First of all we set `\next` equal to `\verbatim@test` in case this macro must call itself recursively in order to skip spaces.

```
  \let\next\verbatim@test
```

We have to distinguish four cases:

1. The next token is a `^^M`, i.e. we reached the end of the line. That means that nothing special was found. Note that we use `\if` for the following comparisons so that the category code of the characters is irrelevant.

```
\if\noexpand#1\noexpand~%
```

We add the characters accumulated in token register `\@temptokena` to the current line. Since `\verbatim@addtoline` does not expand its argument, we have to do the expansion at this point. Then we `\let \next` equal to `\verbatim@` to prepare to read the next line.

```
  \expandafter\verbatim@addtoline
    \expandafter{\the\@temptokena}%
  \verbatim@processline
  \verbatim@startline
  \let\next\verbatim@
```

2. A space character follows. This is allowed, so we add it to `\@temptokena` and continue.

```
\else \if\noexpand#1
  \@temptokena\expandafter{\the\@temptokena#1}%
```

3. An open brace follows. This is the most interesting case. We must now collect characters until we read the closing brace and check whether they form the environment name. This will be done by `\verbatim@testend`, so here we let `\next` equal this macro. Again we will process the rest of the line, character by character. The characters forming the name of the environment will be accumulated in `\@tempc`. We initialize this macro to expand to nothing.

```
\else \if\noexpand#1\noexpand[%
  \let\@tempc\@empty
  \let\next\verbatim@testend
```

Note that the `[` character will be a `{` when this macro is defined.

4. Any other character means that the `\end` was part of the verbatim text. Add the characters to the current line and prepare to call `\verbatim@` to process the rest of the line.

```
\else
  \expandafter\verbatim@addtoline
    \expandafter{\the\@temptokena}%
  \def\next{\verbatim@#1}%
\fi\fi\fi
```

The last thing this macro does is to call `\next` to continue processing.

```
  \next}%
```

\verbatim@testend

\verbatim@testend is called when \end⟨*optional spaces*⟩{ was seen. Its task is to scan everything up to the next } and to call \verbatim@@testend. If no } is found it must reinsert the characters it read and return to \verbatim@. The following definition is similar to that of \verbatim@test: it takes the next character and decides what to do.

```
\def\verbatim@testend#1{%
```

Again, we have four cases:

1. ^^M: As no } is found in the current line, add the characters to the buffer. To avoid a complicated construction for expanding \@temptokena and \@tempc we do it in two steps. Then we continue with \verbatim@ to process the next line.

```
  \if\noexpand#1\noexpand~%
    \expandafter\verbatim@addtoline
      \expandafter{\the\@temptokena[}%
    \expandafter\verbatim@addtoline
      \expandafter{\@tempc}%
    \verbatim@processline
    \verbatim@startline
    \let\next\verbatim@
```

2. }: Call \verbatim@@testend to check if this is the right environment name.

```
  \else\if\noexpand#1\noexpand]%
    \let\next\verbatim@@testend
```

3. \: This character must not occur in the name of an environment. Thus we stop collecting characters. In principle, the same argument would apply to other characters as well, e.g., {. However, \ is a special case, since it may be the first character of \end. This means that we have to look again for \end{⟨*environment name*⟩} Note that we prefixed the ! by a \noexpand primitive, to protect ourselves against it being an active character.

```
  \else\if\noexpand#1\noexpand!%
    \expandafter\verbatim@addtoline
      \expandafter{\the\@temptokena[}%
    \expandafter\verbatim@addtoline
      \expandafter{\@tempc}%
    \def\next{\verbatim@!}%
```

4. Any other character: collect it and continue. We cannot use \edef to define \@tempc since its replacement text might contain active character tokens.

```
  \else \expandafter\def\expandafter\@tempc\expandafter
    {\@tempc#1}\fi\fi\fi
```

As before, the macro ends by calling itself, to process the next character if appropriate.

```
\next}%
```

`\verbatim@@testend` Unlike the previous macros `\verbatim@@testend` is simple: it has only to check if the `\end{`⟨...⟩`}` matches the corresponding `\begin{`⟨...⟩`}`

```
\def\verbatim@@testend{%
```

We use `\next` again to define the things that are to be done. Remember that the name of the current environment is held in `\@currenvir`, the characters accumulated by `\verbatim@testend` are in `\@tempc`. So we simply compare these and prepare to execute `\end{`⟨*current environment*⟩`}` macro if they match. Before we do this we call `\verbatim@finish` to process the last line. We define `\next` via `\edef` so that `\@currenvir` is replaced by its expansion. Therefore we need `\noexpand` to inhibit the expansion of `\end` at this point.

```
  \ifx\@tempc\@currenvir
    \verbatim@finish
    \edef\next{\noexpand\end{\@currenvir}%
```

Without this trick the `\end` command would not be able to correctly check whether its argument matches the name of the current environment and you'd get an interesting LaTeX error message such as:

```
! \begin{verbatim*} ended by \end{verbatim*}.
```

But what do we do with the rest of the characters, those that remain on that line? We call `\verbatim@rescan` to take care of that. Its first argument is the name of the environment just ended, in case we need it again. `\verbatim@rescan` takes the list of characters to be reprocessed as its second argument. (This token list was inserted after the current macro by `\verbatim@@@`.) Since we are still in an `\edef` we protect it by means of`\noexpand`.

```
                \noexpand\verbatim@rescan{\@currenvir}}%
```

If the names do not match, we reinsert everything read up to now and prepare to call `\verbatim@` to process the rest of the line.

```
  \else
    \expandafter\verbatim@addtoline
      \expandafter{\the\@temptokena[}%
      \expandafter\verbatim@addtoline
        \expandafter{\@tempc]}%
    \let\next\verbatim@
  \fi
```

Finally we call `\next`.

```
  \next}%
```

`\verbatim@rescan` In principle `\verbatim@rescan` could be used to analyse the characters remaining after the `\end{...}` command and pretend that these were read "properly", assuming "standard" category codes are in force.[28] But this is not always possible (when there are unmatched curly braces in the rest of the line). Besides, we think that this is not worth the effort: After a `verbatim` or

[28] Remember that they were all read with category codes 11 (letter) and 12 (other) so that control sequences are not recognized as such.

verbatim* environment a new line in the output is begun anyway, and an \end{comment} can easily be put on a line by itself. So there is no reason why there should be any text here. For the benefit of the user who did put something there (a comment, perhaps) we simply issue a warning and drop them. The method of testing is explained in Appendix D, p. 376 of the TEXbook. We use ^^M instead of the ! character used there since this is a character that cannot appear in #1. The two \noexpand primitives are necessary to avoid expansion of active characters and macros.
One extra subtlety should be noted here: remember that the token list we are currently building will first be processed by the \lowercase primitive before TEX carries out the definitions. This means that the 'C' character in the argument to the \@warning macro must be protected against being changed to 'c'. That's the reason why we added the \lccode`\C=`\C assignment above. We can now finish the argument to \lowercase as well as the group in which the category codes were changed.

```
\def\verbatim@rescan#1#2~{\if\noexpand~\noexpand#2~\else
    \@warning{Characters dropped after `\string\end{#1}'}\fi}}
```

30.1.5 The \verbatiminput command

\verbatim@in@stream We begin by allocating an input stream (out of the 16 available input streams).

```
\newread\verbatim@in@stream
```

\verbatim@readfile The macro \verbatim@readfile encloses the main loop by calls to the macros \verbatim@startline and \verbatim@finish, respectively. This makes sure that the user can initialize and finish the command when the file is empty or doesn't exist. The verbatim environment has a similar behaviour when called with an empty text.

```
\def\verbatim@readfile#1{%
  \verbatim@startline
```

When the file is not found we issue a warning.

```
  \openin\verbatim@in@stream #1\relax
  \ifeof\verbatim@in@stream
    \typeout{No file #1.}%
  \else
```

At this point we pass the name of the file to \@addtofilelist so that its appears appears in the output of a \listfiles command. In addition, we use \ProvidesFile to make a log entry in the transcript file and to distinguish files read in via \verbatiminput from others.

```
    \@addtofilelist{#1}%
    \ProvidesFile{#1}[(verbatim)]%
```

While reading from the file it is useful to switch off the recognition of the end-of-line character. This saves us stripping off spaces from the contents of the line.

```
    \expandafter\endlinechar\expandafter\m@ne
```

```
    \expandafter\verbatim@read@file
    \expandafter\endlinechar\the\endlinechar\relax
    \closein\verbatim@in@stream
  \fi
  \verbatim@finish
}
```

`\verbatim@read@file` All the work is done in `\verbatim@read@file`. It reads the input file line by line and recursively calls itself until the end of the file.

```
\def\verbatim@read@file{%
  \read\verbatim@in@stream to\next
  \ifeof\verbatim@in@stream
  \else
```

For each line we call `\verbatim@addtoline` with the contents of the line. `\verbatim@processline` is called next.

```
    \expandafter\verbatim@addtoline\expandafter{\next}%
    \verbatim@processline
```

After processing the line we call `\verbatim@startline` to initialize all before we read the next line.

```
    \verbatim@startline
```

Without `\expandafter` each call of `\verbatim@read@file` uses space in TEX's input stack.[29]

```
    \expandafter\verbatim@read@file
  \fi
}
```

`\verbatiminput` `\verbatiminput` first starts a group to keep font and category changes local. Then it calls the macro `\verbatim@input` with additional arguments, depending on whether an asterisk follows.
PW. I added the TAB checking code.

```
\def\verbatiminput{\begingroup
  \ift@bs
    \def\verbatim@processline{\tabverbatim@processline}%
  \fi
  \@ifstar{\verbatim@input{\@maybeobeytabs}}%
          {\verbatim@input{\frenchspacing\@vobeyspaces\@maybeobeytabs}}}
```

`\verbatim@input` `\verbatim@input` first checks whether the file exists, using the standard macro csIfFileExists which leaves the name of the file found in `\@filef@und`. Then everything is set up as in the `\verbatim` macro.

```
\def\verbatim@input#1#2{%
  \IfFileExists {#2}{\@verbatim #1\relax
```

[29] A standard TEX would report an overflow error if you try to read a file with more than ca. 200 lines. The same error occurs if the first line of code in §390 of *"TeX: The Program"* is missing.

Then it reads in the file, finishes off the `trivlist` environment started by `\@verbatim` and closes the group. This restores everything to its normal settings.

```
    \verbatim@readfile{\@filef@und}\endtrivlist\endgroup\@doendpe}%
```

If the file is not found a more or less helpful message is printed. The final `\endgroup` is needed to close the group started in `\verbatiminput` above.

```
  {\typeout {No file #2.}\endgroup}}
```

That completes my borrowings from verbatim.
The next bunch of code implements wrapping verbatim lines so they, hopefully, stay within the typeblock.

`\verbatimindent` `\verbatimbreakchar` `\setverbatimbreak`

The length `\verbatimindent` is the distance continuation lines are indented from the left margin. `\verbatimbreakchar` is the character to indicate a wrapped line.

```
\newlength{\verbatimindent}
  \setlength{\verbatimindent}{3em}
\newcommand*{\verbatimbreakchar}{\char`\%}
\newcommand*{\setverbatimbreak}{%
  \vspace*{-\baselineskip}%
  \def\@xobeysp{~\discretionary{\verbatimbreakchar}%
    {\kern\verbatimindent}{}}%
}

```

`\raggedwrap` `\wrappingon` `\wrapright` `\wrappingoff`

`\wrappingon` and `\wrappingoff` enable/prohibit wrapping. The default is `\wrappingoff`.
The macro `\wrapright` is used to set paragraph skips; without raggedright the lines may break at the first space *outside* the text area. However, Paul (`paulaugust2003@yahoo.com`) found that wrapped verbatims in a list (e.g., `itemize`) were not indented although regular verbatims were indented. `\raggedwrap` is a variation on `\raggedright` (`\leftskip` is set to `\@totalleftmargin` instead of 0pt), and seems to have fixed the problem with the original code which used `\raggedright`.

```
\newcommand*{\raggedwrap}{%
  \@rightskip\@flushglue
%%%  \rightskip\@rightskip
  \memRTLrightskip\@rightskip
%%%  \leftskip\@totalleftmargin
  \memRTLleftskip\@totalleftmargin
  \parindent\ragrparindent}
\newcommand*{\wrappingon}{%
  \def\@xobeysp{~\discretionary{\verbatimbreakchar}%
    {\kern\verbatimindent}{}}%
  \def\wrapright{\raggedwrap}}
\newcommand*{\wrappingoff}{%
  \def\@xobeysp{\leavevmode\penalty\@M\ }%
  \def\wrapright{}}
\wrappingoff
```

30.2 Writing and boxing verbatim

This bunch of code is from the moreverb package.

verbatimoutput `\begin{verbatimoutput}{⟨filename⟩}` writes all text in its body to a file, the name of which it is given as an argument. (The code was written by Rainer Schöpf but the environment was called `verbatimwrite`). In the code below, uncommenting
`\catcode‘\^^I=12`
will result in TABs being written as `^^I`.

```
\newwrite \verbatim@out
\def\verbatimoutput#1{%
  \@bsphack
  \immediate\openout \verbatim@out #1
  \let\do\@makeother\dospecials
  \catcode‘\^^M\active %% \catcode‘\^^I=12
  \def\verbatim@processline{%
    \immediate\write\verbatim@out
      {\the\verbatim@line}}%
  \verbatim@start}

\def\endverbatimoutput{%
  \immediate\closeout\verbatim@out
  \@esphack}
```

fboxverbatim `fboxverbatim` puts the contents of a verbatim environment in a framing box. (PW: This was originally called `boxedverbatim`).
(Written by Victor Eijkhout.)
Bug fix (supplied by David Carlisle) 1995/12/28, marked `%%%DPC%%%`
First, redefine ‘processline’ to produce only a line as wide as the natural width of the line

```
\def\fboxverbatim{\begingroup%
  \tabsoff %% PW otherwise box fills the width
  \def\verbatim@processline{%
    {\setbox0=\hbox{\the\verbatim@line}%
    \hsize=\wd0 \the\verbatim@line\par}}%
```

Now save the verbatim code in a box

```
  \@minipagetrue%%%DPC%%%
  \@tempswatrue%%%DPC%%%
  \setbox0=\vbox\bgroup \verbatim
}
```

At the end of the environment, we (umm) simply have to stick the results into a frame.

```
\def\endfboxverbatim{%
  \endverbatim
  \unskip\setbox0=\lastbox %%%DPC%%%
```

Now everything's in the box, so we can close it...

```
\egroup
```

To change the code for centering, the next line needs a spot of hacking.

```
\fbox{\box0}% <<<=== change here for centering,...
\endgroup}
```

30.3 The shortvrb package

The following is the shortvrb package code (from doc.dtx by Frank Mittelbach). It has been so useful to me that I wanted to include it in the class.
CODE AND COMMENTARY IS BY FRANK MITTELBACH

\MakeShortVerb

```
\def\MakeShortVerb#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \@shortvrbinfo{Made }{#1}%
    \add@special{#1}%
```

Then the character's current catcode is stored in \cc\⟨c⟩.

```
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
```

The character is spliced into the definition using the same trick as used in \verb (for instance), having activated ~ in a group.

```
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
```

The character's old meaning is recorded in \ac\⟨c⟩ prior to assigning it a new one.

```
        \global\expandafter\let
           \csname ac\string#1\endcsname~%
        \gdef~{\verb~}}%
    \endgroup
```

Finally the character is made active.

```
    \global\catcode`#1\active
```

If we suspect that ⟨c⟩ is already a short reference, we tell the user. Now he or she is responsible if anything goes wrong...

```
  \else
    \@shortvrbinfo\@empty{#1 already}%
  \fi}
```

\DeleteShortVerb Here's the means of undoing a \MakeShortVerb, for instance in a region where you need to use the character outside a verbatim environment. It arranges for \dospecials and \@sanitize to be altered appropriately, restores the saved catcode and, if necessary, the character's meaning (as stored by

\MakeShortVerb). If the catcode wasn't stored in \cc\⟨c⟩ (by \MakeShortVerb) the command is silently ignored.

```
\def\DeleteShortVerb#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
  \else
    \@shortvrbinfo{Deleted }{#1 as}%
    \rem@special{#1}%
    \global\catcode`#1\csname cc\string#1\endcsname
```

We must not forget to reset \cc\⟨c⟩, otherwise the check in \MakeShortVerb for a repeated definition will not work.

```
    \global \expandafter\let \csname cc\string#1\endcsname \relax
    \ifnum\catcode`#1=\active
      \begingroup
        \catcode`\~\active   \lccode`\~`#1%
        \lowercase{%
          \global\expandafter\let\expandafter~%
          \csname ac\string#1\endcsname}%
      \endgroup \fi \fi}
```

\@shortvrbinfo Helper function for info messages.

```
\def\@shortvrbinfo#1#2{%
  \ClassInfo{memoir}{%
     #1\expandafter\@gobble\string#2 a short reference
                                          for \string\verb}}
```

\add@special This helper macro adds its argument to the \dospecials macro which is conventionally used by verbatim macros to alter the catcodes of the currently active characters. We need to add \do\⟨c⟩ to the expansion of \dospecials after removing the character if it was already there to avoid multiple copies building up should \MakeShortVerb not be balanced by \DeleteShortVerb (in case anything that uses \dospecials cares about repetitions).

```
\def\add@special#1{%
  \rem@special{#1}%
  \expandafter\gdef\expandafter\dospecials\expandafter
    {\dospecials \do #1}%
```

Similarly we have to add \@makeother\⟨c⟩ to \@sanitize (which is used in things like \index to re-catcode all special characters except braces).

```
  \expandafter\gdef\expandafter\@sanitize\expandafter
    {\@sanitize \@makeother #1}}
```

\rem@special The inverse of csadd@special is slightly trickier. \do is re-defined to expand to nothing if its argument is the character of interest, otherwise to expand simply to the argument. We can then re-define \dospecials to be the expansion of itself. The space after =`##1 prevents an expansion to \relax!

```
\def\rem@special#1{%
  \def\do##1{%
```

```
    \ifnum`#1=`##1 \else \noexpand\do\noexpand##1\fi}%
  \xdef\dospecials{\dospecials}%
```

Fixing `\@sanitize` is the same except that we need to re-define `\@makeother` which obviously needs to be done in a group.

```
  \begingroup
    \def\@makeother##1{%
      \ifnum`#1=`##1 \else \noexpand\@makeother\noexpand##1\fi}%
    \xdef\@sanitize{\@sanitize}%
  \endgroup}
```

END OF MITTELBACH CODE AND COMMENTARY.

30.4 General verbatim boxing and line numbering

A while ago I wrote a package that I never released. Here it is now, updated and improved. The package was based on code originally posted to CTT by Donald Arseneau on 13 July 2000.
This is DA's posted code.

```
\RequirePackage{verbatim}
\def\boxverbflag{14 }

\def\boxedverbatim{%
   \fboxsep=1em
   \def\verbatim@processline{\leavevmode
     \vrule\vbox{\advance\hsize-.8p@ \@@line
       {\strut\kern\fboxsep\the\verbatim@line\hss}%
        \kern\fbozsep}\vrule\par}%
   \@verbatim % but make some replacement settings
%   \ifdim\@totalleftmargin>\fboxsep \fboxsep\@totalleftmargin \fi
   \leftskip\x@skip \rightskip\z@skip
   \interlinepenalty\boxverbflag
   \parfillskip\z@ plus\p@ minus\p@
   \lineskip-\fboxsep \baselineskip\z@skip
   \frenchspacing\@vobeyspaces
   \boxverb@toprule
   \verbatim@start}

\def\endboxedverbatim{\hrule\endtrivlist}

\@namefdef{boxedverbatim*}{\let\frenchspacing\@gobble \boxedverbatim}
\@namefdef{endboxedverbatim*}{\hrule\endtrivlist}

\output=\expandafter{\expandafter\boxverb@split \the\output}

\def\boxverb@toprule{\hrule \nobreak \vskip-.1\p@
   \@@line{\vrule height2\fboxsep \hss \vrule}}

\def\boxverb@split{\ifnum\outputpenalty=\boxverbflag
```

```
\ifdim\dp\@cclv=\z@
\setbox\@cclv\vbox{\unvbox\@cclv\hrule\kern-.4pt}%
\null \kern-.7\topskip \boxverb@toprule
\fi\fi}
```

That's the end of DA's posted code
I have extended this to provide:

- A (multipage) boxed verbatim
- A (multipage) boxed verbatim input
- Normally each 'page' is boxed, but start/end rules can be switched off at page boundaries
- A heading can be put at the start of 'continuation' pages
- Lines of verbatim text can be numbered
- Can be 'downgraded' to normal verbatim environment

OK, on to the main code.

\boxverbflag
\bvboxsep

```
\def\boxverbflag{14 }
\newlength{\bvboxsep}      % user can change this
\setlength{\bvboxsep}{1em}

\newif\ifbvperpage % start/end lines on every page of multipage verbatim
\bvperpagetrue
```

\bvtopofpage
\b@vtop

Can use \bvtopofpage to put a heading above continued verbatims. For example
\bvtopofpage{\begin{center}\normalfont (Continued)\end{center}}
It only works for \bvperpagetrue.

```
\newcommand{\bvtopofpage}[1]{%
  \long\def\b@vtop{#1}}
\def\b@vtop{}  % used in \boxverb@split for heading
```

\c@memfbvline
\c@bvlinectr

Counter for adjusting the starting line number for boxed verbatims and for the line number itself..

```
\newcounter{memfbvline}
  \c@memfbvline=\z@
\newcounter{bvlinectr}
```

\setbvlinenums `\setbvlinenums{⟨firstline⟩}{⟨startnumsat⟩}` sets the first line number to ⟨*firstline*⟩ and the first line number to be printed is ⟨*startnumsat*⟩.

```
\newcommand*{\setbvlinenums}[2]{%
  \c@bvlinectr #1\relax \advance\c@bvlinectr \m@ne
  \ifnum\z@<\linemodnum%   we are printing line numbers
    \@tempcnta #2\relax
    \divide\@tempcnta\linemodnum
    \multiply\@tempcnta\linemodnum
    \c@memfbvline #2\relax
    \advance\c@memfbvline-\@tempcnta
  \fi}

```

\theb@vlinenumber \resetbvlinenumber \resetbvlinenumber reinitializes the line numbering.

```
\def\theb@vlinenumber{\getthelinenumber{bvlinectr}{memfbvline}}
\newcommand*{\resetbvlinenumber}{\setcounter{bvlinectr}{0}}

```

\b@vdocount \bvnumlength Increment line number if counting. A line number is printed in a space width \bvnumlength, which is given a temporary value here.

```
\def\b@vdocount{\ifbvcountlines\stepcounter{bvlinectr}\fi}
\newlength{\bvnumlength}
%% \settowidth{\bvnumlength}{\vlvnumfont 9999}
\settowidth{\bvnumlength}{\normalfont 999}

```

\ifbvcountinside \bvnumbersinside \bvnumbersoutside Flag and commands for positioning the numbers. Default is to print them inside the box.

```
\newif\ifbvcountinside  % TRUE if line numbers inside box
  \bvcountinsidetrue
\newcommand*{\bvnumbersinside}{\bvcountinsidetrue}
\newcommand*{\bvnumbersoutside}{\bvcountinsidefalse}

```

\b@vdoinside \b@vdooutside Print numbers inside (outside) the box.

```
\def\b@vdoinside{%
  \ifbvcountlines\ifbvcountinside%
    \makebox[\bvnumlength][r]{%
      \vlvnumfont \theb@vlinenumber\space}%
  \fi\fi}

\def\b@vdooutside{%
  \ifbvcountlines\ifbvcountinside\else%
    \llap{\makebox[\bvnumlength][r]{%
      \vlvnumfont \theb@vlinenumber\space}}%
  \fi\fi}

```

\@@m@mline \@@line is defined as \def\@@line{\hb@xt@\hsize} and I need an equivalent for \linewidth instead of \hsize for use in \setupboxverb@line.

```
\newcommand*{\@@m@mline}{\hb@xt@\linewidth}
```

\setupboxverb@line Use a macro for the first main part of DA's code for \boxedverbatim, adding lots of hooks.
Per Starbäck had problems with the combination of adjustwidth with boxedverbatim (see CTT *Re: [memoir] adjustwidth + boxedverbatim*, 2007/02/10) and Lars Madsen came up with a suggestion to use \linewidth instead of \hsize.

```
\newcommand{\setupboxverb@line}{%
  \par
  \ifbvperpage
    \output=\expandafter{\expandafter\boxverb@split \the\output}
  \fi
```

\verbatim@processline

```
  \def\verbatim@processline{\leavevmode
    \b@vdocount%
    \bvleftsidehook\vbox{\advance% \hsize-.8\p@ \@@line % changed to \linewidth
                                    \linewidth-.8\p@ \@@line
      {\b@vdooutside\strut\kern\bvboxsep%
       \b@vdoinside%
       \ift@bs
         \tabverbatim@processline
       \else
         \the\verbatim@line
       \fi
       \hss}%
     \kern\bvboxsep}\bvrightsidehook\par}}
```

\setupbox@verb Use a macro for the second main part of DA's code, and integrate it with the other verbatim codes (e.g., include \@maybeobeytags).

```
\newcommand{\setupbox@verb}{%
  \leftskip\z@skip \rightskip\z@skip
  \interlinepenalty\boxverbflag
  \parfillskip\z@ plus\p@ minus\p@
  \lineskip-\bvboxsep \baselineskip\z@skip
  \frenchspacing\@vobeyspaces\@maybeobeytabs
  \boxverb@toprule}
```

\boxedverbatim Given the two macros above, we can write a briefer version of DA's
\endboxedverbatim \boxedverbatim(*). As noted by Zarko Cucej[30], have to handle a following
\boxedverbatim* (no-) paragraph.
\endboxedverbatim*

[30] zarko.cucej@uni-mb.si

```
\def\boxedverbatim{\begingroup
  \let\@@line\@@m@mline%    new from mempatch v4.9
  \setupboxverb@line
  \@verbatim
  \setupbox@verb
  \verbatim@start}
\def\endboxedverbatim{\bvendrulehook\endtrivlist\endgroup\@doendpe}
\@namedef{boxedverbatim*}{\let\frenchspacing\@gobble \boxedverbatim}
\@namelet{endboxedverbatim*}\endboxverbatim

\def\boxverb@toprule{\bvtoprulehook
  \@@line{\bvleftsidehook \bvtopmidhook \bvrightsidehook}}
```

`\bvendofpage` `\bvendofpage{`⟨*spec*⟩`}` sets the boxed verbatim 'end of page' to ⟨*spec*⟩.

```
\newcommand*{\bvendofpage}[1]{%
  \def\boxverb@botpage{#1}}
%%%%\bvendofpage{\hrule\kern-.4pt}
\bvendofpage{\hrule width\linewidth\kern-.4pt}

\boxverb@split
\def\boxverb@split{\ifnum\outputpenalty=\boxverbflag
  \ifdim\dp\@cclv=\z@
%%%%    \setbox\@cclv\vbox{\unvbox\@cclv\hrule\kern-.4pt}%
    \setbox\@cclv\vbox{\unvbox\@cclv\boxverb@botpage}%
    \null \kern-.7\topskip \b@vtop \boxverb@toprule
  \fi
\fi}
```

`\bvtoprulehook` `\bvendrulehook` `\bvleftsidehook` `\bvrightsidehook` `\bvtopmidhook` The new hooks, for the top, bottom, left and right of the box.

```
\def\bvtoprulehook{\hrule width\linewidth \nobreak\vskip-.1\p@}
\def\bvendrulehook{\hrule width\linewidth}
\def\bvleftsidehook{\vrule}
\def\bvrightsidehook{\vrule}
\def\bvtopmidhook{\rule{0\p@}{2\bvboxsep} \hss}
```

`\boxedverbatiminput` `\boxedverbatiminput{`⟨*filename*⟩`}` read in filename contents as verbatim

`\boxedverbatim@input`

```
\newcommand{\boxedverbatiminput}{\begingroup
  \@ifstar{\let\frenchspacing\@gobble
           \boxedverbatim@input\relax}%
          {\boxedverbatim@input{\frenchspacing\@vobeyspaces}}}

\def\boxedverbatim@input#1#2{%
  \setupboxverb@line
  \IfFileExists{#2}{\@verbatim #1\relax
```

```
  \setupbox@verb
    \verbatim@readfile{\@filef@und}%
      \bvendrulehook\endtrivlist\endgroup\@doendpe}%
  {\typeout {No file #2.}\endgroup}}

```

Some prepackaged boxing styles.

`\bvbox` Original (default) boxing.

```
\newcommand{\bvbox}{%
  \bvperpagetrue%
  \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1\p@}%
  \renewcommand{\bvleftsidehook}{\vrule}%
  \renewcommand{\bvrightsidehook}{\vrule}%
  \renewcommand{\bvendrulehook}{\hrule}}

```

`\nobvbox` No boxing

```
\newcommand{\nobvbox}{%
  \bvperpagefalse%
  \renewcommand{\bvtoprulehook}{}%
  \renewcommand{\bvleftsidehook}{}%
  \renewcommand{\bvrightsidehook}{}%
  \renewcommand{\bvendrulehook}{}}

```

`\bvtopandtail` Head and foot horizontal lines only

```
\newcommand{\bvtopandtail}{%
  \bvperpagefalse%
  \renewcommand{\bvtoprulehook}{\hrule \nobreak \vskip-.1\p@}%
  \renewcommand{\bvleftsidehook}{}%
  \renewcommand{\bvrightsidehook}{}%
  \renewcommand{\bvendrulehook}{\hrule}}

```

`\bvsides` Side vertical lines only

```
\newcommand{\bvsides}{%
  \bvperpagefalse%
  \renewcommand{\bvtoprulehook}{\vskip 3ex}%
  \renewcommand{\bvleftsidehook}{\vrule}%
  \renewcommand{\bvrightsidehook}{\vrule}%
  \renewcommand{\bvendrulehook}{}}

```

30.5 The framed package

The following code is the framed package [Ars03] by Donald Arseneau.

```
   \framed
\endframed11782 \let\framed\relax \let\endframed\relax
   \shaded11783 \let\shaded\relax \let\endshaded\relax
\endshaded11784
```

Way back when DA sent me a pre-release copy of v0.6 of his framed package [Ars03]. We also discussed good ways of embedding it into the class. One result being that I defined some some items `\AtBeginDocument`, but this is no longer required as DA has changed framed to cooperate with memoir. We continue to have sporadic email conversations about the interaction of memoir and framed.
Here is a modified version of `framed.sty`. In particular I moved some of DA's description of the code into the general commentary instead of comments within the code.

```
framed.sty   v 0.95    2007/10/04
Copyright (C) 1992-2007 by Donald Arseneau  (asnd@triumf.ca)
These macros may be freely transmitted, reproduced, or modified
provided that this notice is left intact.

====================== Begin Instructions ======================

                      framed.sty
                      ~~~~~~~~~~
Create framed, shaded, or differently highlighted regions that can
break across pages.  The environments defined are
  framed    -- ordinary frame box (\fbox) with edge at margin
  shaded    -- shaded background (\colorbox) bleeding into margin
  snugshade -- similar
  leftbar   -- thick vertical line in left margin
to be used like
\begin{framed}
 copious text
\end{framed}

But the more general purpose of this package is to facilitate the
creation of environments that enable page breaking
within arbitrary decorations using a simple new-environment definition
incorporating \FrameCommand and
\begin{MakeFramed}{settings} ... \end{MakeFramed}

The "framed" environment uses "\fbox" as its "\FrameCommand" with
the additional settings \fboxrule=\FrameRule and \fboxsep=\FrameSep.
You can change these lengths (using \setlength) and you can change
the definition of \FrameCommand to use much fancier boxes.

In fact, the "shaded" environment just redefines \FrameCommand to be
\colorbox{shadecolor} (and you have to define the color "shadecolor":
\definecolor{shadecolor}...).
```

```
 A page break is allowed, and even encouraged, before the framed
 environment.  If you want to attach some text (a box title) to the
 frame, then the text should be inserted by \FrameCommand.

 The contents of the framed regions are restricted:
 Floats, footnotes, marginpars and head-line entries will be lost.
 (Some of these may be handled in a later version.)
 This package will not work with the page breaking of multicol.sty,
 or other systems that perform column-balancing.

 The MakeFramed environment does the work.  Its "settings" argument
 should contain any adjustments to the text width (applied to \hsize,
 and using the "\width" of the frame itself) as well as a "restore"
 command -- \@parboxrestore or \FrameRestore or something similar;
 as an example, the snugshade environment shows how to suppress excess
 spacing within the box, copying the code from minipage.

 Expert commands:
 \MakeFramed, \endMakeFramed: the "MakeFramed" environment
 \FrameCommand: command to draw the frame around its argument
 \FrameRestore: restore some text settings, but fewer than \@parboxrestore
 \FrameRule: length register; \fboxrule for default "framed".
 \FrameSep: length register; \fboxsep for default "framed".
 \FrameHeightAdjust: macro; height of frame above baseline at top of page

 This is still a 'pre-production' version because I can think of many
 features/improvements that should be made.  Nevertheless, starting
 with version 0.5 it should be bug-free.

 ToDo:
 Test more varieties of list
 Improve and correct documentation
 Propagation of \marks
 Handle footnotes (how??) floats (?) and marginpars.
 Stretchability modification.
======================== End Instructions ========================

 \ProvidesPackage{framed}[2007/10/04 v 0.95:
    framed or shaded text with page breaks]

 \newenvironment{framed}% using default \FrameCommand
   {\MakeFramed {\advance\hsize-\width \FrameRestore}}%
   {\endMakeFramed}

 \newenvironment{shaded}{%
   \def\FrameCommand{\fboxsep=\FrameSep \colorbox{shadecolor}}%
   \MakeFramed {\FrameRestore}}%
   {\endMakeFramed}
```

```
\newenvironment{leftbar}{%
  \def\FrameCommand{\vrule width 3pt \hspace{10pt}}%
  \MakeFramed {\advance\hsize-\width \FrameRestore}}%
 {\endMakeFramed}

\newenvironment{snugshade}{%
  \def\FrameCommand{\colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore\@setminipage}}%
 {\par\unskip\endMakeFramed}
```

Now for the package code.

```
\chardef\FrameRestore=\catcode`\| % for debug
\catcode`\|=\catcode`\% % (debug: insert space after backslash)
```

`\MakeFramed`

```
\def\MakeFramed#1{\par
```

measure added width and height; call result `\width` and `\height`

```
  \fb@sizeofframe\FrameCommand
  \let\width\fb@frw \let\height\fb@frh
```

insert pre-penalties and skips

```
  \begingroup
  \skip@\lastskip
  \if@nobreak\else
     \penalty9999 % updates \page parameters
     \ifdim\pagefilstretch=\z@ \ifdim\pagefillstretch=\z@
```

not infinitely stretchable, so encourage a page break here

```
        \edef\@tempa{\the\skip@}%
        \ifx\@tempa\zero@glue \penalty-30
        \else \vskip-\skip@ \penalty-30 \vskip\skip@
     \fi\fi\fi
     \penalty\z@
```

Give a stretchy breakpoint that will always be taken in preference to the `\penalty` 9999 used to update page parameters. The cube root of 10000/100 indicates a multiplier of 0.21545, but the maximum calculated badness is really 8192, not 10000, so the multiplier is 0.2301.

```
     \advance\skip@ \z@ plus-.5\baselineskip
     \advance\skip@ \z@ plus-.231\height
     \advance\skip@ \z@ plus-.231\skip@
     \advance\skip@ \z@ plus-.231\topsep
     \vskip-\skip@ \penalty 1800 \vskip\skip@
  \fi
  \addvspace{\topsep}%
  \endgroup
```

clear out pending page break

```
\penalty\@M \vskip 2\baselineskip \vskip\height
\penalty9999 \vskip -2\baselineskip \vskip-\height
\penalty9999 % updates \pagetotal
\message{After clearout, \pagetotal=\the\pagetotal,
         \pagegoal=\the\pagegoal. }%
\fb@adjheight
\setbox\@tempboxa\vbox\bgroup
  #1% Modifications to \hsize (can use \width and \height)
  \textwidth\hsize \columnwidth\hsize}
```

`\endMakeFramed`

```
\def\endMakeFramed{\par
    \kern\z@
    \hrule\@width\hsize\@height\z@
    \penalty-100 % put depth into height
\egroup
 % {\showoutput\showbox\@tempboxa}%
\begingroup
  \fb@put@frame\FrameCommand\FirstFrameCommand
\endgroup}
```

`\fb@put@frame` `\fb@put@frame{`⟨*nosplit*⟩`}{`⟨*split*⟩`}` takes the contents of `\@tempboxa` and puts all, or a piece, of it on the page with a frame (`\FrameCommand`, `\FirstFrameCommand`, `\MidFrameCommand`, or `\LastFrameCommand`). It recurses until all of `\@tempboxa` has been used up. (`\@tempboxa` must have zero depth.)
⟨*nosplit*⟩ = attempted framing command, if no split
⟨*split*⟩ = framing command if split.
First iteration: Try to fit with `\FrameCommand`. If it does not fit, split for `\FirstFrameCommand`.
Later iteration: Try to fit with `\LastFrameCommand`. If it does not fit, split for `\MidFrameCommand`.

```
\def\fb@put@frame#1#2{\relax
 \ifdim\pagegoal=\maxdimen \pagegoal\vsize \fi
 \ifinner
   \fb@putboxa#1%
   \fb@afterframe
 \else
  \dimen@\pagegoal \advance\dimen@-\pagetotal % natural space left on page
  \ifdim\dimen@<2\baselineskip % Too little room on page
    \message{Page has only \the\dimen@\space room left; eject. }%
    \eject \fb@adjheight \fb@put@frame#1#2%
  \else % there's appreciable room left on the page
     \fb@sizeofframe#1%
     \message{\string\pagetotal=\the\pagetotal,
         \string\pagegoal=\the\pagegoal,
```

```
        \string\pagestretch=\the\pagestretch,
        \string\pageshrink=\the\pageshrink,
        \string\fb@frh=\fb@frh. \space}
    \message{Box of size \the\ht\@tempboxa\space + \fb@frh}%
    \begingroup % temporarily set \dimen@ to be...
    \advance\dimen@.8\pageshrink  % maximum space available on page
    \advance\dimen@-\fb@frh\relax % space available for frame's contents
    \expandafter\endgroup
```

expand \ifdim, then restore \dimen@ to real room left on page

```
    \ifdim\dimen@>\ht\@tempboxa % whole box does fit
       \message{fits in \the\dimen@. }%
```

Use vsplit anyway to capture the marks
!!!???!!! MERGE THIS WITH THE else CLAUSE!!!

```
       \fb@putboxa#1%
       \fb@afterframe
    \else % box must be split
       \message{must be split to fit in \the\dimen@. }%
```

update frame measurement to use \FirstFrameCommand or \MidFrameCommand

```
       \fb@sizeoframe#2%
       \setbox\@tempboxa\vbox{% simulate frame and flexiblity of the page:
          \vskip \fb@frh \@plus\pagestretch \@minus.8\pageshrink
          \kern137sp\kern-137sp\penalty-30
          \unvbox\@tempboxa}%
       \edef\fb@resto@set{\boxmaxdepth\the\boxmaxdepth
                          \splittopskip\the\splittopskip}%
       \boxmaxdepth\z@ \splittopskip\z@
       \message{Padded box of size \the\ht\@tempboxa\space split
                to \the\dimen@}%
```

Split box here

```
       \setbox\tw@\vsplit\@tempboxa to\dimen@
       \toks99\expandafter{\splitfirstmark}%
       \toks98\expandafter{\splitbotmark}%
       \message{Marks are: \the\toks99, \the\toks98. }%
       \setbox\tw@\vbox{\unvbox\tw@}% natural-sized
       \message{Natural height of split box is \the\ht\tw@, leaving
          \the\ht\@tempboxa\space remainder. }%
```

If the split-to size > (\vsize-\topskip), then set box to full size

```
       \begingroup
       \advance\dimen@\topskip
       \expandafter\endgroup
       \ifdim\dimen@>\pagegoal
         \message{Frame is big -- Use up the full column. }%
         \dimen@ii\pagegoal
         \advance\dimen@ii -\topskip
         \advance\dimen@ii \FrameHeightAdjust\relax
       \else  % suspect this is wrong:
```

If the split-to size > feasible room-on-page, rebox it smaller.

```
          \advance\dimen@.8\pageshrink
          \ifdim\ht\tw@>\dimen@
            \message{Box too tall; rebox it to \the\dimen@. }%
            \dimen@ii\dimen@
          \else % use natural size
            \dimen@ii\ht\tw@
          \fi
        \fi
```

Re-box contents to desired size `\dimen@ii`

```
        \advance\dimen@ii -\fb@frh
        \setbox\tw@\vbox to\dimen@ii \bgroup
```

remove simulated frame and page flexibility:

```
        \vskip -\fb@frh \@plus-\pagestretch \@minus-.8\pageshrink
        \unvbox\tw@ \unpenalty\unpenalty
        \ifdim\lastkern=-137sp % whole box went to next page
           \message{box split at beginning! }%
           % need work here???
           \egroup \fb@resto@set \eject % (\vskip for frame size was discarded)
           \fb@adjheight
           \fb@put@frame#1#2% INSERTED ???
        \else % Got material split off at the head
           \egroup \fb@resto@set
           \ifvoid\@tempboxa % it all fit after all
              \message{box split at end! }%
              \setbox\@tempboxa\box\tw@
              \fb@putboxa#1%
              \fb@afterframe
           \else % it really did split
              \message{box split as expected. Its reboxed height
                       is \the\ht\tw@. }%
              \ifdim\wd\tw@>\z@
                \wd\tw@\wd\@tempboxa
```

PW: the next line was:
`\centerline{#2{\box\tw@}}%` ??
b ad
idea?
but I have changed it as follows so that `\centerline` can be changed from elsewhere if useful.

```
                \memfblineboxtwo{#2{\box\tw@}}%  ??? \centerline bad idea? \\
              \else
                \message{Zero width means likely blank.
                         Don't frame it (guess)}%
                \box\tw@
              \fi
              \hrule \@height\z@ \@width\hsize
              \eject
```

```
11921               \fb@adjheight
11922               \fb@put@frame\LastFrameCommand\MidFrameCommand
11923   \fi\fi\fi\fi\fi}
11924
```

\memfblineboxtwo
\memfblineboxa

PW: this is an addition of mine. There is a question mark against the use of \centerline within the framed code. I have created these two macros which are used in the code instead of \centerline, so that those parts of the code can be easily altered. are used instead

```
11925 \newcommand{\memfblineboxtwo}[1]{\centerline{#1}}
11926 \newcommand{\memfblineboxa}[1]{\centerline{#1}}
11927
```

\fb@putboxa

```
11928 \def\fb@putboxa#1{%
11929   \ifvoid\@tempboxa
11930 %%%%    PackageWarning{framed}{Boxa is void -- discard it. }%
11931     \@memwarn{Boxa is void -- discard it. }%
11932   \else
11933 |   \message{Frame and place boxa. }%
11934 |   %{\showoutput\showbox\@tempboxa}%
11935 %%%%%%%    \centerline{#1{\box\@tempboxa}}%
11936     \memfblineboxa{#1{\box\@tempboxa}}%
11937   \fi}
11938
```

\fb@afterframe

```
11939 \def\fb@afterframe{%
11940     \nointerlineskip \null %{\showoutput \showlists}
11941     \penalty-30 \vskip\topsep \relax}
11942
```

\fb@frw
\fb@frh
\fb@sizeofframe

\fb@sizeofframe{⟨*framecommand*⟩} measures width and height added by frame (⟨*framecommand*⟩) and call the results \fb@frw and \fb@frh.

```
11943 \newdimen\fb@frw
11944 \newdimen\fb@frh
11945 \def\fb@sizeofframe#1{\begingroup
11946  \setbox\z@\vbox{\vskip-5in \hbox{\hskip-5in
11947    #1{\hbox{\vrule \@height 4.7in \@depth.3in \@width 5in}}}%
11948    \vskip\z@skip}%
11949 |  \message{Measuring frame addition for \string#1 in \@currenvir\space
11950 |    gives ht \the\ht\z@\space and wd \the\wd\z@. }%
11951 %{\showoutput\showbox\z@}%
11952  \global\fb@frw\wd\z@ \global\fb@frh\ht\z@
11953  \endgroup}
11954
```

\fb@adjheight

```
11955 \def\fb@adjheight{%
```

```
11956   \vbox to\FrameHeightAdjust{}% get proper baseline skip from above.
11957   \penalty\@M \nointerlineskip
11958   \vskip-\FrameHeightAdjust
11959   \penalty\@M} % useful for tops of pages
11960
```

\zero@glue

```
11961 \edef\zero@glue{\the\z@skip}
11962
11963 \catcode`\|=\FrameRestore
11964
```

\FrameCommand Provide configuration commands:

\FrameRule \FrameSep \FirstFrameCommand \MidFrameCommand \LastFrameCommand

```
11965 \providecommand\FrameCommand{%
11966  \setlength\fboxrule{\FrameRule}\setlength\fboxsep{\FrameSep}%
11967  \fbox}
11968 \@ifundefined{FrameRule}{\newdimen\FrameRule \FrameRule=\fboxrule}{}
11969 \@ifundefined{FrameSep} {\newdimen\FrameSep  \FrameSep =3\fboxsep}{}
11970 \providecommand\FirstFrameCommand{\FrameCommand}
11971 \providecommand\MidFrameCommand{\FrameCommand}
11972 \providecommand\LastFrameCommand{\FrameCommand}
```

\FrameHeightAdjust Height of frame above first baseline when frame starts a page:

```
11973 \providecommand\FrameHeightAdjust{6pt}
11974
```

\FrameRestore \FrameRestore has parts of \@parboxrestore, performing a similar but less complete restoration of a default layout. See how it is used in the "settings" argument of \MakeFrame. Though not a parameter, \hsize should be set to the desired total line width available inside the frame before invoking \FrameRestore.

```
11975 \def\FrameRestore{%
11976   \let\if@nobreak\iffalse
11977   \let\if@noskipsec\iffalse
11978   \let\-\@dischyph
11979   \let\'\@acci\let\`\@accii\let\=\@acciii
11980   %  \message{FrameRestore:
11981   %    \@totalleftmargin=\the \@totalleftmargin,
11982   %    \rightmargin=\the\rightmargin,
11983   %    \@listdepth=\the\@listdepth.  }%
```

Test if we are in a list (or list-like paragraph)

```
11984   \ifnum \ifdim\@totalleftmargin>\z@ 1\fi
11985          \ifdim\rightmargin>\z@ 1\fi
11986          \ifnum\@listdepth>0 1\fi 0>\z@
11987     %     \message{In a list: \linewidth=\the\linewidth,
11988     %       \@totalleftmargin=\the\@totalleftmargin,
11989     %       \parshape=\the\parshape, \columnwidth=\the\columnwidth,
11990     %       \hsize=\the\hsize,
11991     %       \labelwidth=\the\labelwidth. }%
```

Now try to propagate changes of width from \hsize to list parameters. This is deficient, but a more advanced way to indicate modification to text dimensions is not (yet) provided; in particular, no separate left/right adjustment. (PW. I have provided a macro for the next few lines as I think that I may have a reason to allow them to be easily changed).

```
%%%     \@setminipage % snug fit around the item
%%%     \advance\linewidth-\columnwidth \advance\linewidth\hsize
%%%     \parshape\@ne \@totalleftmargin \linewidth
      \memfblistfixparams
    \else % Not in list
      \linewidth=\hsize
      %\message{No list, set \string\linewidth=\the\hsize. }%
    \fi
    \sloppy}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

\memfblistfixparams An extract of some code from \FrameRestore. Try to propagate changes of width from \hsize to list parameters. This is deficient, but a more advanced way to indicate modification to text dimensions is not (yet) provided; in particular, no separate left/right adjustment.

```
\newcommand*{\memfblistfixparams}{%
  \@setminipage % snug fit around the item
  \advance\linewidth-\columnwidth \advance\linewidth\hsize
  \parshape\@ne \@totalleftmargin \linewidth}

```

I have moved the following from the start of the framed package to here.

framed
shaded
leftbar
snugshade

```
\newenvironment{framed}% using default \FrameCommand
  {\MakeFramed {\advance\hsize-\width \FrameRestore}}%
  {\endMakeFramed}

\newenvironment{shaded}{%
  \def\FrameCommand{\fboxsep=\FrameSep \colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore}}%
  {\endMakeFramed}

\newenvironment{leftbar}{%
  \def\FrameCommand{\vrule width 3pt \hspace{10pt}}%
  \MakeFramed {\advance\hsize-\width \FrameRestore}}%
 {\endMakeFramed}

\newenvironment{snugshade}{%
  \def\FrameCommand{\colorbox{shadecolor}}%
  \MakeFramed {\FrameRestore\@setminipage}}%
 {\par\unskip\endMakeFramed}
```

```
12027
```

As the package used `\newenvironment` it will burp if it is used with the class. There are two options: either prevent framed from being loaded, or kill the environments if it is loaded. I'm going for the latter as Donald will be updating his package at some point and author's may want to use it before it's updated here.

```
12028 \AtBeginPackage{framed}{%
12029   \let\framed\relax \let\endframed\relax
12030   \let\shaded\relax \let\endshaded\relax
12031   \let\leftbar\relax \let\endleftbar\relax
12032   \let\snugshade\relax \let\endsnugshade\relax}
12033
```

qframe This is from an email to me from DA shortly after he released framed v 0.95.

> A frame more like a simple parbox. Notably useful for use inside a single list item — the frame matches the text of the other items.

He actually called it `pframe` and said that something similar could be in a later version of framed.
I have found it useful for framing `quote` or `adjustwidth` environments, or similar. Hence the name, from '*q*uote *frame*'.

```
12034 \newenvironment{qframe}{%
12035   \def\FrameCommand##1{\fboxrule=\FrameRule\fboxsep=\FrameSep
12036     \hskip\@totalleftmargin\fbox{##1}% There is no \@totalrightmargin, so...
12037       \hskip-\linewidth \hskip-\@totalleftmargin \hskip\columnwidth}%
12038   \MakeFramed{\advance\hsize-\width
12039               \advance\hsize \FrameSep
12040               \@totalleftmargin\z@ \linewidth=\hsize}}%
12041 {\endMakeFramed}
12042
```

And here is my corresponding version for shaded.

```
12043 \newenvironment{qshade}{%
12044   \def\FrameCommand##1{\fboxsep=\FrameSep
12045     \hskip\@totalleftmargin
12046     \hskip -1\FrameSep
12047     \colorbox{shadecolor}{##1}%
12048       \hskip-\linewidth \hskip-\@totalleftmargin
12049       \hskip -1\FrameSep
12050       \hskip\columnwidth}%
12051   \MakeFramed{\advance\hsize-\width
12052               \advance\hsize 3\FrameSep
12053               \@totalleftmargin\z@ \linewidth=\hsize}}%
12054 {\endMakeFramed}
12055
```

30.6 The newfile package

The following code is from the newfile package. To try and avoid name clashes with other packages, each internal macro in this package includes the character string 'stre@m'.

\newoutputstream `\newoutputstream{⟨stream⟩}` creates a new output stream called ⟨*stream*⟩. Different files may be associated with the ⟨*stream*⟩. Note that TeX permits no more than 16 output streams.

```
\newcommand*{\newoutputstream}[1]{%
  \@ifundefined{#1outstre@m}%
    {\expandafter\newwrite\csname #1outstre@m\endcsname
     \csname newif\expandafter\endcsname
       \csname ifstre@m#1open\endcsname
     \global\csname stre@m#1openfalse\endcsname
     \expandafter\ifx\csname atstreamopen#1\endcsname\relax
       \global\@namedef{atstreamopen#1}{}%
     \fi
     \expandafter\ifx\csname atstreamclose#1\endcsname\relax
       \global\@namedef{atstreamclose#1}{}%
     \fi
    }%
    {\@memwarn{Output stream #1\space is already defined}}}

```

\newinputstream `\newinputstream{⟨stream⟩}` creates a new input stream called ⟨*stream*⟩. Different files may be associated with the ⟨*stream*⟩. Note that TeX permits no more than 16 input streams.

```
\newcommand*{\newinputstream}[1]{%
  \@ifundefined{#1instre@m}%
    {\expandafter\newread\csname #1instre@m\endcsname
     \csname newif\expandafter\endcsname
       \csname ifstre@m#1open\endcsname
     \global\csname stre@m#1openfalse\endcsname
     \expandafter\ifx\csname atstreamopen#1\endcsname\relax
       \global\@namedef{atstreamopen#1}{}%
     \fi
     \expandafter\ifx\csname atstreamclose#1\endcsname\relax
       \global\@namedef{atstreamclose#1}{}%
     \fi
    }%
    {\@memwarn{Input stream #1\space is already defined}}}

```

Some checking macros will be useful as some of the checks occur in multiple places.

\IfStreamOpen `\IfStreamOpen{⟨stream⟩}{⟨TRUE code⟩}{⟨FALSE code⟩}` checks if stream ⟨*stream*⟩ is currently open.

```
\newcommand{\IfStreamOpen}[3]{%
  \csname ifstre@m#1open\endcsname#2\else#3\fi}
```

`\instre@mandopen` `\instre@mandopen{⟨stream⟩}{⟨TRUE code⟩}` checks if ⟨*stream*⟩ is an input stream and is open. If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\instre@mandopen}[2]{%
  \@ifundefined{#1instre@m}{%
    \@memwarn{#1\space is not an input stream}}%
  {\IfStreamOpen{#1}{#2}{%
    \@memwarn{Input stream #1\space is not open}}}}

```

`\instre@mandclosed` `\instre@mandclosed{⟨stream⟩}{⟨TRUE code⟩}` checks if ⟨*stream*⟩ is an input stream and is closed (not open). If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\instre@mandclosed}[2]{%
  \@ifundefined{#1instre@m}{%
    \@memwarn{#1\space is not an input stream}}%
  {\IfStreamOpen{#1}{%
    \@memwarn{Input stream #1\space is open}}{#2}}}

```

`\outstre@mandopen` `\outstre@mandopen{⟨stream⟩}{⟨TRUE code⟩}` checks if ⟨*stream*⟩ is an output stream and is open. If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\outstre@mandopen}[2]{%
  \@ifundefined{#1outstre@m}{%
    \@memwarn{#1\space is not an output stream}}%
  {\IfStreamOpen{#1}{#2}{%
    \@memwarn{Output stream #1\space is not open}}}}

```

`\outstre@mandclosed` `\outstre@mandclosed{⟨stream⟩}{⟨TRUE code⟩}` checks if ⟨*stream*⟩ is an output stream and is closed (not open). If so, it executes ⟨*TRUE code*⟩.

```
\newcommand{\outstre@mandclosed}[2]{%
  \@ifundefined{#1outstre@m}{%
    \@memwarn{#1\space is not an output stream}}%
  {\IfStreamOpen{#1}{%
    \@memwarn{Output stream #1\space is open}}{#2}}}

```

`\openoutputfile` `\openoutputfile{⟨filename⟩}{⟨stream⟩}` opens the file called ⟨*filename*⟩ and attaches it to the stream ⟨*stream*⟩ for writing. However, if the `\nofiles` command has been given the file is *not* attached to the stream. No more than one file can be attached to a stream at any given time.

```
\newcommand*{\openoutputfile}[2]{%
  \outstre@mandclosed{#2}{%
    \global\@namedef{#1@filename}{#1}%
    \if@filesw
      \immediate\openout\@nameuse{#2outstre@m}=\@nameuse{#1@filename}%
```

```
12117     \fi
12118     \global\csname stre@m#2opentrue\endcsname%
12119     \@nameuse{atstreamopen#2}}}
12120
```

\closeoutputstream `\closeoutputstream{⟨stream⟩}` closes the stream ⟨*stream*⟩.

```
12121 \newcommand*{\closeoutputstream}[1]{%
12122   \outstre@mandopen{#1}{%
12123     \@nameuse{atstreamclose#1}%
12124     \immediate\closeout\@nameuse{#1outstre@m}%
12125     \global\csname stre@m#1openfalse\endcsname}}
12126
```

\openinputfile `\openinputfile{⟨filename⟩}{⟨stream⟩}` opens the file called ⟨*filename*⟩ and attaches it to the stream ⟨*stream*⟩ for reading. The file is added to the list of files. No more than one file can be attached to a stream at any given time.

```
12127 \newcommand{\openinputfile}[2]{%
12128   \IfFileExists{#1}{%                    file exists
12129     \instre@mandclosed{#2}{%
12130       \@addtofilelist{#1}%
12131       \global\@namedef{#1@filename}{#1}%
12132       \immediate\openin\@nameuse{#2instre@m}=\@nameuse{#1@filename}%
12133       \global\csname stre@m#2opentrue\endcsname%
12134       \@nameuse{atstreamopen#2}}}%
12135   {%                                     file not found
12136     \typeout{No file #1.}
12137   }}
12138
```

\closeinputstream `\closeinputstream{⟨stream⟩}` closes the stream ⟨*stream*⟩.

```
12139 \newcommand{\closeinputstream}[1]{%
12140   \instre@mandopen{#1}{%
12141      \@nameuse{atstreamclose#1}%
12142      \immediate\closein\@nameuse{#1instre@m}%
12143      \global\csname stre@m#1openfalse\endcsname}}
12144
```

writeverbatim `\begin{writeverbatim}{⟨stream⟩}` writes the contents of the environment as verbatim text to the given ⟨*stream*⟩.

```
12145 \def\writeverbatim#1{%
12146   \@bsphack
12147   \let\do\@makeother\dospecials
12148   \catcode`\^^M\active
12149   \def\verbatim@processline{%
12150     \immediate\write\@nameuse{#1outstre@m}{\the\verbatim@line}}%
12151   \verbatim@start}
12152 \def\endwriteverbatim{\@esphack}
12153
```

\addtostream `\addtostream{⟨stream⟩}{⟨text⟩}` writes ⟨*text*⟩ to the given ⟨*stream*⟩.

```
\newcommand{\addtostream}[2]{%
  \@bsphack
  \outstre@mandopen{#1}{%
    {\let\protect\string
     \immediate\write\@nameuse{#1outstre@m}{#2}%
    }}%
  \@esphack}
```

\ifstre@mnoteof \checkstre@mnoteof `\checkstre@mnoteof{⟨stream⟩}` sets `\ifstre@mnoteof` to TRUE if ⟨*stream*⟩ is not at the end of the file (i.e., it is the opposite of `\ifeof`).

```
\newif\ifstre@mnoteof
\newcommand{\checkstre@meof}[1]{%
  \stre@mnoteoftrue\ifeof\@nameuse{#1instre@m}\stre@mnoteoffalse\fi}
```

\readstream `\readstream{⟨stream⟩}` reads the contents of the given ⟨*stream*⟩ as `\input` text.

```
\def\readstream#1{
  \instre@mandopen{#1}{%
    \loop \checkstre@meof{#1} \ifstre@mnoteof
      \read\@nameuse{#1instre@m} to\temptokstre@m
     \temptokstre@m
    \repeat
    }}
```

\readaline `\readaline{⟨stream⟩}` reads what TeX considers to be one line from the given ⟨*stream*⟩ as `\input` text.

```
\def\readaline#1{
  \instre@mandopen{#1}{%
    \ifeof\@nameuse{#1instre@m}
      \@memwarn{No more to read from stream #1}
    \else
      \read\@nameuse{#1instre@m} to\temptokstre@m
      \temptokstre@m
    \fi}}
```

\readverbatim `\readverbatim{⟨stream⟩}` reads the contents of the given ⟨*stream*⟩ as verbatim text.

The read verbatim code is a slight variation on code from the verbatim package. Most of the setup is done by the macros `\stre@mverb@input{⟨setup⟩}{⟨stream⟩}` and `\verbatim@readstre@m{⟨stream⟩}`. Finally, `\verbatim@read@file` is a verbatim package macro.

```
\def\readverbatim{\begingroup
  \ift@bs
    \def\verbatim@processline{\tabverbatim@processline}%
```

```
  \fi
  \@ifstar{\stre@mverb@input{\@maybeobeytabs}}%
     {\stre@mverb@input{\frenchspacing\@vobeyspaces\@maybeobeytabs}}}

```

`\stre@mverb@input` `\stre@mverb@input{⟨setup⟩}{⟨stream⟩}` is a stream version of `\verbatim@input`. It defines `\@verbinstre@m` to be ⟨*stream*⟩.
`\@verbinstre@m`

```
\newcommand{\stre@mverb@input}[2]{%
  \IfStreamOpen{#2}%
    {\@verbatim #1\relax
     \def\@verbinstre@m{\@nameuse{#2instre@m}}
     \verb@readstre@m\endtrivlist\endgroup\@doendpe}%
    {\@memwarn{Stream #2\space is not open}\endgroup}}

```

`\verb@readstre@m` `\verb@readstre@m` is the stream analogue to `\verbatim@readfile`.

```
\newcommand{\verb@readstre@m}{%
  \verbatim@startline
  \expandafter\endlinechar\expandafter\m@ne
  \expandafter\verbatim@read@stre@m
  \expandafter\endlinechar\the\endlinechar\relax
  \verbatim@finish}

```

`\verbatim@read@stre@m` `\verbatim@read@stre@m` is the analogue to `\verbatim@readfile`. It gets its input from the stream `\@verbinstre@m`, which is defined in the `\stre@mverb@input` macro.

```
\newcommand{\verbatim@read@stre@m}{%
  \read\@verbinstre@m to\next
  \ifeof\@verbinstre@m
  \else
    \expandafter\verbatim@addtoline\expandafter{\next}%
    \verbatim@processline
    \verbatim@startline
    \expandafter\verbatim@read@stre@m
  \fi}

```

`\readboxedverbatim` `\readboxedverbatim{⟨stream⟩}` is the equivalent of `\readverbtim` except that it generates a boxed verbatim.

```
\newcommand{\readboxedverbatim}{\begingroup
  \@ifstar{\stre@mbvin\relax}%
          {\stre@mbvin{\frenchspacing\@vobeyspaces}}}

```

`\stre@mbvin` `\stre@mbvin{⟨setup⟩}{⟨stream⟩}` is the workhorse for `\readboxedverbatim`, and in its turn it uses `\verb@readstre@m`.

```
\newcommand{\stre@mbvin}[2]{%
  \IfStreamOpen{#2}%
```

```
    {\setupboxverb@line
     \@verbatim #1\relax
     \def\@verbinstre@m{\@nameuse{#2instre@m}}%
     \setupbox@verb
     \verb@readstre@m\bvendrulehook\endtrivlist\endgroup\@doendpe}%
    {\@memwarn{Stream #2\space is not open}\endgroup}}

```

31 Utilities

31.1 Extra 'provide' commands

`\provideenvironment` Works like `\providecommand` but for environments.

```
\newcommand{\provideenvironment}{\@star@or@long\m@mprovenv}
```

`\m@mprovenv` This checks if the environment has been defined. If not it calls the kernel code for `\newenvironment` otherwise it calls code to discard arguments.

```
\newcommand{\m@mprovenv}[1]{\@ifundefined{#1}%
  {\new@environment{#1}}%       % create new environment
  {\@memwarn{Environment `#1' already defined}%
   \m@mgobbleoptsandtwo}}
```

`\m@mgobbleoptsandtwo` `\m@mgobbleoptsandtwo` gobbles the `[opt][opt]{begin}{end}` arguments to `\m@mgobbleoptandtwo` `\provideenvironment`. It is actually more general than that as it will gobble any sequence of any number of optional arguments followed by two required arguments. If there are no optional arguments it disposes of the two required ones, otherwise it calls a recursive procedure to discard any number of optionals, then two required ones.
`\m@mgobbleoptandtwo` recursively discards optional arguments, then finally two required arguments.

```
\newcommand{\m@mgobbleoptsandtwo}{%
  \@ifnextchar [{\m@mgobbleoptandtwo}{\@gobbletwo}}
\def\m@mgobbleoptandtwo[#1]{%
  \@ifnextchar [{\m@mgobbleoptandtwo}{\@gobbletwo}}

```

`\providecounter` Works like the other `\provide...` commands, but for counters. Code is based on `\newcounter` in `ltcounts.dtx`.

```
\newcommand*{\providecounter}[1]{%
  \@ifundefined{c@#1}%
    {\newcounter{#1}}%
    {\@memwarn{Counter `#1' already defined}%
      \@ifnextchar[{\m@mgobbleopt}{}}}

```

`\m@mgobbleopt` Gobble an optional argument.

```
\def\m@mgobbleopt[#1]{}

```

\providelength Works like the other \provide... commands, but for lengths. Code is based on \provide@command in ltdefns.dtx.

```
\newcommand*{\providelength}[1]{%
  \begingroup
    \escapechar\m@ne\xdef\@gtempa{{\string#1}}%
  \endgroup
  \expandafter\@ifundefined\@gtempa
    {\newlength{#1}}%
    {\@memwarn{Length #1 already defined}}}

```

\newloglike \m@mnewlog \m@mnewlogs For defining a new log-like math function. \newloglike{\fun}{string} or \newloglike*{\fun}{string}. The starred version is for functions with limits (like an integral sign).

```
\newcommand*{\newloglike}{\@ifstar{\m@mnewlogs}{\m@mnewlog}}
\newcommand*{\m@mnewlogs}[2]{%
  \newcommand*{#1}{\mathop{\operator@font #2}}}
\newcommand*{\m@mnewlog}[2]{%
  \newcommand*{#1}{\mathop{\operator@font #2}\nolimits}}

```

\provideloglike \m@mprovlogs \m@mprovlog The provide version of \newloglike.

```
\newcommand*{\provideloglike}{\@ifstar{\m@mprovlogs}{\m@mprovlog}}
\newcommand*{\m@mprovlogs}[2]{%
  \providecommand*{#1}{\mathop{\operator@font #2}}}
\newcommand*{\m@mprovlog}[2]{%
  \providecommand*{#1}{\mathop{\operator@font #2}\nolimits}}

```

31.2 Changing counters

In LaTeX, a new counter called, say 'ctr', is created by the command \newcounter{ctr}[within]. If the optional within argument is given the the counter 'ctr' is reset to zero each time the counter 'within' changes. The command \thectr typesets the value of the counter ctr. This is automatically defined by \newcounter and is initialised to typeset arabic numerals. Sometimes it may be desireable to change the counter definitions.
The following code is based on the chngcntr package [Wil01e].

\@removefromreset This code uses David Carlisle's \@removefromreset command as specified in the remreset package [Car98b]. It is provided here as a convenience to the user, and with David Carlisle's permission.
START OF DAVID CARLISLE'S CODE

```
\providecommand{\@removefromreset}[2]{{%
  \expandafter\let\csname c@#1\endcsname\@removefromreset
  \def\@elt##1{%
    \expandafter\ifx\csname c@##1\endcsname\@removefromreset
    \else
```

```
      \noexpand\@elt{##1}%
    \fi}%
  \expandafter\xdef\csname cl@#2\endcsname{%
    \csname cl@#2\endcsname}}
```

END OF DAVID CARLISLE'S CODE

`\@ifbothcntrs` This is called as `\@ifbothcntrs{`⟨*cntr*⟩`}{`⟨*within*⟩`}{`⟨*code when both are counters*⟩`}`. This tests if both ⟨*cntr*⟩ and ⟨*within*⟩ are counters.

```
\newcommand{\@ifbothcntrs}[3]{%
  \@ifundefined{c@#1}{% counter undefined
    \@mememrror{#1 is not a counter}{\@eha}}%
  {% else counter is defined
    \@ifundefined{c@#2}{% within undefined
      \@mememrror{#2 is not a counter}{\@eha}}%
    {% else both counter and within  are defined
     #3}}}
```

`\counterwithin` `\counterwithin*` `\@csinstar` `\@csin` It is sometimes desireable to change a counter that has been defined by `\newcounter{ctr}` to act as though it had been defined as `\newcounter{ctr}[within]`. The command `\counterwithin{ctr}{within}` accomplishes this. By default, it also redefines the `\thectr` command so that it typesets values in the style `\thewithin.\arabic{ctr}`. The starred version of the command suppresses the redefinition of `\thectr`.

```
\newcommand{\counterwithin}{\@ifstar{\@csinstar}{\@csin}}
\newcommand{\@csinstar}[2]{%
  \@ifbothcntrs{#1}{#2}{\@addtoreset{#1}{#2}}}
\newcommand{\@csin}[2]{%
  \@ifbothcntrs{#1}{#2}{\@addtoreset{#1}{#2}%
                        \@namedef{the#1}{\@nameuse{the#2}.\arabic{#1}}}}
```

`\counterwithout` `\counterwithout*` `\@csoutstar` `\@csout` Likewise, the command `\counterwithout{ctr}{within}` changes a counter that has been created by `\newcounter{ctr}[within]` to act as though it had been created by `\newcounter{ctr}`. By default it also redefines the `\thectr` command so that it just typesets an arabic numeral. The starred version of the command suppresses the redefinition of `\thectr`.

```
\newcommand{\counterwithout}{\@ifstar{\@csoutstar}{\@csout}}
\newcommand{\@csoutstar}[2]{%
  \@ifbothcntrs{#1}{#2}{\@removefromreset{#1}{#2}}}
\newcommand{\@csout}[2]{%
  \@ifbothcntrs{#1}{#2}{\@removefromreset{#1}{#2}%
                        \@namedef{the#1}{\arabic{#1}}}}
```

Any number of `\counterwithin{ctr}{...}` and `\counterwithout{ctr}{...}` commands can be issued for a given counter, ctr, if you wish to toggle between

the two styles. The current value of ctr is unaffected by \counterwithin and \counterwithout. If you want to change the value after one of these commands, use \setcounter{ctr}{...}, and to change the typeseting style use \renewcommand{\thectr}{...}.

\letcountercounter
\c@m@morig@ctr
\unletcounter

At times it is handy to 'let' one counter act as if it was a different counter. Say you have two constructions, each with their own counter A and B, now you want them to cooperate, counting in unison. This can be done using the \letcountercounter.

\letcountercounter{⟨*counterA*⟩}{⟨*counterB*⟩} \lets (make the same) ⟨*counterA*⟩ to ⟨*counterB*⟩. The original of ⟨*counterA*⟩ is kept in \c@m@morig@ctr⟨*counterA*⟩.
\unletcounter{⟨*counterA*⟩} restores ⟨*counterA*⟩ to its un\let condition.
The code is based on Donald Arseneau's 'Re: \let, \csname and \endcsname', CTT, 2004/12/08. The long version, where \xp is actually \expandafter is along the lines of:

```
\xp\xp\xp\let\xp\csname\xp c@#1\xp\endcsname\csname c@#2\endcsname
```

which employs 6 \expandafters. DA's version uses only 2. The technique can be applied to other kinds of \lets.

```
\newcommand*{\letcountercounter}[2]{%
  \expandafter\let\csname c@m@morig@ctr#1\expandafter\endcsname%
                   \csname c@#1\endcsname
  \expandafter\let\csname c@#1\expandafter\endcsname%
                  \csname c@#2\endcsname}
\newcommand*{\unletcounter}[1]{%
 \expandafter\let\csname c@#1\expandafter\endcsname%
                 \csname c@m@morig@ctr#1\endcsname}
```

31.3 Odd/even page checking

It is difficult to check robustly if the current page is odd or even. This code, which is based on the chngpage package [Wil01b], provides a method that has proved (so far) to be robust. It works by writing out a label and on the next LaTeX run checks the page reference for the label.

\ifoddpage
\oddpagetrue
\oddpagefalse

The boolean \ifoddpage is TRUE if the checked page is odd. The boolean \ifstrictpagecheck is for turning on (TRUE) and off (FALSE) the strictest method of page checking.

\ifstrictpagecheck
\strictpagechecktrue
\strictpagecheckfalse
\strictpagecheck
\easypagecheck

```
\newif\ifoddpage
\newif\ifstrictpagecheck
  \strictpagecheckfalse
\newcommand*{\strictpagecheck}{\strictpagechecktrue}
\newcommand*{\easypagecheck}{\strictpagecheckfalse}
```

\c@cp@cntr
\cplabel
The counter cp@cntr is used to make unique labels, which start with \cplabel.

```
\newcounter{cp@cntr}
\newcommand{\cplabel}{^_}
```

\checkoddpage
This is the user level command to check for odd/even page. It does a robust check for \strictpagecheck otherwise the simple minded check. It sets \ifoddpage to TRUE if the page is odd, otherwise it sets it to FALSE.
This is now fixed so that it should work for non-arabic page numbering. It uses a new label/pageref variant based on the page counter value, not its printed representation. This also gets rid of worrying about hyperref!! The problem was discovered by Bastiaan Niels Veelo

```
\DeclareRobustCommand{\checkoddpage}{%
  \oddpagefalse%
  \ifstrictpagecheck%
    \stepcounter{cp@cntr}\pmemlabel{\cplabel\thecp@cntr}%
    \@memcnta=\pmemlabelref{\cplabel\thecp@cntr}\relax
    \ifodd\@memcnta\oddpagetrue\fi
  \else
    \ifodd\c@page\oddpagetrue\fi
  \fi}

```

\thepmemc@@page
The value of the page counter.

```
\gdef\thepmemc@@page{\the\c@page}

```

\pmemprotected@write
\pmemprotected@write is a modified version of the kernel's \protected@write. This version of the macro was provided by Romano Giannetti (romano@dea.icai.upco.es) on 15 July 2003.

```
\long\def\pmemprotected@write#1#2#3{%
  \begingroup
  \let\thepmemc@@page\relax
  #2%
  \let\protect\@unexpandable@protect
  \edef\reserved@a{\write#1{#3}}%
  \reserved@a
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi}

```

\pmemlabel
A version of \label that uses \pmemprotected@write.

```
\newcommand{\pmemlabel}[1]{\@bsphack
  \pmemprotected@write\@auxout{}%
    {\string\newpmemlabel{#1}{\thepmemc@@page}}%
  \@esphack}
```

\newpmemlabel
\pmemlabelref
We have to be able to cope with a particular label not (yet) being in the the .aux file when we come to check the page number.

```
\newcommand{\newpmemlabel}[2]{{\global\@namedef{m@#1}{#2}}}
\newcommand{\pmemlabelref}[1]{%
  \expandafter\ifx\csname m@#1\endcsname\relax
    0%
  \else
    \csname m@#1\endcsname
  \fi}
```

31.4 Checking for empty arguments

Like page checking, testing for an empty macro argument is more difficult than it might appear at first sight.
The following code is from the ifmtarg package [ArWi00].

`\@ifmtarg` `\@ifmtarg{`⟨*arg*⟩`}{`⟨*code when empty*⟩`}{`⟨*code when arg not empty*⟩`}`
`\@ifnotmtarg` `\@ifnotmtarg{`⟨*arg*⟩`}{`⟨*code when arg not empty*⟩`}`

```
\begingroup
\catcode`\Q=3
\long\gdef\@ifmtarg#1{\@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
\long\gdef\@xifmtarg#1#2Q#3#4#5\@nil{#4}
\long\gdef\@ifnotmtarg#1{\@xifmtarg#1QQ\@firstofone\@gobble\@nil}
\endgroup
```

Some example uses:

```
\newcommand{\isempty}[1]{%
   \@ifmtarg{#1}{\typeout{YES}}{\typeout{NO}}}
\newcommand{\isnotempty}[1]{%
   \@ifnotmtarg{#1}{\typeout{YES}}}

\isempty{}    -> YES  \isnotempty{}    ->
\isempty{   } -> YES  \isnotempty{   } ->
\isempty{A}   -> NO   \isnotempty{A}   -> YES
\isempty{ A } -> NO   \isnotempty{ A } -> YES
```

31.5 Changing the page layout in the document

You should not do this, but...
The following code is essentially from the chngpage package [Wil01b].

`\ch@ngetext` This macro sets the page output parameters.

```
\DeclareRobustCommand{\ch@ngetext}{%
  \setlength{\@colht}{\textheight}\setlength{\@colroom}{\textheight}%
  \setlength{\vsize}{\textheight}\setlength{\columnwidth}{\textwidth}%
  \if@twocolumn%
    \advance\columnwidth-\columnsep \divide\columnwidth\tw@%
```

```
    \@firstcolumntrue%
  \fi%
  \setlength{\hsize}{\columnwidth}%
  \setlength{\linewidth}{\hsize}}
```

`\changetext` `\changetext{⟨H⟩}{⟨W⟩}{⟨E⟩}{⟨O⟩}{⟨G⟩}` adds the given lengths to 5 main bits of the page layout. An empty argument means 'no change'.

```
\DeclareRobustCommand{\changetext}[5]{%
  \@ifmtarg{#1}{}{\addtolength{\textheight}{#1}}%
  \@ifmtarg{#2}{}{\addtolength{\textwidth}{#2}}%
  \@ifmtarg{#3}{}{\addtolength{\evensidemargin}{#3}}%
  \@ifmtarg{#4}{}{\addtolength{\oddsidemargin}{#4}}%
  \@ifmtarg{#5}{}{\addtolength{\columnsep}{#5}}%
  \ch@ngetext}
```

Any given change lasts until another `\changetext` command is given. A `\changetext` command should only be issued between two paragraphs. `\changetext` should not be used by anyone unless they really know what they are doing. If you really know what you are doing then you should know enough not to use it.

`\changepage` Change the page layout, but DON'T.

```
\DeclareRobustCommand{\changepage}[9]{%
  \@ifmtarg{#1}{}{\addtolength{\textheight}{#1}}%
  \@ifmtarg{#2}{}{\addtolength{\textwidth}{#2}}%
  \@ifmtarg{#3}{}{\addtolength{\evensidemargin}{#3}}%
  \@ifmtarg{#4}{}{\addtolength{\oddsidemargin}{#4}}%
  \@ifmtarg{#5}{}{\addtolength{\columnsep}{#5}}%
  \ch@ngetext
  \@ifmtarg{#6}{}{\addtolength{\topmargin}{#6}}%
  \@ifmtarg{#7}{}{\addtolength{\headheight}{#7}}%
  \@ifmtarg{#8}{}{\addtolength{\headsep}{#8}}%
  \@ifmtarg{#9}{}{\addtolength{\footskip}{#9}}}
```

31.6 Temporarily changing the text width

`adjustwidth` `adjustwidth*` `\begin{adjustwidth}{⟨left⟩}{⟨right⟩}` adds the given lengths to the left and right hand margins. A positive value will shorten the text and a negative value will widen it. The starred version of the environment will cause the margin changes to switch between odd and even pages.
This code is based on the chngpage package.

```
\newenvironment{adjustwidth}[2]{%
  \begin{list}{}{%
    \topsep\z@%
    \listparindent\parindent%
```

```
    \parsep\parskip%
    \@ifmtarg{#1}{\setlength{\leftmargin}{\z@}}%
                 {\setlength{\leftmargin}{#1}}%
    \@ifmtarg{#2}{\setlength{\rightmargin}{\z@}}%
                 {\setlength{\rightmargin}{#2}}%
    }
    \item[]}{\end{list}}

\newenvironment{adjustwidth*}[2]{%
  \begin{list}{}{%
    \topsep\z@%
    \listparindent\parindent%
    \parsep\parskip%
    \checkoddpage
    \ifoddpage  % odd numbered page
      \@ifmtarg{#1}{\setlength{\leftmargin}{\z@}}%
                   {\setlength{\leftmargin}{#1}}%
      \@ifmtarg{#2}{\setlength{\rightmargin}{\z@}}%
                   {\setlength{\rightmargin}{#2}}%
    \else       % even numbered page
      \@ifmtarg{#2}{\setlength{\leftmargin}{\z@}}%
                   {\setlength{\leftmargin}{#2}}%
      \@ifmtarg{#1}{\setlength{\rightmargin}{\z@}}%
                   {\setlength{\rightmargin}{#1}}%
    \fi
    }
    \item[]}{\end{list}}
```

The environments only work for complete paragraphs.

31.7 Centering text

`\calccentering` This macro calculates the amount to be added to the spine margin (and subtracted from the foredge margin) in order to center the textblock. Call as `\calccentering{\length}` and it sets `\length` to the required value. Then use as
`\begin{adjustwidth*}{\length}{-\length}...`

```
\newcommand{\calccentering}[1]{
  #1 = \paperwidth
  \advance #1 by -\textwidth
  \divide #1 by \tw@
  \advance #1 by -\spinemargin}
```

`vplace` `\begin{vplace}[`⟨*decimal number*⟩`]` centers its body vertically. The optional argument ⟨*decimal number*⟩ increases (> 1.0) or decreases (< 1.0) the space above the body with respect to the space below. This is more robust than the method proposed in the manual.

```
\newenvironment{vplace}[1][1]{%
  \par\vspace{\stretch{#1}}%
}{%
  \vspace*{\stretch{1}}%
  \par}
```

31.8 Moving from the current page

Much of this code is taken from the nextpage package [Wil00c].
Clear to next page is \clearpage.

\cleartoevenpage `\cleartoevenpage[`⟨*text*⟩`]` clears to the next even numbered page, putting the optional ⟨*text*⟩ on the skipped page, if any.

```
\newcommand{\cleartoevenpage}[1][\@empty]{%
  \clearpage%
  \ifodd\c@page\hbox{}#1\clearpage\fi}
```

Move to the next page without flushing floats is \newpage.

\movetoevenpage `\movetoevenpage[`⟨*text*⟩`]` moves to the next even numbered page without flushing floats, putting the optional ⟨*text*⟩ on the skipped page, if any.

```
\newcommand{\movetoevenpage}[1][\@empty]{%
  \newpage%
  \ifodd\c@page\hbox{}#1\newpage\fi}
```

Clear to next odd numbered page is \cleardoublepage.

\cleartooddpage `\cleartooddpage[`⟨*text*⟩`]` clears to the next odd numbered page, putting the optional ⟨*text*⟩ on the skipped page, if any.

```
\newcommand{\cleartooddpage}[1][\@empty]{%
  \clearpage%
  \ifodd\c@page\else\hbox{}#1\clearpage\fi}
```

\movetooddpage `\movetooddpage[`⟨*text*⟩`]` moves to the next odd numbered page without flushing floats, putting the optional ⟨*text*⟩ on the skipped page, if any.

```
\newcommand{\movetooddpage}[1][\@empty]{%
  \newpage%
  \ifodd\c@page\else\hbox{}#1\newpage\fi}
```

Example uses:

```
\cleartooddpage                          % same as \cleardouble page
\cleartooddpage[\thispagestyle{empty}]   % No headings on the skipped page
\cleartoevenpage                         % go to next even numbered page
```

```
Next example puts text on a skipped page
\cleartoevenpage[\vspace*{\hfill}THIS PAGE LEFT BLANK\vspace*{\hfill}]
```

31.9 Needing space at the bottom of a page

\needspace \needspace{⟨*length*⟩} checks if the is ⟨*length*⟩ amount of vertical space left on the page. If there is not it will start a new page. The code is a variant of code thought of for the needspace package [Wil00d].

```
12446 \newcommand{\needspace}[1]{\begingroup\setlength{\dimen@}{#1}%
12447   \vskip\z@\@plus\dimen@\penalty -100\vskip\z@\@plus-\dimen@
12448   \vskip\dimen@\penalty 9999\vskip-\dimen@\endgroup}
12449
```

\Needspace \Needspace and \Needspace* are more robust versions of \needspace, which
\Needspace* depends on penalties. If either \needspace or \Needspace produce a short page it will be ragged bottom, even if \flushbottom is in effect. The \Needspace* version honours the \...bottom declaration.

```
12450 \newcommand{\Needspace}{\@ifstar{\M@sneedsp@}{\M@needsp@}}
```

\M@sneedsp@ These implement \Needspace* and \needspace respectively.
\M@needsp@

```
12451 \newcommand{\M@sneedsp@}[1]{\par \penalty-100\begingroup
12452   \setlength{\dimen@}{#1}%
12453   \dimen@ii\pagegoal \advance\dimen@ii-\pagetotal
12454   \ifdim \dimen@>\dimen@ii
12455     \break
12456   \fi\endgroup}
12457 \newcommand{\M@needsp@}[1]{\par \penalty-100\begingroup
12458   \setlength{\dimen@}{#1}%
12459   \dimen@ii\pagegoal \advance\dimen@ii-\pagetotal
12460   \ifdim \dimen@>\dimen@ii
12461     \ifdim \dimen@ii>\z@
12462       \vfil
12463     \fi
12464     \break
12465   \fi\endgroup}
12466
```

31.10 Overlong lines

LaTeX provides \fussy and \sloppy to control the amount of slack in a line while trying to make justified lines. Their definitions are:

```
\def\fussy{%
  \emergencystretch\z@
  \tolerance 200%
  \hfuzz .1\p@
  \vfuzz\hfuzz}
\def\sloppy{%
```

```
  \tolerance 9999%
  \emergencystretch 3em%
  \hfuzz .5\p@
  \vfuzz\hfuzz}
```

`\midsloppy` Somewhere between `\fussy` and `\sloppy`.

```
\newcommand*{\midsloppy}{%
  \tolerance 5000%
  \hbadness 4000%
  \emergencystretch 1.5em%
  \hfuzz .1\p@
  \vfuzz\hfuzz}
```

`midsloppypar` A paragraph form of `\midsloppy`; equivalent to `\par \midsloppy ... \par`

```
\newenvironment{midsloppypar}{\par\midsloppy}{\par}
```

31.11 Text spacing commands

While we're at it, new text spacing commands to supplement the kernel's `\, (\thinspace) = 3/18 em`

`\medspace` Medium space `\: (\medspace) = 4/18 em`

`\:`

```
\newcommand{\medspace}{\kern .22222em }
\DeclareRobustCommand{\:}{%
  \relax\ifmmode\mskip\medmuskip\else\medspace\fi}
```

`\!` Negative thin space (- 3/18 em). Patrik Nyman (private email 2003/10/06) pointed out I had missed the final `\fi` !!!!.

```
\DeclareRobustCommand{\!}{%
  \relax\ifmmode\mskip-\thinmuskip\else\negthinspace\fi}
```

31.12 Fractions and subscripts

Styles for fractions like 3/4.

`\slashfracstyle` `\slashfracstyle` based on the kernel `\textsuperscript` macro

```
\DeclareRobustCommand*{\slashfracstyle}[1]{%
  {\m@th\ensuremath{\mbox{\fontsize\sf@size\z@\selectfont #1}}}}
```

`\slashfrac` `\slashfrac` based on TeXbook exercise 11.6

```
\DeclareRobustCommand*{\slashfrac}[2]{\leavevmode
  \raise.5ex\hbox{\slashfracstyle{#1}}\kern-.13em/%
  \kern-.15em\lower.25ex\hbox{\slashfracstyle{#2}}}
```

\textsubscript
\@textsubscript

\textsubscript, for text subscripts, based on the kernel \textsuperscript macro.

```
\DeclareRobustCommand*{\textsubscript}[1]{%
  \@textsubscript{\selectfont#1}}
\newcommand*{\@textsubscript}[1]{%
  {\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
```

31.13 Numbers to names

```
%%%%%%%%%%%%%%%%%%% number formatting
```

The macros here convert an integer number in the range 0–2147483647 (TeX's maximum) into the natural language name of the number.

\iflowernumtoname

A flag for lowercasing the initial letters.

```
\newif\iflowernumtoname
  \lowernumtonamefalse
```

\ifpriornum
\ifminusnumber
\ifnotnumtonameallcaps
\ifmakeordinal

Some new booleans

```
\newif\ifpriornum
\newif\ifminusnumber
\newif\ifnotnumtonameallcaps
\newif\ifmakeordinal
```

\namenumberand
\namenumbercomma
\lcminusname
\ucminusname
\minusname

Some macros representing characters/words for number formatting.

```
\newcommand*{\namenumberand}{ and }
\newcommand*{\namenumbercomma}{, }
\newcommand*{\lcminusname}{minus }
\newcommand*{\ucminusname}{Minus }
\let\minusname\lcminusname
```

\fnumbersep
\tensunitsep
\nthstring
\iststring
\iindstring
\iiirdstring
\tiethstring
\teenstring

More characters/words for formatting

```
\newcommand*{\fnumbersep}{,}
\newcommand*\tensunitsep{-}
\newcommand*{\nthstring}{th}      % nth
\newcommand*{\iststring}{st}      % 1st
\newcommand*{\iindstring}{nd}     % 2nd
\newcommand*{\iiirdstring}{rd}    % 3rd
\newcommand*{\tiethstring}{tieth} % tieth
\newcommand*{\teenstring}{teen}   % teen
```

\ordscript

How to format the ordinal string

```
\newcommand{\ordscript}[1]{#1}
```

\m@mten

An internal shorthand for 10

```
\chardef\m@mten=10 % shorthand for 10
```

\c@ism@mctr
\c@xsm@mctr
\c@csm@mctr
\c@ksm@mctr
\c@xksm@mctr
Counters for digits (units to ten thousands) in a number

```
\newcounter{ism@mctr}  % units
\newcounter{xsm@mctr}  % tens
\newcounter{csm@mctr}  % hundreds
\newcounter{ksm@mctr}  % thousands
\newcounter{xksm@mctr} % ten thousands
```

\c@cksm@mctr
\c@msm@mctr
\c@xmsm@mctr
\c@cmsm@mctr
\c@bsm@mctr
Counters for digits (hundred thousands to billions) in a number

```
\newcounter{cksm@mctr} % hundred thousands
\newcounter{msm@mctr}  % millions
\newcounter{xmsm@mctr} % ten millions
\newcounter{cmsm@mctr} % hundred millions
\newcounter{bsm@mctr}  % billions
```

\c@workm@mctr
A 'work' counter.

```
\newcounter{workm@mctr}

```

\numdigits
\numdigits{⟨*number*⟩} splits the ⟨*number*⟩ into individual digits. It sets \minusnumbertrue if the number is negative, otherwise it is false.

```
\newcommand*{\numdigits}[1]{%
  \setcounter{ism@mctr}{0}%
  \setcounter{xsm@mctr}{0}%
  \setcounter{csm@mctr}{0}%
  \setcounter{ksm@mctr}{0}%
  \setcounter{xksm@mctr}{0}%
  \setcounter{cksm@mctr}{0}%
  \setcounter{msm@mctr}{0}%
  \setcounter{xmsm@mctr}{0}%
  \setcounter{cmsm@mctr}{0}%
  \setcounter{bsm@mctr}{0}%
  \setcounter{workm@mctr}{#1}%
  \minusnumberfalse
  \ifnum \c@workm@mctr < \z@  % negative
    \minusnumbertrue
    \c@workm@mctr = -\c@workm@mctr
  \fi
  \ifnum \c@workm@mctr > \m@ne     % units
    \c@ism@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@ism@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@    % tens
    \c@xsm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@xsm@mctr by -\c@workm@mctr
```

```
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@   % hundreds
    \c@csm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@csm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@   % thousands
    \c@ksm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@ksm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@   % ten thousands
    \c@xksm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@xksm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@   % hundred thousands
    \c@cksm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@cksm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@   % millions
    \c@msm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@msm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@  % ten millions
    \c@xmsm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@xmsm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
  \fi
  \ifnum \c@workm@mctr > \z@  % hundred millions
    \c@cmsm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@cmsm@mctr by -\c@workm@mctr
    \divide \c@workm@mctr by \m@mten
```

```
  \fi
  \ifnum \c@workm@mctr > \z@% billions
    \c@bsm@mctr = \c@workm@mctr
    \divide \c@workm@mctr by \m@mten
    \multiply \c@workm@mctr by \m@mten
    \advance \c@bsm@mctr by -\c@workm@mctr
  \fi}

```

`\form@tnumber` `\form@tnumber{`⟨*number*⟩`}` formats ⟨*number*⟩ as digits.

```
\newcommand*{\form@tnumber}[1]{%
  \numdigits{#1}%
  \ifminusnumber-\fi
  \priornumfalse
  \ifnum \c@bsm@mctr > \z@ % billions
    \priornumtrue
    \thebsm@mctr\fnumbersep
  \fi
  \ifpriornum                 % hundred millions
    \thecmsm@mctr
  \else
    \ifnum \c@cmsm@mctr > \z@
      \priornumtrue
      \thecmsm@mctr
    \fi
  \fi
  \ifpriornum                 % ten millions
    \thexmsm@mctr
  \else
    \ifnum \c@xmsm@mctr > \z@
      \priornumtrue
      \thexmsm@mctr
    \fi
  \fi
  \ifpriornum                 % millions
    \themsm@mctr\fnumbersep
  \else
    \ifnum \c@msm@mctr > \z@
      \priornumtrue
      \themsm@mctr\fnumbersep
    \fi
  \fi
  \ifpriornum               % hundred thousands
    \thecksm@mctr
  \else
    \ifnum \c@cksm@mctr > \z@
      \priornumtrue
      \thecksm@mctr
    \fi
  \fi
```

```
  \ifpriornum                % ten thousands
    \thexksm@mctr
  \else
    \ifnum \c@xksm@mctr > \z@
      \priornumtrue
      \thexksm@mctr
    \fi
  \fi
  \ifpriornum                % thousands
    \theksm@mctr\fnumbersep
  \else
    \ifnum \c@ksm@mctr > \z@
      \priornumtrue
      \theksm@mctr\fnumbersep
    \fi
  \fi
  \ifpriornum                % hundreds
    \thecsm@mctr
  \else
    \ifnum \c@csm@mctr > \z@
      \priornumtrue
      \thecsm@mctr
    \fi
  \fi
  \ifpriornum                % tens
    \thexsm@mctr
  \else
    \ifnum \c@xsm@mctr > \z@
      \priornumtrue
      \thexsm@mctr
    \fi
  \fi
  \theism@mctr}              % units

```

`\cardinal` `\cardinal{`⟨*number*⟩`}` prints ⟨*number*⟩ unformatted.

```
\newcommand*{\cardinal}[1]{%
  \begingroup
  \let\fnumbersep\relax
  \form@tnumber{#1}%
  \endgroup}
```

`\fcardinal` `\fcardinal{`⟨*number*⟩`}` prints ⟨*number*⟩ formatted.

```
\newcommand*{\fcardinal}[1]{%
  \begingroup
  \form@tnumber{#1}%
  \endgroup}

```

`\ordinal` `\ordinal{`⟨*number*⟩`}` prints ⟨*number*⟩ as an unformatted ordinal.

```
\newcommand*{\ordinal}[1]{%
  \begingroup
  \let\fnumbersep\relax
  \form@tnumber{#1}%
  \let\ordstring\nthstring
  \ifnum \c@xsm@mctr=\@ne\else
    \ifcase \c@ism@mctr
      \or \let\ordstring\iststring%    1st
      \or \let\ordstring\iindstring%   2nd
      \or \let\ordstring\iiirdstring%  3rd
    \fi
  \fi
  \ordscript{\ordstring}%
  \endgroup
}
```

\fordinal `\fordinal{`⟨*number*⟩`}` prints ⟨*number*⟩ as a formatted ordinal.

```
\newcommand*{\fordinal}[1]{%
  \begingroup
  \form@tnumber{#1}%
  \let\ordstring\nthstring
  \ifnum \c@xsm@mctr=\@ne\else
    \ifcase \c@ism@mctr
      \or \let\ordstring\iststring%    1st
      \or \let\ordstring\iindstring%   2nd
      \or \let\ordstring\iiirdstring%  3rd
    \fi
  \fi
  \ordscript{\ordstring}%
  \endgroup
}

```

The next, tedious, code is for translating numbers into names.

\nNameo \nNamec \nNamem \nNamemm \nNamemmm Names of major numbers: 0, 10^2, 10^3, 10^6, and 10^9.

```
\newcommand*\nNameo{\iflowernumtoname z\else Z\fi ero}
\newcommand*\nNamec{\iflowernumtoname h\else H\fi undred}
\newcommand*\nNamem{\iflowernumtoname t\else T\fi housand}
\newcommand*\nNamemm{\iflowernumtoname m\else M\fi illion}
\newcommand*\nNamemmm{\iflowernumtoname b\else B\fi illion}

```

\nNamei \nNameii \nNameiii \nNameiv \nNamev These are the names for numbers 1 to 5.

```
\newcommand*\nNamei{\iflowernumtoname o\else O\fi ne}
\newcommand*\nNameii{\iflowernumtoname t\else T\fi wo}
\newcommand*\nNameiii{\iflowernumtoname t\else T\fi hree}
\newcommand*\nNameiv{\iflowernumtoname f\else F\fi our}
\newcommand*\nNamev{\iflowernumtoname f\else F\fi ive}
```

\nNamevi \nNamevii \nNameviii \nNameix \nNamex

These are the names for numbers 6 to 10.

```
\newcommand*\nNamevi{\iflowernumtoname s\else S\fi ix}
\newcommand*\nNamevii{\iflowernumtoname s\else S\fi even}
\newcommand*\nNameviii{\iflowernumtoname e\else E\fi ight}
\newcommand*\nNameix{\iflowernumtoname n\else N\fi ine}
\newcommand*\nNamex{\iflowernumtoname t\else T\fi en}
```

\nNamexi \nNamexii \nNamexiii \nNamexiv \nNamexv

These are the names for numbers 11 to 15.

```
\newcommand*\nNamexi{\iflowernumtoname e\else E\fi leven}
\newcommand*\nNamexii{\iflowernumtoname t\else T\fi welve}
\newcommand*\nNamexiii{\iflowernumtoname t\else T\fi hir\teenstring}
\newcommand*\nNamexiv{\iflowernumtoname f\else F\fi our\teenstring}
\newcommand*\nNamexv{\iflowernumtoname f\else F\fi if\teenstring}
```

\nNamexvi \nNamexvii \nNamexviii \nNamexix \nNamexx

These are the names for numbers 16 to 20.

```
\newcommand*\nNamexvi{\iflowernumtoname s\else S\fi ix\teenstring}
\newcommand*\nNamexvii{\iflowernumtoname s\else S\fi even\teenstring}
\newcommand*\nNamexviii{\iflowernumtoname e\else E\fi igh\teenstring}
\newcommand*\nNamexix{\iflowernumtoname n\else N\fi ine\teenstring}
\newcommand*\nNamexx{\iflowernumtoname t\else T\fi wenty}
```

\nNamexxx \nNamexl \nNamel \nNamelx \nNamelxx

These are the names for numbers 30 to 70.

```
\newcommand*\nNamexxx{\iflowernumtoname t\else T\fi hirty}
\newcommand*\nNamexl{\iflowernumtoname f\else F\fi orty}
\newcommand*\nNamel{\iflowernumtoname f\else F\fi ifty}
\newcommand*\nNamelx{\iflowernumtoname s\else S\fi ixty}
\newcommand*\nNamelxx{\iflowernumtoname s\else S\fi eventy}
```

\nNamelxxx \nNamexc

These are the names for numbers 80 to 90.

```
\newcommand*\nNamelxxx{\iflowernumtoname e\else E\fi ighty}
\newcommand*\nNamexc{\iflowernumtoname n\else N\fi inety}
```

\unitnumbername Get the name of a unit (0 – 9).

```
\newcommand*{\unitnumbername}[1]{%
  \ifcase #1 \nNameo\or
   \nNamei\or
   \nNameii\or
   \nNameiii\or
   \nNameiv\or
   \nNamev\or
   \nNamevi\or
   \nNamevii\or
   \nNameviii\or
   \nNameix\fi}
```

\teennumbername Get the name of a 'teen number (10 – 19)

```
\newcommand*{\teennumbername}[1]{%
```

```
  \ifcase #1 \nNamex\or
   \nNamexi\or
   \nNamexii\or
   \nNamexiii\or
   \nNamexiv\or
   \nNamexv\or
   \nNamexvi\or
   \nNamexvii\or
   \nNamexviii\or
   \nNamexix\fi}

```

`\tensnumbername` Get the name of a tens number ($20 - 90$)

```
\newcommand*{\tensnumbername}[2]{%
  \ifnum #1=\@ne
    \teennumbername{#2}\ifnotnumtonameallcaps\lowernumtonametrue\fi
  \else
    \ifcase #1
    \or
    \or \nNamexx
    \or \nNamexxx
    \or \nNamexl
    \or \nNamel
    \or \nNamelx
    \or \nNamelxx
    \or \nNamelxxx
    \or \nNamexc
    \fi
    \ifnotnumtonameallcaps\lowernumtonametrue\fi
    \ifnum #2 > \z@ \tensunitsep\unitnumbername{#2}\fi
  \fi}

```

Names of small ordinals. The use of `\nthstring` instead of 'th' saves some tokens.

```
\newcommand*\nthNameo{\nNameo\nthstring}
\newcommand*\nthNamei{\iflowernumtoname f\else F\fi irst}
\newcommand*\nthNameii{\iflowernumtoname s\else S\fi econd}
\newcommand*\nthNameiii{\iflowernumtoname t\else T\fi hird}
\newcommand*\nthNameiv{\nNameiv\nthstring}
\newcommand*\nthNamev{\iflowernumtoname f\else F\fi if\nthstring}
\newcommand*\nthNamevi{\nNamevi\nthstring}
\newcommand*\nthNamevii{\nNamevii\nthstring}
\newcommand*\nthNameviii{\iflowernumtoname e\else E\fi igh\nthstring}
\newcommand*\nthNameix{\iflowernumtoname n\else N\fi in\nthstring}
\newcommand*\nthNamexii{\iflowernumtoname t\else T\fi welf\nthstring}

```

`\unitordinalname` Get the ordinal name of a unit ($0 - 9$)

```
\newcommand*{\unitordinalname}[1]{%
  \ifcase #1 \nthNameo\or
  \nthNamei\or
  \nthNameii\or
  \nthNameiii\or
  \nthNameiv\or
  \nthNamev\or
  \nthNamevi\or
  \nthNamevii\or
  \nthNameviii\or
  \nthNameix\fi}

```

\teenordinalname Get the ordinal name of a 'teen number (10 – 19). Using \nthstring instead of 'th' to save some tokens.

```
\newcommand*{\teenordinalname}[1]{%
  \ifcase #1 \nNamex\nthstring\or
  \nNamexi\nthstring\or
  \nthNamexii\or
  \nNamexiii\nthstring\or
  \nNamexiv\nthstring\or
  \nNamexv\nthstring\or
  \nNamexvi\nthstring\or
  \nNamexvii\nthstring\or
  \nNamexviii\nthstring\or
  \nNamexix\nthstring\fi}

```

\tensordinalname Get the ordinal name of a tens number (20 – 90) (Mathew Dafilis (mpd@swin.edu.au) sent Email on 2003/11/14 saying that \ordinaltoname didn't work for 20, 30, etc. He was correct. It is now fixed.

```
\newcommand*{\tensordinalname}[2]{%
  \ifnum #1=\@ne
    \teenordinalname{#2}\ifnotnumtonameallcaps\lowernumtonametrue\fi
  \else
    \ifnum #2> \z@
      \ifcase #1
      \or
      \or \nNamexx
      \or \nNamexxx
      \or \nNamexl
      \or \nNamel
      \or \nNamelx
      \or \nNamelxx
      \or \nNamelxxx
      \or \nNamexc
      \fi
      \ifnotnumtonameallcaps\lowernumtonametrue\fi
      \tensunitsep\unitordinalname{#2}
```

```
    \else
      \ifcase #1
      \or
      \or \nthNamexx
      \or \nthNamexxx
      \or \nthNamexl
      \or \nthNamel
      \or \nthNamelx
      \or \nthNamelxx
      \or \nthNamelxxx
      \or \nthNamexc
      \fi
      \ifnotnumtonameallcaps\lowernumtonametrue\fi
    \fi
  \fi}

```

The names of tens ordinals. The use of `\tiethstring` instead of 'tieth' saves somes tokens.

```
\newcommand*\nthNamexx{\iflowernumtoname t\else T\fi wen\tiethstring}
\newcommand*\nthNamexxx{\iflowernumtoname t\else T\fi hir\tiethstring}
\newcommand*\nthNamexl{\iflowernumtoname f\else F\fi or\tiethstring}
\newcommand*\nthNamel{\iflowernumtoname f\else F\fi if\tiethstring}
\newcommand*\nthNamelx{\iflowernumtoname s\else S\fi ix\tiethstring}
\newcommand*\nthNamelxx{\iflowernumtoname s\else S\fi even\tiethstring}
\newcommand*\nthNamelxxx{\iflowernumtoname e\else E\fi igh\tiethstring}
\newcommand*\nthNamexc{\iflowernumtoname n\else N\fi ine\tiethstring}

```

`\n@me@number` `\n@me@number{`⟨*number*⟩`}` is an internal macro to convert a ⟨*number*⟩ to names.

```
\newcommand*{\n@me@number}[1]{%
\begingroup
    \numdigits{#1}%
    \ifminusnumber\minusname\fi
    \priornumfalse
%% billions
    \ifnum \c@bsm@mctr > \z@
      \unitnumbername{\thebsm@mctr}\space
      \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamemmm
      \priornumtrue
    \fi
%% hundred millions
    \ifnum \c@cmsm@mctr > \z@
      \ifpriornum\namenumbercomma\fi
      \unitnumbername{\thecmsm@mctr}\space
      \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
      \priornumtrue
    \fi
%% tens/units millions
    \ifnum \c@xmsm@mctr > \z@
```

```
      \ifpriornum
        \ifnum\c@cmsm@mctr>\z@\namenumberand\else\namenumbercomma\fi
      \fi
      \tensnumbername{\thexmsm@mctr}{\themsm@mctr}%
      \priornumtrue
    \else
      \ifnum \c@msm@mctr > \z@
        \ifpriornum
          \ifnum\c@cmsm@mctr>\z@\namenumberand\else\namenumbercomma\fi
        \fi
        \unitnumbername{\themsm@mctr}%
        \ifnotnumtonameallcaps\lowernumtonametrue\fi
        \priornumtrue
      \fi
    \fi
    \ifnum \c@cmsm@mctr > \z@
      \ifpriornum\space\fi
      \nNamemm
    \else
      \ifnum \c@xmsm@mctr > \z@
        \ifpriornum\space\fi
        \nNamemm
      \else
        \ifnum \c@msm@mctr > \z@
          \ifpriornum\space\fi
          \nNamemm
        \fi
      \fi
    \fi
%% hundred thousands
    \ifnum \c@cksm@mctr > \z@
      \ifpriornum\namenumbercomma\fi
      \unitnumbername{\thecksm@mctr}\space
        \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
      \priornumtrue
    \fi
%% tens/units thousands
    \ifnum \c@xksm@mctr > \z@
      \ifpriornum
        \ifnum\c@cksm@mctr>\z@\namenumberand\else\namenumbercomma\fi
      \fi
      \tensnumbername{\thexksm@mctr}{\theksm@mctr}%
      \priornumtrue
    \else
      \ifnum \c@ksm@mctr > \z@
        \ifpriornum
          \ifnum\c@cksm@mctr>\z@\namenumberand\else\namenumbercomma\fi
        \fi
        \unitnumbername{\theksm@mctr}%
        \ifnotnumtonameallcaps\lowernumtonametrue\fi
```

```
        \priornumtrue
      \fi
    \fi
    \ifnum \c@cksm@mctr > \z@
      \ifpriornum\space\fi
      \nNamem
    \else
      \ifnum \c@xksm@mctr > \z@
        \ifpriornum\space\fi
        \nNamem
      \else
        \ifnum \c@ksm@mctr > \z@
          \ifpriornum\space\fi
          \nNamem
        \fi
      \fi
    \fi
%% hundreds
    \ifnum \c@csm@mctr > \z@
      \ifpriornum\namenumbercomma\fi
      \unitnumbername{\thecsm@mctr}\space
        \ifnotnumtonameallcaps\lowernumtonametrue\fi\nNamec
      \priornumtrue
    \fi
%% tens/units
  \ifmakeordinal
    \ifnum \c@xsm@mctr > \z@
      \ifpriornum\namenumberand\fi
      \tensordinalname{\thexsm@mctr}{\theism@mctr}%
    \else
      \ifnum \c@ism@mctr > \z@
        \ifpriornum\namenumberand\fi
        \unitordinalname{\theism@mctr}%
      \else
        \ifpriornum\nthstring\else\unitordinalname{\theism@mctr}\fi
      \fi
    \fi
  \else  % not ordinal
    \ifnum \c@xsm@mctr > \z@
      \ifpriornum\namenumberand\fi
      \tensnumbername{\thexsm@mctr}{\theism@mctr}%
    \else
      \ifnum \c@ism@mctr > \z@
        \ifpriornum\namenumberand\fi
        \unitnumbername{\theism@mctr}%
      \else
        \ifpriornum\else\unitnumbername{\theism@mctr}\fi
      \fi
    \fi
  \fi % end ifmakeordinal
```

```
\endgroup}
```

`\numtoname` Lowercase all names

```
\DeclareRobustCommand{\numtoname}[1]{%
  \makeordinalfalse
  \notnumtonameallcapstrue%
  \lowernumtonametrue%
  \n@me@number{#1}}
```

`\numtoName` Uppercase first letter of first name (all else lowercase).

```
\DeclareRobustCommand{\numtoName}[1]{%
  \makeordinalfalse
  \notnumtonameallcapstrue%
  \lowernumtonamefalse%
  \n@me@number{#1}}
```

`\NumToName` Uppercase first letter of all names (all else lowercase).

```
\DeclareRobustCommand{\NumToName}[1]{%
  \makeordinalfalse
  \notnumtonameallcapsfalse%
  \lowernumtonamefalse%
  \n@me@number{#1}}
```

`\ordinaltoname` Lowercase all ordinal names

```
\DeclareRobustCommand{\ordinaltoname}[1]{%
  \makeordinaltrue
  \notnumtonameallcapstrue%
  \lowernumtonametrue%
  \n@me@number{#1}}
```

`\ordinaltoName` Uppercase first letter of first ordinal name (all else lowercase).

```
\DeclareRobustCommand{\ordinaltoName}[1]{%
  \makeordinaltrue
  \notnumtonameallcapstrue%
  \lowernumtonamefalse%
  \n@me@number{#1}}
```

`\OrdinalToName` Uppercase first letter of all ordinal names (all else lowercase).

```
\DeclareRobustCommand{\OrdinalToName}[1]{%
  \makeordinaltrue
  \notnumtonameallcapsfalse%
  \lowernumtonamefalse%
  \n@me@number{#1}}
```

31.14 A fix for two column headings

This is from a posting by Donald Arseneau to CTT on 23 April 2001 to fix a problem — 'When I use \onecolumn\chapter... its headline is printed lower on the page than for two-column chapters'
DA and I had previously discussed this in relation to the Index and I had put a hack into the tocbibind package to fix the Index.
Donald posted the following.
\vspace* gives bad spacing after a pagebreak and \@makechapterhead starts with \vspace*. The biggest problem is the definition of \@topnewpage, which is used for the two-column spanning text. Here is a redefinition:

\@topnewpage

```
\long\def \@topnewpage [#1]{%
  \@nodocument
  \@next\@currbox\@freelist{}{}%
  \global \setbox\@currbox
    \vbox {%
      \break
      \prevdepth\z@
      \begingroup
      \normalcolor
      \hsize\textwidth
      \@parboxrestore
      \col@number \@ne
      #1%
      \vskip -\dbltextfloatsep
      \endgroup
      \null % ordinary \baselineskip
      \vskip -\topskip
  }%
  \begingroup %% \showbox\@currbox
    \splitmaxdepth\maxdepth \splittopskip\topskip
    \setbox\@tempboxa \vsplit\@currbox to \z@
  \endgroup %% \showbox\@currbox
  \ifdim \ht\@currbox>\textheight
    \ht\@currbox \textheight
  \fi
  \global \count\@currbox \tw@
  \@tempdima -\ht\@currbox
  \advance \@tempdima -\dbltextfloatsep
  \global \advance \@colht \@tempdima
  \ifx \@dbltoplist \@empty
  \else
    \@latexerr{Float(s) lost}\@ehb
    \let \@dbltoplist \@empty
  \fi
  \@cons \@dbltoplist \@currbox
  \global \@dbltopnum \m@ne
```

```
\ifdim \@colht<2.5\baselineskip
  \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
      too tall on page \thepage}%
  \@emptycol
  \if@firstcolumn
  \else
    \@emptycol
  \fi
\else
  \global \vsize \@colht
  \global \@colroom \@colht
  \@floatplacement
\fi}
```

The original version of `\@topnewpage` is in `ltouput.dtx`, line 159.

31.15 Time of day

William Adams (2006/08/28) supplied a basis for `\printime`, which he needed for `\quarkmarks` but I have used one from *TeX for the Impatient* as it saves some counters.

`\m@mcalchm` Calculate the hours and minutes from `\time`.

```
\newcommand*{\m@mcalchm}{%
  \count0 = \time \divide \count0 by 60\relax
  \count2 = \count0\relax%    the hour
  \count4 = \time \multiply\count0 by 60\relax
  \advance\count4 by -\count0\relax% the minute
  \ifnum\count4<10 \toks1 = {0}%  make a leading zero
  \else \toks1 = {}%
  \fi}
```

`\hmpunct` `\amname` `\pmname` User format controls for `\printtime`

```
%%% punctuation, am and pm for \printtime
\newcommand*{\hmpunct}{:}% hours minutes separator
\newcommand*{\amname}{am}% ante meridiem
\newcommand*{\pmname}{pm}% post meridiem
```

`\printtime` `\printtime*` Print the time of day as 24 hour clock or 12 hour clock `\printtime` prints the time par 24 hour clock and `\printtime*` per 12 hour clock.

```
\newcommand*{\printtime}{%
  \@ifstar{\m@msprtime}{\m@mprtime}}
```

`\m@mprtime` `\m@msprtime` These implement time printing: `\m@mprtime` as 24 hour clock and `\m@msprtime` as 12 hour clock

```
\newcommand*{\m@mprtime}{\begingroup
  \m@mcalchm
```

```
  \number\count2\hmpunct\the\toks1 \number\count4
  \endgroup}
\newcommand*{\m@msprtime}{\begingroup
  \m@mcalchm
  \def\@mpm{\pmname}%
  \ifnum\count2<1\relax% early in the morning
    \count2=12\relax
    \ifnum\count4>0\relax% not midnight
      \def\@mpm{\amname}%
    \fi
  \else
    \ifnum\time<721\relax% noon or earlier
      \def\@mpm{\amname}%
    \else
      \ifnum\time>779\relax% 1300 hrs or later
        \advance\count2 by -12\relax
      \fi
    \fi
  \fi
  \number\count2\hmpunct\the\toks1 \number\count4\ \@mpm
  \endgroup}

```

31.16 Sequential sheet (page) numbers

`\c@sheetsequence` Peter Heslin asked for the ability to add a sequential page number (1 for the first page, N for the last page, no matter what value the page counter has) to the trimming marks.
`\c@sheetsequence` is a new counter for pages starting at the beginning and independent of the standard page counter. This should not be reset by anything. The counter increment has to be added to the output routine.
This may also be useful for page N of M numbering.

```
\newcounter{sheetsequence}
  \setcounter{sheetsequence}{1}
  \renewcommand{\thesheetsequence}{\@arabic\c@sheetsequence}
\g@addto@macro{\@outputpage}{\stepcounter{sheetsequence}}

```

`\c@lastsheet` `\c@lastpage` While we're at it, might as well provide for lastpage and lastsheet counters

```
\newcounter{lastsheet}
  \setcounter{lastsheet}{0}
\newcounter{lastpage}
  \setcounter{lastpage}{0}

```

`\dol@stsheet` `\dol@stpage` These two macros write the values of lastsheet and lastpage to the `aux` file. They have to be called at the end of the document after a `\clearpage` to flush out any floats.

```
\newcommand{\dol@stsheet}{%
```

```
  \if@filesw
    \addtocounter{sheetsequence}{-1}%
    \immediate\write\@auxout%
      {\string\memsetcounter{lastsheet}{\the\c@sheetsequence}}%
    \stepcounter{sheetsequence}%
  \fi}
\newcommand{\dol@stpage}{%
  \if@filesw
    \addtocounter{page}{-1}%
    \immediate\write\@auxout%
      {\string\memsetcounter{lastpage}{\the\c@page}}%
    \stepcounter{page}%
  \fi}
```

I originally used this: `\AtEndDocument{\clearpage\dol@stsheet\dol@stpage}` but following the CTT thread *AtEndDocument produces unwanted page break*, 2004/03/11, and in particular Dan Luecking's response I now try as hard as possible to do it right at the end, but even this is not 100% reliable. The only way to ensure reliability is to modify the kernel `\enddocument` which doesn't seem to be a particularly wise thing to do.

```
\AtBeginDocument{\AtEndDocument{\clearpage\dol@stsheet\dol@stpage}}

```

31.17 Leaves per gathering

Traditionally books are assembled in terms of gatherings, or signatures, with perhaps 8 or 16 pages (or leaves) per grouping (a leaf has two pages, recto and one verso). At the request of Alan Ristow code for this has been included in the class.

`\ifcntrmod` `\ifnotcntrmod` `\iscntrmod` `\iscntrmod{`⟨*counter*⟩`}{`⟨*number*⟩`}` returns `\cntrmod(true|false)` and `\notcntrmod(false|true)` if ⟨*counter*⟩ is a multiple of ⟨*number*⟩.

```
\newif\ifcntrmod
\newif\ifnotcntrmod
\newcommand*{\iscntrmod}[2]{
  \@tempcnta=\@nameuse{c@#1}%
  \@tempcntb=\@tempcnta
  \divide\@tempcnta #2\relax
  \multiply\@tempcnta #2\relax
  \advance\@tempcntb-\@tempcnta
  \ifnum\@tempcntb=0\relax
    \cntrmodtrue
    \notcntrmodfalse
  \else
    \cntrmodfalse
    \notcntrmodtrue
  \fi}

```

\@memensuresigpages Ouput enough (empty) pages to make up a complete final signature.

```
\newcommand*{\@memensuresigpages}{%
  \ifnum\@mempagespersig<\@ne
  \else
    \iscntrmod{sheetsequence}{\@mempagespersig}
    \ifcntrmod
    \else
        \clearpage
        \pagestyle{empty}
        \mbox{}
      \loop
        \iscntrmod{sheetsequence}{\@mempagespersig}
        \ifnotcntrmod
        \clearpage
        \pagestyle{empty}
        \mbox{}
      \repeat
    \fi
  \fi}

```

\leavespergathering \leavespergathering{⟨*num*⟩} is the user command for specifying that there must be ⟨*num*⟩ leaves per gathering (2⟨*num*⟩ pages per gathering). For ⟨*num*⟩ more than one the total number of pages output is exactly divisible by 2⟨*num*⟩. ⟨*num*⟩ less than two (the default) has no effect.

```
\newcommand*{\leavespergathering}[1]{\@memcnta=#1\relax
                                     \ifnum\@memcnta<\tw@
                                       \def\@mempagespersig{-1}%
                                     \else
                                       \multiply\@memcnta \tw@
                                       \edef\@mempagespersig{\@memcnta}%
                                     \fi}
\leavespergathering{0}

```

Finally, make sure that the requested number of pages is output.

```
\AtEndDocument{\@memensuresigpages}

```

32 Initialization

32.1 Words and phrases

This document class is for documents prepared in the English language. To prepare a version for another language, various English words and phrases must be replaced. The English elements that require replacement are defined below in command names.

\abstractname This list is for titles of document sections.
\contentsname
\listfigurename
\listtablename
\bookname
\partname
\chaptername
\appendixname
\appendixtocname
\appendixpagename
\bibname
\indexname
\glossaryname

```
\newcommand*{\abstractname}{Abstract}
\newcommand*{\contentsname}{Contents}
\newcommand*{\listfigurename}{List of Figures}
\newcommand*{\listtablename}{List of Tables}
\newcommand*{\bookname}{Book}
\newcommand*{\partname}{Part}
\newcommand*{\chaptername}{Chapter}
\newcommand*{\appendixname}{Appendix}
\newcommand*{\appendixtocname}{Appendices}
\newcommand*{\appendixpagename}{Appendices}
\newcommand*{\bibname}{Bibliography}
\newcommand*{\indexname}{Index}
\newcommand*{\glossaryname}{Glossary}
```

\figurename These are the names and phrases used for general elements.
\tablename
\figurerefname
\tablerefname
\pagename
\pagrefename

```
\newcommand*{\figurename}{Figure}
\newcommand*{\tablename}{Table}
\newcommand*{\figurerefname}{Figure}
\newcommand*{\tablerefname}{Table}
\newcommand*{\pagename}{page}
\newcommand*{\pagerefname}{page}
```

\bookrefname More names for referencing.
\partrefname
\chapterrefname
\sectionrefname
\appendixrefname

```
\newcommand*{\bookrefname}{Book~}
\newcommand*{\partrefname}{Part~}
\newcommand*{\chapterrefname}{Chapter~}
\newcommand*{\sectionrefname}{\S}
\newcommand*{\appendixrefname}{Appendix~}
```

32.2 Date

\today This macro uses the TEX primitives \month, \day and \year to provide the date of the LATEX-run.

```
\newcommand{\today}{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
```

32.3 Two column mode

\columnsep This gives the distance between two columns in two column mode.

```
\setlength\columnsep{10\p@}
```

\columnseprule This gives the width of the rule between two columns in two column mode. We have no visible rule.

```
\setlength\columnseprule{0\p@}
```

32.4 The page style and counters

We use the page style *headings* by default and arabic page numbering.

```
\pagestyle{headings}
\pagenumbering{arabic}
```

We set the sectional counters to zero, the `tocdepth` to one (sections and above listed), the `secnumdepth` to two (sections and above numbered), and `\maxsecnumdepth` to the same.

```
\setcounter{part}{0}
\setcounter{chapter}{0}
\setcounter{tocdepth}{1}
\setcounter{secnumdepth}{2}
\maxsecnumdepth{section}
```

Set the `\linenumberfrequency` to zero to prohibit line numbering and also set the font for line numbers. Can now set the final space for boxed verbatim line numbers.

```
\linenumberfrequency{0}
\linenumberfont{\small\rmfamily}
\settowidth{\bvnumlength}{\vlvnumfont 9999}
```

32.5 Single or double sided printing

Unless the twoside wasn't specified, We do not try to make each page of equal height.

```
\if@twoside
\else
  \raggedbottom
\fi
```

When the twocolumn option was specified we call `\twocolumn` to activate this mode. We try to make each column as long as the others, but call `\sloppy` to make our life easier.

```
\if@twocolumn
  \twocolumn
  \sloppy
  \flushbottom
```

Normally we call `\onecolumn` to initiate typesetting in one column.

```
\else
  \onecolumn
\fi
```

32.6 Floats

Here are the implementations of the figure and table environments and their accompanying List of...

figure
figure*

This is the definition of the actual environment. The form with the * is used for double column figures. We use \newfloat to set it. In this class figures are numbered per chapter, but we need to change the default definition of \thefigure if a figure is in a pre-numbered chapter.

```
\newfloat[chapter]{figure}{lof}{\figurename}
%%  \kill@lastcounter{lofdepth}
\renewcommand{\thefigure}{\thechapter.\@arabic\c@figure}

```

\listoffigures
\listoffigures*

These macros request that LaTeX produces a list of figures. The LoF heading is added to the ToC unless the starred version is used.

```
\newlistof{listoffigures}{lof}{\listfigurename}
%%  \kill@lastcounter{lofdepth}
```

\l@figure

\l@figure{⟨*title*⟩}{⟨*page*⟩} typesets the LoF entry for a \figure caption heading.

```
\newlistentry[chapter]{figure}{lof}{0}
  \cftsetindents{figure}{0em}{2.3em}
%%  \kill@lastcounter{lofdepth}

```

table
table*

The definition for tables is almost identical.

```
\newfloat[chapter]{table}{lot}{\tablename}
%%  \kill@lastcounter{lotdepth}
\renewcommand{\thetable}{\thechapter.\@arabic\c@table}

```

\listoftables
\listoftables*

These macros request that LaTeX produces a list of tables. The LoT heading is added to the ToC unless the starred version is used.

```
\newlistof{listoftables}{lot}{\listtablename}
%%  \kill@lastcounter{lotdepth}
```

\l@table

\l@table{⟨*title*⟩}{⟨*page*⟩} typesets the LoT entry for a \table caption heading.

```
\newlistentry[chapter]{table}{lot}{0}
  \cftsetindents{table}{0em}{2.3em}
%%  \kill@lastcounter{lotdepth}

```

The subfigure package defines lofdepth and lotdepth counters. If is not used, then we have to define them. The subfig package replaced subfigure in 2005, and this only defined the counters if they were not previously defined. It now seems sensible to ignore any use of the subfigure package. This makes life a lot simpler.

```
%%%\AtBeginDocument{%
```

```
%%%  \@ifundefined{c@lofdepth}%
%%%     {\newcounter{lofdepth}\setcounter{lofdepth}{1}}{}
%%%  \@ifundefined{c@lotdepth}%
%%%     {\newcounter{lotdepth}\setcounter{lotdepth}{1}}{}}

```

32.7 The article option

The article option requires changes to the default chapterstyle, and the numbering of floats, etc.
Emanuele Vicentini (2003/07/21) suggested making `\maketitle` more closely match the real article's appearance. Alan Budden[31] commented on 2003/12/18 that the equation counter should be continuous.

```
\ifartopt
  \chapterstyle{article}
  \counterwithout{figure}{chapter}
  \counterwithout{table}{chapter}
  \counterwithout{footnote}{chapter}
  \counterwithout{equation}{chapter}
  \renewcommand{\chaptername}{}
  \renewcommand{\maketitlehookb}{%
    \vskip -1.5\topsep\vskip -1.5\partopsep}
  \renewcommand{\maketitlehookc}{%
    \vskip -1.5\topsep\vskip -1.5\partopsep}
\fi

```

32.8 The ms option

This should be done last as it makes various changes to the defaults.

`\msdoublespacing` `\mssinglespacing` These do nothing unless the ms option is used; then they change the `\baselinestretch`.

```
\newcommand{\msdoublespacing}{}
\newcommand{\mssinglespacing}{}

\ifmsdoc
  \renewcommand{\msdoublespacing}{%
    \renewcommand{\baselinestretch}{1.6}\large\normalsize}
  \renewcommand{\mssinglespacing}{%
    \renewcommand{\baselinestretch}{1.0}\large\normalsize}
  \renewcommand{\familydefault}{cmtt}
  \renewcommand{\rmdefault}{cmtt}
  \renewcommand{\sfdefault}{cmtt}
  \renewcommand{\bfdefault}{m}
  \renewcommand{\itdefault}{n}
  \renewcommand{\sldefault}{n}
```

[31] alan.s.budden@bristol.ac.uk

```
  \renewcommand{\scdefault}{n}
  \renewcommand{\baselinestretch}{1.6}
  \@twocolumnfalse
  \onecolumn
  \sloppy
  \@twosidefalse
  \raggedbottom
  \pagestyle{plain}
\fi

```

32.9 Emulated packages

Many of the 'emulations' are extensions and integration of package facilities. In some cases an 'emulated' package just won't work with the class, so I've added it here to prevent it from being loaded. In any case, most of the packages are mine.

```
\EmulatedPackage{abstract}[2008/07/23]
\EmulatedPackage{appendix}[2008/07/23]
\EmulatedPackage{array}[2008/07/23]
\EmulatedPackage{booktabs}[2008/07/23]
\EmulatedPackage{ccaption}[2008/07/23]
\EmulatedPackage{changepage}[2008/07/23]
\EmulatedPackage{chngcntr}[2008/07/23]
\EmulatedPackage{chngpage}[2008/07/23]
\EmulatedPackage{crop}
\EmulatedPackage{dcolumn}[2008/07/23]
\EmulatedPackage{delarray}[2008/07/23]
\EmulatedPackage{enumerate}[2008/07/23]
\EmulatedPackage{epigraph}[2008/07/23]
%%%%%\EmulatedPackage{framed}[2008/07/23]
\EmulatedPackage{ifmtarg}[2008/07/23]
\ifm@mifetex\EmulatedPackage{ifetex}[2008/07/23]\fi
\ifm@mifluatex\EmulatedPackage{ifluatex}[2008/07/23]\fi
\ifm@mifpdf\EmulatedPackage{ifpdf}[2008/07/23]\fi
\ifm@mifxetex\EmulatedPackage{ifxetex}[2008/07/23]\fi
\EmulatedPackage{index}[2008/07/23]
\EmulatedPackage{makeidx}[2008/07/23]
\EmulatedPackage{moreverb}[2008/07/23]
\EmulatedPackage{mparhack}[2008/07/23]
\EmulatedPackage{needspace}[2008/07/23]
\EmulatedPackage{newfile}[2008/07/23]
\EmulatedPackage{nextpage}[2008/07/23]
\EmulatedPackage{pagenote}[2008/07/23]
\EmulatedPackage{parskip}[2008/07/23]
\EmulatedPackage{patchcmd}[2008/07/23]
\EmulatedPackage{setspace}[2008/07/23]
\EmulatedPackage{shortvrb}[2008/07/23]
\EmulatedPackage{showidx}[2008/07/23]
\EmulatedPackage{tabularx}[2008/07/23]
```

```
\EmulatedPackage{titleref}[2008/07/23]
\EmulatedPackage{titling}[2008/07/23]
\EmulatedPackage{tocbibind}[2008/07/23]
\EmulatedPackage{tocloft}[2008/07/23]
\EmulatedPackage{tocvsec2}[2008/07/23]
\EmulatedPackage{verbatim}[2008/07/23]
\EmulatedPackage{verse}[2008/07/23]

```

32.10 Interaction with the caption package

Although the author of the caption package has, over the years, made it work with many other classes he has not extended it to either recognise or work with the memoir class. Some authors want to use the caption package, hence...
Kill changes to the caption macros when the caption package is used. The caption package checks the definitions of the `\@makecaption`, `\caption` and `\@caption` macros. These need to be identical to the definitions in the standard classes for the package to disbelieve that the memoir class is being used.

\@makecaption
\caption
\@caption

```
%%% revert changes to captioning macros if the caption package is used.
\AtBeginPackage{caption}{
\ClassWarningNoLine{memoir}{%
  You are using the caption package with the memoir \MessageBreak
  class. This may cause unexpected or inconsistent \MessageBreak
  results if you use any of memoir's captioning facilities}

\long\def\@makecaption##1##2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{##1: ##2}%
  \ifdim \wd\@tempboxa >\hsize
    ##1: ##2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}

\def\caption{%
   \ifx\@captype\@undefined
     \@latex@error{\noexpand\caption outside float}\@ehd
     \expandafter\@gobble
   \else
     \refstepcounter\@captype
     \expandafter\@firstofone
   \fi
   {\@dblarg{\@caption\@captype}}%
}

```

```
\long\def\@caption##1[##2]##3{%
  \par
  \addcontentsline{\csname ext@##1\endcsname}{##1}%
    {\protect\numberline{\csname the##1\endcsname}{\ignorespaces ##2}}%
  \begingroup
    \@parboxrestore
    \if@minipage
      \@setminipage
    \fi
    \normalsize
    \@makecaption{\csname fnum@##1\endcsname}{\ignorespaces ##3}\par
  \endgroup}
}

```

32.11 Interaction with the float package

The float package also defines `\newfloat`, so kill memoir's version when the float package gets used.

```
\AtBeginPackage{float}{\let\newfloat\relax}

```

32.12 Patch file

At the suggestion of Danie Els (`DanieEls@sun.ac.za`), Dan Leucking (`luecking@uark.edu`) and others, input a patch file, if one exists, as the final act. (This is preferable to my series of `memfixa.sty`, `memfixb.sty` packages, and so on, for each release of the class.)

```
\IfFileExists{mempatch.sty}{%
  \RequirePackage{mempatch}}{}

```

The end of the class definitions.

```
⟨/class⟩
```

33 Glossary Makeindex style file

Here is the basic style (configuration) file for Makeindex for use with the default glossary setup.

```
⟨*gst⟩
%%%%% basic.gst    basic makeindex glossary configuration file for memoir
%%%%% Output style parameters
preamble "\\begin{theglossary}"
postamble "\n\\end{theglossary}\n"
group_skip "\n\\glossaryspace\n"
item_0    "\n\\glossitem"
delim_0   "{\\memglonum{"
```

```
encap_suffix "}}}"
%%% Input style parameters
keyword "\\glossaryentry"

```
⟨/gst⟩

The end of the configuration file code

References

[ABH90] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. *TeX for the Impatient.* Addison-Wesley, Reading, Massachusetts, 1990. (Available from CTAN in `info/impatient`)

[Ars01a] Donald Arseneau. *Titleref package (version 3.1).* April 2001. (Available from CTN as `macros/latex/contrib/misc/titleref.sty`)

[Ars01b] Donald Arseneau. *Chapterbib package (version 1.9).* September 2001. (Available from CTN as `macros/latex/contrib/misc/chapterbib.sty`)

[Ars03] Donald Arseneau. *Framed package (version 0.8a).* July 2003. (Available from CTAN as `macros/latex/contrib/misc/framed.sty`)

[ArWi00] Donald Arseneau and Peter Wilson. *The ifmtarg package.* March, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)

[Car94] David Carlisle. *The delarray package.* March 1994. (Available from CTAN in `/macros/latex/required/tools`)

[Car98a] David Carlisle. *The enumerate package.* August, 1998. (Available from CTAN in `/macros/latex/required/tools`)

[Car98b] David Carlisle. *The remreset package.* August, 1998. (Available from CTAN in `/macros/latex/contrib/carlisle`)

[Car99] David Carlisle. *The tabularx package.* January 1999. (Available from CTAN in `/macros/latex/required/tools`)

[Car01] David Carlisle. *The dcolumn package.* May 2001. (Available from CTAN in `/macros/latex/required/tools`)

[Coc02] Steven Douglas Cochran. *The subfigure package.* March, 2002. (Available from CTAN in `/macros/latex/contrib/subfigure`)

[Dal99] Patrick W. Daly. *Natural Sciences Citations and References.* May, 1999. (Available from CTAN in `/macros/latex/contrib/natbib`)

[Dow00] Michael J. Downes *The patchcmd package.* July 2000. (Available from CTAN in `/macros/latex/contrib/patchcmd`)

[Fai98] Robin Fairbairns. *The moreverb package.* December, 1998. (Available from CTAN in `/macros/latex/contrib/moreverb`)

[Fai03] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in LaTeX.* February 2003. (Available from CTAN in `macros/latex/contrib/footmisc`)

[Fea03] Simon Fear. *Publication quality tables in LaTeX.* March, 2003. (Available from CTAN in `macros/latex/contrib/booktabs`)

[Fra00] Melchior Franz. *The crop package.* February, 2000. (Available from CTAN in `/macros/latex/contrib/crop`)

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

[Knu84] Donald E. Knuth. *The TeXbook.* Addison-Wesley, Reading, Massachusetts, 1984.

[Lam94] Leslie Lamport. *LaTeX — A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, 1994.

[LMB99] Leslie Lamport, Frank Mittelbach and Johannes Braams. *Standard Document Classes for LaTeX version 2e.* September, 1999. (Available from CTAN as `/macros/latex/base/classes.dtx`)

[MC98] Frank Mittelbach and David Carlisle. *A new implementation of LaTeX's tabular and array environment* May 1998. (Available from CTAN in `/macros/latex/required/tools`)

[Oos96] Piet van Oostrum. *Page layout in LaTeX.* June, 1996. (Available from CTAN in `/macros/latex/contrib/fancyhdr`)

[Rah01] Sebastian Rahtz. *Section name references in LaTeX.* January 2001. (Available from CTAN in `/macros/latex/contrib/hyperref`)

[Rah02] Sebastian Rahtz. *Hypertext marks in LaTeX.* March 2002. (Available from CTAN in `/macros/latex/contrib/hyperref`)

[Sch98] Martin Schröder. *The everyshi package.* August, 1998. (Available from CTAN in `/macros/latex/contrib/ms`)

[SRR01] Rainer Schöpf, Bernd Raichle and Chris Rowley. *A new implementation of LaTeX's verbatim and verbatim* environments.* March, 2001. (Available from CTAN in `/macros/latex/required/tools`)

[Wil99] Peter Wilson. *The tocvsec2 package*. January, 1999. (Available from CTAN in `/macros/latex/contrib/tocvsec2`)

[Wil00a] Peter Wilson. *The epigraph package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/epigraph`)

[Wil00b] Peter Wilson. *LaTeX files for typesetting ISO standards*. February, 2000. (Available from CTAN in `/macros/latex/contrib/isostds/iso`)

[Wil00c] Peter Wilson. *The nextpage package*. February, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)

[Wil00d] Peter Wilson. *The needspace package*. March, 2000. (Available from CTAN in `/macros/latex/contrib/misc`)

[Wil01a] Peter Wilson. *The abstract package*. February, 2001. (Available from CTAN in `/macros/latex/contrib/abstract`)

[Wil01b] Peter Wilson. *The chngpage package*. February, 2001. (Available from CTAN in `/macros/latex/contrib/misc`)

[Wil01c] Peter Wilson. *The appendix package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/appendix`)

[Wil01d] Peter Wilson. *The ccaption package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/ccaption`)

[Wil01e] Peter Wilson. *The chngcntr package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/misc`)

[Wil01f] Peter Wilson. *The hanging package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/hanging`)

[Wil01g] Peter Wilson. *The titling package*. March, 2001. (Available from CTAN in `/macros/latex/contrib/titling`)

[Wil01h] Peter Wilson. *The tocbibind package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocbibind`)

[Wil01i] Peter Wilson. *The tocloft package*. April, 2001. (Available from CTAN in `/macros/latex/contrib/tocloft`)

[Wil01j] Peter Wilson. *Typesetting simple verse with LaTeX*. August, 2001. (Available from CTAN in `/macros/latex/contrib/verse`)

[Wil03] Peter Wilson. *ledmac: A presumptuous attempt to port EDMAC and TABMAC to LaTeX*. August 2003. (Available from CTAN in `macros/latex/contrib/ledmac`)

[Wil07] Peter Wilson. 'Glisterings' *TUGboat*, 28(2):229–232, 2007.

[Wil08] Peter Wilson. 'Glisterings' *TUGboat*, 29(2):324–327, 2008.

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.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

Q

R

S

T

U

V

W

X

Z