The sectsty package v2.0.2

Rowland McDonnell
rowland.mcdonnell@physics.org

25th February 2002

Contents

1 Introduction

The secsty package provides a set of commands for changing the fount[1] used for the various sectional headings in the standard LaTeX 2_ε document classes: `article`, `book`, and `report`. This package also works with the KOMA-Script classes `scrartcl`, `scrbook`, and `scrreprt`.

When I refer to sectional headings in this document, I mean the stuff that is printed in the body of a document to indicate the start of a part, chapter, section, and so on. Appendix headings are treated as chapter headings.

I'd appreciate any bug reports, comments, or suggestions emailed to the address above. I'm especially interested in any problems you might have had understanding the documentation; and, of course, anything else you'd like to tell me about this package. I'm certainly interested in any problems you might have had in using sectsty with other packages, although I can't promise to be able to do anything useful about such problems.

The titlesec package offers a different approach to changing the appearance of sectional headings. The fncychap package has a set of alternative 'canned' chapter headings. Both are available from CTAN.

Donald Arseneau won't let me say what bits of sectsty he wrote, but I'd like to thank him anyway: I couldn't have managed without his help. Thanks are also due to the various people who sent me bug reports – I've not kept track so rather than mention some of you and miss others out, I'll just say thanks and leave it at that. You know who you are and I appreciate your help.

2 Basic use of sectsty

Make sure you include the package in your document by saying in your document preamble:

```
\usepackage{sectsty}
```

You will then have some new commands available. For example:

```
\allsectionsfont{\sffamily}
```

will give you sanserif for all sectional headings.

Sectsty doesn't make any changes to the fount used until you tell it what to do. The idea is that you supply some standard LaTeX fount selection commands which change the default selection. The standard classes use Computer Modern Bold Extended by default for sectional headings, and

[1] I am one of the few remaining English speaking people on the planet to use this spelling to refer to a set of type in one size and style.

your commands change this. This means the example above gives you Computer Modern Sanserif Bold Extended by default.

You might want to change all headings so that they used medium italic. Since the default headings use bold extended, you need to switch to the medium weight as well as switching to italic using the standard LaTeX fount selection commands, like this:

```
\allsectionsfont{\mdseries\itshape}
```

You can change the style of the different sectional headings separately. If you want `\section`s to be set in bold Adobe Times, you might use standard LaTeX fount selection commands like this:

```
\sectionfont{\fontfamily{ptm}\selectfont}
```

Similar commands exist to change the style of the rest of LaTeX's standard headings independently: `\chapterfont{`⟨*commands*⟩`}` will change chapter headings, and so on. The full list of commands is given in section 5 on page 5.

If you don't understand the fount selection commands above, you might like to look at the file `fntguide.tex` that comes with the standard LaTeX distribution.

3 Raggedright, raggedleft, and centred headings

The arguments to sectsty's sectional heading style changing commands can contain any LaTeX commands you like. For example, you might say:

```
\allsectionsfont{\sffamily\raggedright}
```

and all sectional headings would be typeset raggedright in bold extended sanserif. Not all LaTeX commands will work properly when used this like, although these three commands all have the expected effect when used in one of sectsty's `\`⟨*section*⟩`font` commands:

`\centering`	Headings centred on the page
`\raggedright`	Headings set flush left with a ragged right margin
`\raggedleft`	Headings set flush right with a ragged left margin

It's a bad idea to use any of these commands to change paragraph or subparagraph sectional headings: these two sections begin with 'run-in' headings that are on the same line as the start of the section's text, and don't like being messed around like this. You can change the indentation before these two sectional headings like this:

```
\subparagraphfont{\hspace*{5em}}
```

This adds an extra 5 em space before each subparagraph heading.

As a further example, you might want chapter headings set flush with the right margin. This will do the job:

```
\chapterfont{\raggedleft}
```

If you want to play around with this sort of thing, it might be useful to know what sort of positioning you get by default with the standard classes: all sectional headings except chapter and part are fully justified (text set flush with the left and right margins), and chapter headings are raggedright. Part headings vary: they are raggedright in the article class, and centred in the book and report classes.

3.1 A complication affecting multiple line headings

The standard `\section`, `\subsection`, and `\subsubsection` commands ensure that, if you've got a multiple line heading, lines beyond the first are typeset with a small amount of indentation. This is called hanging indentation, and is entirely appropriate with the standard formatting. However, two problems arise from this: firstly, this hanging indentation usually looks pretty stupid if you've decided to have centred headings; and secondly, due to a bug in LaTeX, the hanging indentation is suppressed if you use `\\` in a heading that's typeset raggedright or centred.

The first problem can be dealt with: if you use the `\nohang` command in the appropriate `\`⟨*section*⟩`font` command, the hanging indentation will be suppressed. For example:

```
\sectionfont{\nohang\centering}
```

will ensure that any multiple line section headings come out centred as expected.

The second problem – the suppression of hanging indentation when you use `\\` in a raggedright or centred heading – remains unsolved. I'm told that it's caused by the definition of `\@centercr`. This is part of the LaTeX kernel and beyond my abilities to patch up. This problem has been reported as a LaTeX bug, so it might well be sorted out in a future LaTeX release.

Donald Arseneau's ulem package makes its own arrangements in this area, so sectional headings that are underlined using sectsty and ulem have correct hanging indentation even if you do use `\\` to end a line early.

4 The KOMA-script classes

If you just want to change the fount used for all sectional headings, you shouldn't use sectsty with any of the KOMA-script classes. Instead, you

should redefine the `\sectfont` command provided by the KOMA-script classes. This is akin to using sectsty's `\allsectionsfont` command.

For example, the default definition is:

```
\newcommand*\sectfont{\sffamily\bfseries}
```

you might want to change from this bold sanserif to medium italic roman. To do this, you could add the following line to your document preamble:

```
\renewcommand*\sectfont{\rmfamily\mdseries\itshape}
```

If you'd like different sectional headings to be printed with different styles of type to each other, or if you'd like to underline sectional headings or play other games that you can't do with the KOMA-script `\sectfont` command, then sectsty might be of use with the KOMA-script classes.

You should note that the modifications applied by sectsty commands happen immediately after the `\sectfont` command is executed.

The documentation for this package is written with the standard LaTeX classes in mind, so might not tie up exactly to the KOMA-script classes. Despite that, and despite differences in behaviour when things go wrong, sectsty should work as expected with the KOMA-script classes.

5 All the commands

The full list of commands is:

`\allsectionsfont{`⟨*commands*⟩`}` Changes the style of all sectional headings by executing `{`⟨*commands*⟩`}` before printing each heading.

`\partfont{`⟨*commands*⟩`}` Changes the style of 'part' headings only by executing `{`⟨*commands*⟩`}` before printing each heading; this affects both the title of the part and the part number.

`\chapterfont{`⟨*commands*⟩`}` Changes the style of 'chapter' headings only by executing `{`⟨*commands*⟩`}` before printing each heading; this affects both the title of the chapter and the chapter number.

`\sectionfont{`⟨*commands*⟩`}` Changes the style of 'section' headings only by executing `{`⟨*commands*⟩`}` before printing each heading.

`\subsectionfont{`⟨*commands*⟩`}` Changes the style of 'subsection' headings only by executing `{`⟨*commands*⟩`}` before printing each heading.

`\subsubsectionfont{`⟨*commands*⟩`}` Changes the style of 'subsubsection' headings only by executing `{`⟨*commands*⟩`}` before printing each heading.

`\paragraphfont{`⟨*commands*⟩`}` Changes the style of 'paragraph' headings only by executing `{`⟨*commands*⟩`}` before printing each heading; you should not use text positioning commands for this heading.

`\subparagraphfont{`⟨*commands*⟩`}` Changes the style of the 'subparagraph' heading only by executing `{`⟨*commands*⟩`}` before printing each heading; you should not use text positioning commands for this heading.

`\minisecfont{`⟨*commands*⟩`}`]Changes the style of the 'minisec' heading only by executing `{`⟨*commands*⟩`}` before printing each heading. This command always works, but since these headings only exist in the KOMA-script classes, it's useless with the standard LaTeX classes.

There are also these commands:

`\partnumberfont{`⟨*commands*⟩`}` Changes the style of 'part' heading numbers only; this does not affect the title of the part heading.

`\parttitlefont{`⟨*commands*⟩`}` Changes the style of 'part' heading titles only; this does not affect the number of the part heading.

`\chapternumberfont{`⟨*commands*⟩`}` Changes the style of 'chapter' heading numbers only; this does not affect the title of the chapter heading.

`\chaptertitlefont{`⟨*commands*⟩`}` Changes the style of 'chapter' heading titles only; this does not affect the number of the chapter heading.

And finally, an anomalous helper command:

`\nohang` For use in multiple-line headings: this command stops lines beyond the first having hanging indentation, which looks pretty daft with centred headings.

6 Some notes

The way sectsty works is by re-defining all the commands that produce the sectional headings. The change arranges things so that the fount selection commands you specify are executed immediately before the sectional heading is printed. You can change sizes, for example, with the normal LaTeX size changing commands.

You can include almost any LaTeX commands in the argument to one of sectsty's commands. Whether any particular command will work properly or have the effect you want is a different matter. If you come across anything interesting you can do with this package that I've not mentioned and you think someone else might be interested in, you could let me know and I'll put it in the documentation.

If you use long section names you might like to make all sectional headings `\raggedright`:

```
\allsectionsfont{\raggedright}
```

This means that LaTeX won't try and justify sectional headings and should manage to avoid a hyphentated sectional heading which tends to look pretty bad.

If you use sectsty and get a complaint that you are using an old version of LaTeX, there's probably no need to worry. Sectsty will probably work correctly with any version of LaTeX 2_ε after June 1996; because the earliest version I've tested sectsty with is the June 1998 release of LaTeX, that's the version I've specified as being needed. Sectsty is not meant to work with LaTeX 2.09; it's a good idea to get LaTeX 2_ε if at all possible.

7 Underlining

Some people have to underline sectional headings because of regulations governing thesis submission and things like that. If you're not forced to use underlining, it's probably best to avoid it: underlining is usually considered bad typographical practice (well-placed horizontal rules are a different matter). On top of that, underlining sectional headings with sectsty is not as trivial a job as I'd like – you've either got to get another package or accept quite a lot of limitations.

To underline a sectional heading with sectsty, you need to put the `\underline` command as the last command in a `\`⟨*section*⟩`font` command. For example:

```
\usepackage{sectsty}
\allsectionsfont{\sffamily\underline}
```

will give you underlined sanserif headings.

There are two main problems with this way of doing things: firstly, the standard LaTeX `\underline` command produces a one line box in LR mode that can't be split across lines. This means that long headings can't be split into two or more lines and will always result in overfull `\hbox`es. Secondly, the underline rule is placed underneath the lowest extent of the text and affects vertical spacing. This is particularly ugly in the case of `\paragraph` and `\subparagraph` headings which are on the same line as the first line of text of the section.

Both these problems can be avoided by using the ulem package (see section 7.1.2 on page 11 for how to get it):

```
\documentclass{article}
\usepackage{sectsty}
```

```
\usepackage[normalem]{ulem}
\allsectionsfont{\sffamily\underline}
```

Don't forget that the `\underline` command must always be the last item in the `\`⟨*section*⟩`font` command. This example uses the familiar `\underline` command as before, but because ulem has been loaded, sectsty automatically redefines `\underline` inside sectional headings to use ulem's underline code. The `\underline` command has its normal meaning outside sectional headings, so you won't get any nasty surprises.

I prefer the results you get using the ulem package, but you might prefer the appearance of underlined sectional headings produced without ulem's help. If so, and you're not using `\paragraph` or `\subparagraph` sections, or section headings that need to be split over two lines, there's no need to use ulem.

Sectsty always plays some dirty tricks which involve re-defining the `\underline` command inside sectional headings only. These tricks work differently if ulem has been loaded, but it doesn't matter whether you load ulem before or after sectsty.

Despite these fun and games, the `\underline` command always has its usual definition outside sectional headings and will work normally in the rest of your document.

7.1 A detailed look at underlining

It is possible to underline sectional headings in other ways. This section explains how, and also explains a little more about underlining sectional headings with sectsty.

The secsty package lets you use two different commands to underline sectional headings: `\underline` and `\ulemheading`. Both commands suffer from the same restriction in use: they must always be the last command in the argument to the `\`⟨*section*⟩`font` command. Both commands can be used whether or not you've loaded ulem: they are defined differently depending on whether or not the ulem package has been loaded. The decision on which definition to use is made at the `\begin{document}` command, so you can load ulem before or after sectsty.

If you want to use underlined headings, I strongly suggest you get the ulem package as well. Without the ulem package, you get identical results (with a few problems) in sectional headings using both `\underline` and `\ulemheading`.

Sectsty re-defines the `\underline` command inside sectional headings. This re-definition is what allows underlining to work correctly in all of the standard sectional headings, and allows the same command to behave differently if you've use the ulem package. TEX's usual grouping rules mean that the `\underline` command is restored to its usual meaning at the end

of each sectional heading: the re-definition used by sectsty should have no effect on the rest of your document.

Assuming for the moment that you're not using ulem, you can get a straightforward single underline using the normal LaTeX underline command like this:

```
\allsectionsfont{\underline}
```

This has a few problems. Most noticeably, it won't allow sectional headings to break at the end of the line, so long headings will extend beyond the right hand edge of the text. This is because the standard LaTeX `\underline` command puts its argument into a box made in LR mode, so headings underlined this way will be one line headings no matter what you do.

Another problem is that the underline rule used sits below the lowest possible extent of the text it's underlining, and so increases the space between the underlined heading and the text below. This is especially noticeable if you're using `\paragraph` or `\subparagraph` headings. And remember that the `\underline` command must be the last in this list, like this:

```
\allsectionsfont{\sffamily\underline}
```

If you put `\sffamily` (or anything else) after `\underline`, it won't work properly.

7.1.1 Underlining with the ulem package

The ulem package has some nice commands for improved underlining in normal text. The main advantages are that the underline rule is positioned better (so it doesn't mess up the line spacing), and it allows long underlined headings to break normally at the end of the line. All you need to do is load ulem, and sectsty will do the rest. Try the following example as it is here:

```
\documentclass{article}
\usepackage{sectsty}
\usepackage[normalem]{ulem}
\allsectionsfont{\sffamily\raggedright\underline}
\begin{document}
\section{A very long sectional heading title that will
         have to be split over two lines}
\end{document}
```

You'll notice that the long section heading was split over two lines. Now try this:

```
\documentclass{article}
```

```
\usepackage{sectsty}
\allsectionsfont{\sffamily\raggedright\underline}
\begin{document}
\section{A very long sectional heading title that will
         have to be split over two lines}
\end{document}
```

You'll notice that the underlining rule is positioned differently, and the heading could not be split over two lines.

If you want to use in sectional headings some of the other types of underlining allowed by the ulem package, you can use the `\ulemheading` command, as in this example to give double underlining and raggedright sectional headings:

```
\usepackage{sectsty,ulem}
\allsectionsfont{\raggedright\ulemheading{\uuline}}
```

The `\ulemheading` command is meant to be used only inside sectional headings like this. It takes two arguments: the first argument is the ulem package command to generate a particular style of underlining; the second argument is the text to be underlined, and is provided by the command that generates the sectional heading. All you have to do is provide the first argument and ensure that you put the `\ulemheading` at the end of the `\`⟨*section*⟩`font` command's argument, as in this example to give you wavy underlined sans-serif headings:

```
\usepackage{sectsty,ulem}
\allsectionsfont{\sffamily\ulemheading{\uwave}}
```

The full list of modifications allowed by the `\ulemheading` command is:

`\ulemheading{\uline}`	single underline
`\ulemheading{\uuline}`	double underline
`\ulemheading{\uwave}`	wavy underline
`\ulemheading{\sout}`	strike out with `-`
`\ulemheading{\xout}`	cross out with `/`

It's possible to define more by following the directions in the ulem package.

If you typeset a document that uses `\ulemheading` on a system where the ulem package is unavailable, the document will be processed but the output will be different. If you've not loaded the ulem package, `\ulemheading` will produce underlining exactly the same as if you'd used the `\underline` command: `\ulemheading` is defined to ignore the first argument and pass the heading text to the normal underlining code. This means you can produce a document using ulem's fancy underlining, and process it on a LaTeX system without ulem by making a single modification: commenting out the `\usepackage{ulem}` command. The results will of course be different, but the document will typeset.

7.1.2 More notes on ulem and underlining

The ulem package is available via anonymous ftp from the Comprehensive TEX Archive Network (CTAN). You can find a list of CTAN sites and a Web interface to CTAN at `http://www.tug.org/ctan.html`. The following ftp urls work now (August 1998):

```
ftp://ftp.dante.de/tex-archive/macros/latex/contrib/other/misc/ulem.sty
ftp://ftp.tex.ac.uk/tex-archive/macros/latex/contrib/other/misc/ulem.sty
ftp://ctan.tug.org/tex-archive/macros/latex/contrib/other/misc/ulem.sty
```

The ulem package was written by Donald Arseneau, who contributed some code and lots of useful advice to sectsty.

It doesn't matter if you load ulem before or after sectsty, because sectsty waits until the `\begin{document}` command before checking to see if ulem has been loaded.

Sectsty uses a cunning trick to re-define the `\underline` command inside sectional headings only; `\underline` will work normally outside sectional headings. The trick is cunning because the re-definition is different if the ulem package is in use in that document: the commands that make this trick work wait until the `\begin{document}` command has been executed, so it doesn't matter if you load ulem before or after sectsty.

The `\underline` and `\ulemheading` commands produce identical results if the ulem package is not loaded; they will both use LaTeX's normal underlining mechanism. If you do load ulem, `\underline` will produce nicer results, and `\ulemheading` will allow you to use any of ulem's different underlining styles.

The ulem package changes LaTeX's emphasis commands so that they use underlining rather than italics for emphasis. If you don't want this, you should load ulem with the `normalem` option:

```
\usepackage[normalem]{ulem}
```

8 Other things you can do with sectsty

Here are a few other things you can do using sectsty; the techniques for modifying section heading numbers work with or without sectsty. If you find that you can already get sectsty to do what you want to do, it might be an idea to leave reading this section until later.

If you are going to read on, an apology: the sectsty package was originally meant to be a very simple thing to use. Various requests from people who wanted to do various things mean that it's no longer as simple to use as I'd like. This can't be helped: some people *need* underlining, and that's that. Unfortunately, the documentation has ended up much more complicated that I'd like. This section makes it even worse. I hope that the use

people get from the over long documentation outweighs the extra hassle it causes.

8.1 Suppressing a page number

It's occasionally useful to be able to suppress page numbering on the first page of a chapter. You can do this by saying:

```
\chapterfont{\thispagestyle{empty}}
```

This works because the argument to the `\chapterfont` command is (obviously) executed on the page that the chapter begins on. As luck has it, it's executed after the `\thispagestyle{plain}` command issued by the `\chapter` command, so the page style selected by the `\chapterfont` command is the one that's used.

8.2 Rules around sectional headings

I've included a 'unofficial' command, `\sectionrule`, in sectsty that can put a solid rule across the entire width of the page, above and below a sectional heading. It's probably a good idea to visit the library and study some typography books before using this command. The command must be used as the last command in a `\`⟨*section*⟩`font` command, and has this syntax:

`\sectionrule` `{`⟨*raise top by*⟩`}{`⟨*top rule height*⟩`}`
`{`⟨*raise bottom by*⟩`}{`⟨*bottom rule height*⟩`}`

`{`⟨*raise top by*⟩`}` is the length that the top rule is raised above the baseline of the top line of the heading; `{`⟨*top rule height*⟩`}` is the height (thickness) of the top rule; `{`⟨*raise bottom by*⟩`}` is the length that the bottom rule is raised above the baseline of the bottom line of the heading (a negative length places the rule underneath the last line); and `{`⟨*bottom rule height*⟩`}` is the height (thickness) of the bottom rule.

The `\sectionrule` command works like this:

```
\sectionfont{\sectionrule{3ex}{3pt}{-1ex}{1pt}}
```

What this example does is place two rules across the full width of the text. One is a 3 pt rule 3 ex above the baseline of the top line of the heading, and the other is a 1 pt rule 1 ex below the baseline of the bottom line of the heading.

The vertical space before and after sectional headings can be affected by these rules: the vertical space around a heading is placed before and after *everything* that is printed for the sectional heading, so if a rule is the topmost item in the sectional heading, then the vertical space will be measured to that rule.

Setting the height of either rule to 0 pt will make it invisible, although such a rule can still have an effect on spacing if you raise or lower it.

The `\sectionrule` command has a few problems: in particular, it can't place rules around centred headings, and it doesn't work properly with run in headings such as the standard `\paragraph` and `\subparagraph` headings. You might like to try the titlesec package if sectsty can't help you do what you want.

Because `\sectionrule` must be the last command in a `\`⟨*section*⟩`font` command, it can't be used with underlined section headings: the underlining commands must also be placed last, so you can't have both underlining and rules around a sectional heading.

I wrote the `\sectionrule` command and it will be staying in sectsty. I call it 'unofficial' because sectsty is meant to be very simple, and this command takes things a long way away from my original 'design concept'; this is why it's not mentioned in the bulk of the documentation above.

8.3 A box around sectional headings

You can put a sectional heading inside a frame using the standard LaTeX `\fbox` command like this:

```
\sectionfont{\noindent\fbox}
```

This has two limits: it doesn't work with `\chapter` or `\part` headings, and it only works with single line headings. The reason for this last problem is that the `\fbox` command typesets text in LR mode and will therefore always produce a single line of text.

I've heard that other box making commands can be used to place a frame around a sectional heading in a similar way (`\colorbox` apparently works). If you use any box making command like this, you will need to add a `\noindent` as with `\fbox`. This is because the box making command switches TeX from vertical mode to horizontal mode, which inserts a paragraph indent unless there's a `\noindent` at that point. There is a `\noindent` command buried deep inside the code that produces sectional headings, but too late on to help with this.

Another way of putting a box round a sectional heading gets round the one-line limit of the simple approach above:

```
\makeatletter

\newcommand{\sectbox}[1]{%
  \noindent\protect\fbox{%
    \@tempdima=\hsize
    \advance\@tempdima by-2\fboxsep
    \advance\@tempdima by-2\fboxrule
```

```
    \protect\parbox{\@tempdima}{%
      \smallskip
      % extra commands here
      #1\smallskip
      }}}

\makeatother

\sectionfont{\sectbox}
```

Again, this doesn't work properly with `\chapter` or `\part` headings; another limit is that the `\sectbox` command must be the last command in the \⟨*section*⟩`font` command.

This technique has two significant differences to the first suggestion: firstly, it puts a bit of extra vertical space between the text and the frame, which improves the appearance of the heading; secondly, it places the text of the heading inside a `\parbox` that stretches across the full width of the text. This means that long headings can be broken across lines, and also that the surrounding frame extends across the full width of the text, rather than just enclosing the text of the heading.

While you can use fount changing commands as usual with the `\sectbox` command – `\sectionfont{\sffamily\sectbox}` will work as expected – the use of `\parbox` means that `\raggedright` and similar commands won't work in \⟨*section*⟩`font` commands. There is a way round this problem: put the appropriate command in the definition of `\sectbox` at the place marked `extra commands here`. For example, you could say:

```
\makeatletter

\newcommand{\sectbox}[1]{%
\noindent\protect\fbox{%
\@tempdima=\hsize
\advance\@tempdima by-2\fboxsep
\advance\@tempdima by-2\fboxrule
\protect\parbox{\@tempdima}{%
\smallskip
\raggedright% extra commands here
#1 \smallskip
}}}

\makeatother

\sectionfont{\sectbox}
```

and you'd get raggedright headings within the box. Both `\raggedleft` and `\centering` also work, although you might like to use `\nohang` with

`\centering`.

8.4 Changes to section numbers

The two suggestions below don't require sectsty; I'm including them because if you're using sectsty, you might want to do something like this.

The suggestions modify the `\@seccntformat` command. This is the internal LaTeX command which prints the number part of all sectional headings except chapter and part headings. It's quite in order to change this sort of command: one of the reasons this command exists in the first place is so that people can change it to modify the output they get with LaTeX. You can find out more about it by LaTeXing `classes.dtx`, part of the standard LaTeX distribution.

The best place in general for the modifications I suggest in this section is in a class or package file. If you put the modifications there, you won't need the surrounding `\makeatletter` and `\makeatother` commands.

If you find yourself wanting to do more stuff along these lines, it might be an idea to look at the titlesec package from CTAN.

8.4.1 Printing the section number in the left margin

This is a quick hack I stumbled over by accident one day and I hope someone else finds it useful. I haven't tested it thoroughly, so do be aware that it might produce unwanted effects; I can't see that it will do anything untoward, but I think it's best to be paranoid when dealing with computers.

Putting this code in the preamble of your document:

```
\makeatletter
\def\@seccntformat#1{\protect\makebox[0pt][r]{\csname
                      the#1\endcsname\quad}}
\makeatother
```

will typeset the number of each section in the left margin, with the start of each instance of sectional heading text aligned with the left hand edge of the body text. It doesn't affect chapter or part headings. This code won't affect the normal paragraph and subparagraph headings. If you change the normal form of these headings by telling LaTeX to print section numbers with them, paragraph headings will work as you'd expect, but subparagraph headings will have the section number to the left of the normal position although not in the left margin.

8.4.2 Putting a full stop (period) after the section number

Requests on how to do this appear in `news://comp.text.tex` quite often. One solution is:

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

This will put a full stop after sectional numbers in sectional headers other than part and chapter headings. It doesn't affect cross-references or table of contents entries.

An interesting extension to the simple original command allows you to control the style of numbering you get with each level of sectional heading independently:

```
\makeatletter
\def\@seccntformat#1{\@ifundefined{#1@cntformat}%
{\csname the#1\endcsname\quad}%  default
{\csname #1@cntformat\endcsname}% individual control
}
\def\section@cntformat{\thesection.\quad}
\def\subsection@cntformat{\thesubsection.\quad}
\makeatother
```

this example gives you a full stop after section and subsection headings, and has no effect on other levels.

9 Compatibility with other packages

The sectsty package works by blindly re-defining the various sectioning commands, so any package that requires that LaTeX's sectioning commands are precisely as they are defined in the standard classes will have trouble working with sectsty.

The fncychap package, for example, re-defines the code that produces chapter headings. If you use fncychap with sectsty, you will have to load fncychap after sectsty:

```
\usepackage{sectsty}
\usepackage{fncychap}
```

You'll find that sectsty will have no effect on chapter headings when you use it like this.

Some packages perform interesting tricks to add code to the standard sectioning commands. These packages might well work quite happily with sectsty, but they must be loaded after sectsty: because sectsty re-defines the sectioning commands, any earlier changes will be thrown away.

In particular, the minitoc package modifies the sectioning commands (rather than just re-defining them like sectsty does), but if you say:

```
\usepackage{sectsty,minitoc}
```

to load minitoc after sectsty, everything should work perfectly.