The **savetrees** package*

Scott Pakin
scott+st@pakin.org

July 19, 2004

1   Introduction

The goal of the **savetrees** package is to pack as much text as possible onto each page of a LaTeX document. Admittedly, this makes the document far less attractive. Nevertheless, **savetrees** is a simple way to save paper when printing draft copies of a document. It can also be useful when trying to meet a tight page-length requirement for a conference or journal submission.

Table 1 lists the various ways that **savetrees** compresses documents. Each of these techniques can be selectively disabled through the use of package options. As may be apparent from the table, only a few techniques are beyond the capabilities of a LaTeX novice. **savetrees**'s "value added" is the way that it aggregates a variety of space-saving mechanisms into a single package and makes it easy to manage the tradeoff between lower page count and higher readability.

In addition to providing a LaTeX 2_ε style file, the **savetrees** package also provides a BibTeX style file. `savetrees.bst` exhibits the following salient differences from `plain.bst`:

- Abbreviations are used wherever possible:

chapter	$\implies$	chap.
edition	$\implies$	ed.
editor *or* editors	$\implies$	ed. *or* eds.
January, February, …	$\implies$	Jan., Feb., …
page *or* pages	$\implies$	p. *or* pp.
Technical Report	$\implies$	Tech. Rep.

- At most two authors are listed. The remainder are replaced by "et al."

- Authors' names are abbreviated to their initials plus surname (e.g., "S. D. Pakin").

---

*This document corresponds to **savetrees** v1.1, dated 2004/07/17.

Space-saving technique	Implementation
Typeset section titles smaller and with less surrounding whitespace	Use the titlesec package
Reduce page margins	Use the geometry package
Reduce indentation and remove inter-item spacing from the various list environments	Redefine list environments
Relax float placement (more floats per page, increased ability to share pages with text, etc.)	Reassign $\text{\LaTeX}\,2_\varepsilon$ variables
Decrease paragraph indentation	Reassign \parindent
Typeset document title with smaller fonts and with less surrounding whitespace	Redefine \maketitle
Reduced interline spacing	Reassign \baselinestretch
Typeset bibliographies smaller and with no inter-item spacing	Redefine thebibliography

Table 1: savetrees's space-saving techniques

In addition, `savetrees.bst` does not normally typeset `NOTE` fields, although it can be instructed to via a **savetrees** package option.

Finally, the **savetrees** package includes a Perl script called "`makethin`", which automatically generates narrower versions of TeX fonts plus configuration files for Dvips and pdfLaTeX.

2   `savetrees.sty`—reduce document whitespace

To use `savetrees.sty`, merely load it into a document by putting "\usepackage{savetrees}" in the document's preamble. By default, all of **savetrees**'s space-saving techniques are enabled. However, package options can disable any features that violate given formatting requirements, that conflict with other LaTeX 2_ε packages, or that you simply consider excessively ugly. **savetrees** supports the following options:

normalsections Don't modify section headers.

normalmargins Don't modify page margins.

normallists Don't modify the `itemize`, `enumerate`, or `description` environments.

normalfloats Don't modify LaTeX 2_ε's float parameters.

normalindent Don't modify paragraph indentation.

normaltitle Don't modify the formatting of the document title.

normalleading Don't modify interline spacing.

normalbib Don't modify bibliography formatting.

normalbibnotes Don't omit `NOTE` fields from the bibliography (only meaningful with the **savetrees** bibliography style).

For example, to keep page margins and interline spacing as they are but save space everyplace else, you should put the following \usepackage line in your document's preamble:

```
\usepackage[normalmargins,normalleading]{savetrees}
```

3   `savetrees.bst`—abbreviate bibliographic information

One of the advantages of a tool like BibTeX is that the bibliographic database can—and should—contain complete bibliographic information for each reference, while style files determine the subset of that information that is actually typeset.

`savetrees.bst` saves space by truncating `AUTHOR` fields to two authors plus "et al.", listing authors by initials and surname only, abbreviating the `MONTH` field to three letters, and (by default) omitting the `NOTE` field altogether.

To use the `savetrees.bst`, simply add "\bibliographystyle{savetrees}" to your document (or replace an existing \bibliographystyle). Then, to give `savetrees.bst`—or *any* BiBTeX style file—maximum flexibility, you should obey the following rules when writing your `.bib` file:

1. Use the three-letter month macros defined by virtually all BiBTeX style files instead of spelling out month names explicitly:

Good:	`MONTH = sep,`
	Can be typeset as "September", "Sept.", "SEP", "Septiembre", etc.
Bad:	`MONTH = {September},`
	Can be typeset only as "September".

2. Include authors' full names (or as much of each name as is available); let BiBTeX abbreviate as necessary:

Better:	`AUTHOR = {Rufus Xavier Sarsaparilla},`
	Can be either typeset in full or abbreviated to "Rufus X. Sarsaparilla", "R. X. Sarsaparilla", etc.
Worse:	`AUTHOR = {R. X. Sarsaparilla},`
	Can be typeset as "R. X. Sarsaparilla"—or even a more abbreviated form—but can't be expanded to the full name.

3. Include the names of *all* authors; let BiBTeX decide where to truncate the list:

Good:	`AUTHOR = {Rufus Xavier Sarsaparilla and Rafaella Gabriela Sarsaparilla and Albert Andreas Armadillo},`
	All authors can be named, or the list can be truncated at any point with "et al.", "and others", or whatever.
Bad:	`AUTHOR = {Rufus Xavier Sarsaparilla and others},`
	At most one author can be named, but "and others" can still be replaced by "et al." or a different phrase, the font can be varied, and the author's name can be abbreviated, as in the previous rule.
Worse:	`AUTHOR = {{Rufus Xavier Sarsaparilla, et al.}},`
	Can be typeset only precisely as "Rufus Xavier Sarsaparilla, et al."

The savetrees BiBTeX style utilizes the same fields as the standard BiBTeX styles (`plain`, `alpha`, `abbrv`, `unsrt`, etc.), with the exception that the `NOTE` field is normally suppressed. To include `NOTE` fields in your Bibliography/References sections, pass the normalbibnotes option to the savetrees LaTeX style file.

`savetrees.bst` can't normally be used independently of `savetrees.sty`, because the former typesets `NOTE` fields within a \savetreesbibnote{...} call, which is defined by the latter. If you desperately want to use `savetrees.bst` without having to load `savetrees.sty`, then you should examine the definition of the \savetreesbibnote macro on page 20 of this document (code lines 147–151).

4 makethin—make thinner versions of TeX fonts

Question: What's the narrowest font?

Answer: It depends upon how you measure. Table 2 shows, for various fonts, the width in points of 1000 lowercase letters with relative frequencies chosen to match "typical" English text. There are 130 e's, 93 t's, 78 n's, 77 r's, and so forth down the frequency distribution. According to the table, Times Roman is statistically likely to be the best typeface for maximizing the amount of text on the page. However, Times Roman may not be the narrowest for *your* document; you'll have to experiment and see.

Typeface	Package	Avg. width (pt.)
Times Roman	times	4.26901
Computer Modern	(default)	4.62675
Charter	charter	4.67613
Helvetica	helvetic	4.70108
Palatino	palatino	4.79744
Utopia	utopia	4.92876
New Century Schoolbook	newcent	4.98047
Avant Garde	avantgar	5.22113
Bookman	bookman	5.23056
Courier	courier	6

Table 2: Common fonts sorted by increasing width

We can do better than merely *selecting* a narrow font, though. We can *generate* a narrower version of an existing font. The idea is to present TeX with font metrics that indicate thinner characters and to tell Dvips to use the PostScript "ExtendFont" operator to compress all of the glyphs. The savetrees package comes with a Perl script, makethin, which automates the task of generating narrow font variants. From a typography viewpoint, the results are deplorable, as the glyphs are simply squeezed rather than optically scaled to a new width. But from the perspective of trying to pack more text onto a page with minimal effort, makethin can be quite efficacious.

Warning

> To avoid confusion with the original font-metric files, do not distribute the `.tfm` or `.vf` files generated by makethin unless you rename them first.

In addition to requiring Perl, makethin depends upon two helper programs: kpsewhich and dvitype. kpsewhich searches the TeX installation directory for a given file or file type. While kpsewhich is a standard part of Kpathsea-based TeX distributions, such as teTeX, it does not normally function on non-Kpathsea

based distributions, such as MiKTEX.[1] `dvitype` outputs a DVI file in a textual format that's easy for `makethin` to process. `makethin` extracts font names from lines like the following:

```
Font 44: cmtt10---loaded at size 655360 DVI units
Font 43: cmtt12 scaled 1200---loaded at size 943718 DVI units
Font 33: cmss10---loaded at size 655360 DVI units
```

The `makethin` script contains documentation in POD (Plain Old Documentation) format. This can be extracted using `makethin`'s `--man` or `--man-ps` options, as described below, or any of Perl's "`pod2`⟨*something*⟩" converters, such as `pod2text` or `pod2html`. Beginning on the following page is the `makethin` documentation, as extracted by `pod2latex` (with some minor formatting modifications).

---

[1]I have no current plans to port `makethin` to non-Kpathsea based TEX distributions, but volunteers to do the work are certainly welcome.

NAME

makethin—make thinner versions of PostScript fonts for TEX

SYNOPSIS

makethin [**--verbose**] [**--xscale**=*factor*] [**--cleanfirst**] [**--clean**] [**--config**=*extension*] [**--pdftex**] [**--extramaps**=*file*[,*file*]...] [**--skip-fonts**=*font*[,*font*]...] [**--basename**=*string*] [**--extrafonts**=*font*[,*font*]...] *.dvi file*

makethin [**--verbose**] **--help**

makethin **--man**=*man page* | **--man-ps**=*PostScript file*

DESCRIPTION

makethin produces thinner versions of PostScript fonts for use with TEX/LATEX and Dvips. More precisely, it finds all of the *.tfm* and *.vf* fonts referred to by a *.dvi* file, scales the character-width metrics by a given amount, and writes new *.tfm* and *.vf* files to the current directory. **makethin** then generates a customized *.map* and *config* file for Dvips and, optionally, a customized *pdftex.cfg* file for pdfLATEX.

The general procedure for using **makethin** is as follows:

1. Run **latex** on your *.tex* source file to produce a *.dvi* file.

2. Run **makethin** on the *.dvi* file to create new *.tfm*, *.vf*, *.map*, and *config* files.

3. Re-run **latex** on your *.tex* source file to typeset it with the new fonts.

4. Run **dvips** on the *.dvi* file, specifying the newly generated *.map* and *config* files, to produce a *.ps* file.

OPTIONS

The following are the command-line options that **makethin** accepts:

-v, --verbose
> Increase the verbosity of the status output. **--verbose** can be specified multiple times on the same command line, with each **--verbose** futher increasing the verbosity. (Currently, two **--verbose**s have maximal impact.)

-h, --help

  Display basic usage information. When combined with **--verbose**, additionally describes each of the command-line options. When combined with a second **--verbose**, **--help** outputs the complete **makethin** manual page.

-x *factor*, **--xscale=***factor*

  Scale fonts horizontally by a factor of *factor*. The default, `0.5`, produces nearly illegible fonts but is useful for verifying that **makethin** actually worked. Factors of `0.90`–`0.99` are more reasonable.

-C, --cleanfirst

  Delete all files generated by a previous run of **makethin** before generating new ones.

-c, --clean

  Delete all files generated by a previous run of **makethin** and then exit.

-P *extension*, **--config=***extension*

  Process all of the *.map* files named in *config.extension* (found in Dvips's configuration directory). **--config** can be specified multiple times on the same command line. *config.ps* is processed implicitly. The same **-P** arguments that you would normally pass to **dvips** to utilize entirely PostScript fonts (e.g., **-Pcmz** and **-Pamz**) should also be passed to **makethin**.

-p, --pdftex

  In addition to producing the files needed by Dvips, also produce a *pdftex.cfg* file that is usable by pdfLATEX.

-m *file*[,*file*]..., **--extramaps=***file*[,*file*]...

  Additionally process the named Dvips *.map* files even if they're not referred to by any of the *config.extension* files specified with **--config**. **--extramaps** can be specified multiple times on the same command line.

-k *font*[,*font*]..., **--skipfonts=***font*[,*font*]...

  Don't make thin versions of the named fonts, even if they're listed in the *.dvi* file. Fonts are named using the Berry scheme (i.e., "`pcrr8a`" as opposed to "`COURB`" or "`Courier-Bold`"). **--skipfonts** can be specified multiple times on the same command line.

-b *string*, **--basename=***string*

  Tell **makethin** to use *string* as the base name for the Dvips configuration files it generates. The default is "`thin`", so **makethin** normally produces files named *config.thin* and *thin.map*, but **--basename** enables alternate filenames to be used.

-e *font*[,*font*]..., **--extrafonts=***font*[,*font*]...

  Make thin versions of the named fonts, even if they're not listed in the *.dvi* file. Fonts are named using the Berry scheme (i.e., "`pcrr8a`" as opposed to "`COURB`" or "`Courier-Bold`"). **--extrafonts** can be specified multiple times on the same command line.

--man=*man page*
> Create a Unix manual page for **makethin** in the standard, ∗roff format.
> Typical usage is:
>
>   makethin --man=/usr/man/man1/makethin.1

--man-ps=*PostScript file*
> Create a Unix manual page for **makethin** in PostScript format instead of
> ∗roff format.

In addition to the options listed above, **makethin** has a required argument, which
is the name of a *.dvi* file from which to read font information.

EXAMPLES

The following are some examples of how to use **makethin**.

A typical case

First, we need to produce *myfile.dvi*, because that contains the font information
that **makethin** will read:

    latex myfile.tex

Next, we invoke **makethin**, telling it to process *config.cmz* (which, in turn, causes
psfonts.cmz to be processed), This tells **makethin** to use PostScript versions of
the Computer Modern fonts instead of bitmapped versions. (**makethin** can scale
only PostScript fonts.) We also specify maximal verbosity:

    makethin -Pcmz --verbose --verbose myfile.dvi

The preceding line reads *config.ps*, *config.cmz*, various map files, such as *psfonts.map* and *psfonts.cmz*, and all of the *.tfm* and *.vf* files mentioned in *myfile.dvi*. It then writes *config.thin*, *thin.map*, and modified versions of all of the
.tfm and *.vf* files to the current directory.

We now need to re-run **latex**, so it can produce a new *myfile.dvi* using the thinner
metrics listed in the current directory's *.tfm* and *.vf* files:

    latex myfile.tex

Finally, we produce a PostScript file using the newly generated using *config.thin* and *thin.map* files:

```
dvips -Pthin myfile.dvi -o myfile.ps
```

If all worked according to plan, *myfile.ps* should be typeset using extremely thin (half-width) versions of its original fonts.

Producing thin fonts for use in pdfLaTeX

Because **makethin** can read only *.dvi* files, not *.pdf* files, we first need to produce a *.dvi* file:

```
latex too-long.tex
```

too-long.dvi is typeset entirely using the Times family of fonts. Therefore, we don't need to specify **-Pcmz**. However, **pdflatex** normally embeds Times, thereby precluding **makethin**'s ability to scale it. (**makethin** requires a *.pfb* font file in order to scale the corresponding font.) Fortunately, */usr/share/texmf/dvips/config/ar-std-urw-kb.map* already contains the proper mapping of TeX names to *.pfb* files for Times, Courier, and Helvetica. We can tell **makethin** to use that file:

```
makethin --cleanfirst -v -v too-long.dvi --pdftex --xscale=0.9
  --extramaps=/usr/share/texmf/dvips/config/ar-std-urw-kb.map
```

In the preceding line, we changed the scaling factor from the default of 0.5 to a more reasonable 0.9. Because we had some 0.5-scaled *.tfm* and *.vf* files left over from the previous example, we specified **--cleanfirst** to delete those old font files. We specified **--pdftex** to make **makethin** produce a local *pdftex.cfg* file. And we told **makethin** where to find the extra map file needed to force the usage of *.pfb* files.

All that's left is to run **pdflatex** to produce a *.pdf* file:

```
pdflatex too-long.tex
```

pdflatex will read the font metric files (*.tfm* and *.vf*) and *pdftex.cfg* from the current directory. This will tell it to load *thin.map*, which specifies the scaling factor. The result should be a document with each character squeezed to 90% of its original width.

FILES

perl
> interpreter/compiler needed to run the **makethin** script

kpsewhich
> finds files within the T_EX directory tree

dvitype
> outputs the typesetting commands contained within a *.dvi* file

∗.tfm **and** *∗.vf*
> T_EX font metrics and virtual fonts–metrics specifying the width of each character in a font

config.∗
> Dvips configuration files, each containing (among other information) a list of font-map files

psfonts.∗, *∗.map*
> Dvips font-map files, which map T_EX font names to PostScript font names and *.pfb* files

pdftex.cfg
> pdfT_EX and pdfLaT_EX configuration files, each containing (among other information) a list of font-map files

groff
> used by **--man-ps** to produce a PostScript version of the Unix manual page for **makethin**

RESTRICTIONS

The most serious restriction is that **makethin** doesn't work on Computer Modern Roman 10pt. (*cmr10*)—the default T_EX/LaT_EX font. The reason, I believe, is that T_EX and LaT_EX preload that font's metrics (*cmr10.tfm*) and therefore ignore the scaled *cmr10.tfm* on disk. It may be possible to work around this limitation by copying *cmr10.tfm* to a new name and convincing LaT_EX to use that name where it would otherwise have used *cmr10.tfm*. However, it's much easier merely to use a different font family (e.g., with "`\usepackage{times}`") for typesetting your document when you know you want to run **makethin**.

SEE ALSO

dvips(1), *latex*(1), *pdflatex*(1), the LaT_EX `savetrees` package

AUTHOR

Scott Pakin, *scott+st@pakin.org*

5 Implementation of `savetrees.sty`

This section lists the complete, commented source code for the savetrees package. Although reading this section—and the subsequent implementation sections—is not necessary for understanding how to use savetrees, it may be a useful teaching tool for LaTeX newcomers. For instance, if you want narrow page margins, but find savetrees's defaults to be *too* narrow, then seeing how savetrees alters margins may be instructive for learning how to select your own page margins.

We start by telling DocStrip that this is where `savetrees.sty` begins:

1 ⟨∗package⟩

5.1 Option processing

By default, savetrees tries to make documents extremely dense. However, this also makes them rather ugly. The package options defined in this section let the author specify which space-saving routines are unacceptably grotesque and should not be utilized.

We now define one new `\if` statement for each package option and define each package option to set the corresponding "`\@st@normal@...`" flag to *true*.

`\if@st@normal@sections`
`\@st@normal@sectionstrue`
`\@st@normal@sectionsfalse`

These are used to conditionally disable savetrees's modifications to section titles.

```
2 \newif\if@st@normal@sections
3 \DeclareOption{normalsections}{\@st@normal@sectionstrue}
```

`\if@st@normal@margins`
`\@st@normal@marginstrue`
`\@st@normal@marginsfalse`

These are used to conditionally disable savetrees's modifications to page margins.

```
4 \newif\if@st@normal@margins
5 \DeclareOption{normalmargins}{\@st@normal@marginstrue}
```

`\if@st@normal@lists`
`\@st@normal@liststrue`
`\@st@normal@listsfalse`

These are used to conditionally disable savetrees's modifications to the various list environments.

```
6 \newif\if@st@normal@lists
7 \DeclareOption{normallists}{\@st@normal@liststrue}
```

`\if@st@normal@floats`
`\@st@normal@floatstrue`
`\@st@normal@floatsfalse`

These are used to conditionally disable savetrees's modifications to LaTeX's float-placement parameters.

```
8 \newif\if@st@normal@floats
9 \DeclareOption{normalfloats}{\@st@normal@floatstrue}
```

`\if@st@normal@indent`
`\@st@normal@indenttrue`
`\@st@normal@indentfalse`

These are used to conditionally disable **savetrees**'s modifications to paragraph indentation.

```
10 \newif\if@st@normal@indent
11 \DeclareOption{normalindent}{\@st@normal@indenttrue}
```

`\if@st@normal@title`
`\@st@normal@titletrue`
`\@st@normal@titlefalse`

These are used to conditionally disable **savetrees**'s modifications to title formatting.

```
12 \newif\if@st@normal@title
13 \DeclareOption{normaltitle}{\@st@normal@titletrue}
```

`\if@st@normal@leading`
`\@st@normal@leadingtrue`
`\@st@normal@leadingfalse`

These are used to conditionally disable **savetrees**'s modifications to interline spacing. This spacing is known as "leading" because of the additional strips of lead placed between lines in the days of metal type.

```
14 \newif\if@st@normal@leading
15 \DeclareOption{normalleading}{\@st@normal@leadingtrue}
```

`\if@st@normal@bib`
`\@st@normal@bibtrue`
`\@st@normal@bibfalse`

These are used to conditionally disable **savetrees**'s modifications to bibliography formatting.

```
16 \newif\if@st@normal@bib
17 \DeclareOption{normalbib}{\@st@normal@bibtrue}
```

`\if@st@normal@bibnotes`
`\@st@normal@bibnotestrue`
`\@st@normal@bibnotesfalse`

These are used to conditionally include `NOTE` fields when using `savetrees.bst`.

```
18 \newif\if@st@normal@bibnotes
19 \DeclareOption{normalbibnotes}{\@st@normal@bibnotestrue}
```

We need to process our options immediately, because they'll be needed in the following code to determine which packages to load and what macros to define.

```
20 \ProcessOptions\relax
```

5.2   Section titles

The LaTeX default is to typeset section titles in a large font and with significant surrounding whitespace. We use the **titlesec** package to typeset section titles in the same font size as the body text and to leave only a single blank line above and below them.

```
21 \if@st@normal@sections
22 \else
23   \RequirePackage[tiny,compact]{titlesec}
24 \fi
```

5.3 Page margins

The typesetting wisdom of the ages says that the human eye is most comfortable reading approximately 60 characters per line of text, and this is what LaTeX's default margins aim to achieve. Of course, narrower margins mean fewer pages, and that's what savetrees is striving for.

\@st@marginsize 1.5 cm might not be the best margin size, so this value is stored in the \@st@marginsize register, where it can easily be changed.

```
25 \newlength{\@st@marginsize}
26 \setlength{\@st@marginsize}{1.5cm}
```

Use the geometry package to narrow our page margins, unless the author wants to keep LaTeX's original ones. Note that we accept geometry's default of zero space allocated to marginal notes.

```
27 \if@st@normal@margins
28 \else
29   \RequirePackage[lmargin=\@st@marginsize,
30                   rmargin=\@st@marginsize,
31                   tmargin=0pt,
32                   bmargin=\@st@marginsize]{geometry}
33 \fi
```

5.4 List spacing

We try to save space in itemized lists, enumerated lists, and description lists by reducing indentation slightly and by eliminating inter-item spacing altogether.

We make no modifications if the author prohibits us from doing so.

```
34 \if@st@normal@lists
35 \else
```

The calc package helps simplify our list redefinitions.

```
36   \RequirePackage{calc}
```

itemize Except where indicated, the following code was taken directly from LaTeX 2ε's definition of the itemize environment, in ltlists.dtx:

```
37   \def\itemize{%
38     \ifnum \@itemdepth >\thr@@\@toodeep\else
39       \advance\@itemdepth\@ne
40       \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
41       \expandafter
42       \list
```

15

```
43          \csname\@itemitem\endcsname
44          {\def\makelabel##1{\hss\llap{##1}}%
```

The following lines have been modified from the original.

```
45          \settowidth{\leftmargin}{\csname\@itemitem\endcsname}%
46          \addtolength{\leftmargin}{\labelsep * \@itemdepth}%
47          \setlength{\topsep}{4pt plus 1pt minus 2pt}%
48          \setlength{\itemsep}{0pt}%
49          \setlength{\parsep}{0pt}%
```

That's it for the modifications. We can now finish up the redefinition of `itemize`.

```
50          }%
51      \fi}
```

enumerate   Except where indicated, the following code was taken directly from LaTeX 2ε's definition of the `enumerate` environment, in `ltlists.dtx`:

```
52  \def\enumerate{%
53      \ifnum \@enumdepth >\thr@@\@toodeep\else
54        \advance\@enumdepth\@ne
55        \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
56          \expandafter
57          \list
58            \csname label\@enumctr\endcsname
59            {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
```

The following lines have been modified from the original.

```
60              \settowidth{\leftmargin}{\csname label\@enumctr\endcsname}%
61              \addtolength{\leftmargin}{\labelsep * \@enumdepth}%
62              \setlength{\topsep}{4pt plus 1pt minus 2pt}%
63              \setlength{\itemsep}{0pt}%
64              \setlength{\parsep}{0pt}%
```

That's it for the modifications. We can now finish up the redefinition of `enumerate`.

```
65              }
66      \fi}
```

description The `description` environment is a bit simpler than the `itemize` and `enumerate` environments; it's a direct application of `list`. All we need to do is reduce the left margin from the `list` default of 2 em to a slightly denser 1 em.

```
67  \renewenvironment{description}{%
68      \begin{list}{}{\setlength{\leftmargin}{1em}%
69                    \labelwidth\z@ \itemindent-\leftmargin
70                    \let\makelabel\descriptionlabel}%
71  }{%
```

16

```
72     \end{list}
73   }
74 \fi
```

5.5 Float placement

\topfraction
\bottomfraction
\textfraction
\floatpagefraction
\dbltopfraction
\dblfloatpagefraction

LaTeX normally doesn't try very hard to pack floats onto a page. The following parameter changes attempt to reduce the number of float pages (and hence, total pages). As always, the author can require that the original float-placement parameters be used instead.

```
75 \if@st@normal@floats
76 \else
77   \renewcommand{\topfraction}{0.85}
78   \renewcommand{\bottomfraction}{0.85}
79   \renewcommand{\textfraction}{0.1}
80   \renewcommand{\floatpagefraction}{0.85}
81   \renewcommand{\dbltopfraction}{0.85}
82   \renewcommand{\dblfloatpagefraction}{.85}
83   \setcounter{topnumber}{25}
84   \setcounter{bottomnumber}{25}
85   \setcounter{totalnumber}{25}
86   \setcounter{dbltopnumber}{25}
87 \fi
```

5.6 Paragraph indentation

LaTeX normally provides 2 em of indentation at the start of each paragraph. We can save a little space by reducing that to 1 em.

```
88 \if@st@normal@indent
89 \else
90   \setlength{\parindent}{1em}
91 \fi
```

5.7 Document title formatting

By default, the document title is typeset in the `\LARGE` font size, and the author list and date are typeset `\large`. We redefine `\@maketitle` to typeset the title `\large` and everything else in the body font. In addition, we remove the extra whitespace above the title and lessen the whitespace below the title.

We proceed only with the author's permission.

```
92 \if@st@normal@title
93 \else
```

\@maketitle The following was taken largely from `classes.dtx`, but modified as specified above.

```
 94   \def\@maketitle{%
 95     \newpage
 96     \null
 97     \begin{center}%
 98       \let \footnote \thanks
 99       {\large \textbf{\@title}\par}
100       \vskip 0.5\baselineskip
101       \begin{tabular}[t]{c}%
102         \@author
103       \end{tabular}\par
104       \vskip 0.5\baselineskip
105       \@date
106     \end{center}%
107     \par
108     \vskip \baselineskip
109   }
110 \fi
```

5.8 Interline spacing

A document's page count can be reduced quite significantly by reducing the amount of whitespace between successive lines of text, so that's exactly what we do below.

\baselinestretch 90% of normal leading gives very good compression but still prevents descenders from running into successive ascenders. 95% would look more subtle but, of course, would fit less text per page.

```
111 \if@st@normal@leading
112 \else
113   \renewcommand{\baselinestretch}{0.9}
114 \fi
```

5.9 Bibliography formatting

There are two ways we save space when typesetting bibliographies. First, we omit blank linkes between entries. And second, we typeset the entire bibliography— excluding the section title—with `\small`.

We start, as always, by giving the author a chance to override our changes.

```
115 \if@st@normal@bib
116 \else
```

thebibliography   The following was taken largely from `classes.dtx`; see that file for additional documentation. savetrees's modifications are indicated below.

```
117    \renewenvironment{thebibliography}[1]{%
```

In the `article` document class, a bibliography is a *section* called "`\refname`". In the `report` and `book` document classes, a bibliography is a *chapter* called "`\bibname`". In `classes.dtx`, the correct code is extracted by DocStrip. Here, we have to use an `\ifx` primitive to select the appropriate title and formatting.

```
118      \ifx\chapter\@undefined
119        \section*{\refname
120          \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}}%
121      \else
122        \chapter*{\bibname
123          \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}}%
124      \fi
```

Back to the original code. . .

```
125      \list{\@biblabel{\@arabic\c@enumiv}}%
126           {\settowidth\labelwidth{\@biblabel{#1}}%
127            \leftmargin\labelwidth
128            \advance\leftmargin\labelsep
```

We eliminate the space between paragraphs, and we set the space between items to only 1 pt. We could have set this to 0 pt., but the extra space helps keep the citation numbers' brackets from getting too close to each other, vertically.

```
129            \setlength{\parsep}{0pt}%
130            \setlength{\itemsep}{1pt}%
```

Back to the original code. . .

```
131            \@openbib@code
132            \usecounter{enumiv}%
133            \let\p@enumiv\@empty
134            \renewcommand\theenumiv{\@arabic\c@enumiv}}%
```

Although we kept the section title in its original size, we typeset the rest of the bibliography a little bit smaller.

```
135        \small
```

We finish up using the original code. . .

```
136        \sloppy
137        \clubpenalty4000
138        \@clubpenalty \clubpenalty
139        \widowpenalty4000%
140        \sfcode`\.\@m
```

```
141  }{%
142    \def\@noitemerr
143        {\@latex@warning{Empty 'thebibliography' environment}}%
144    \endlist
145  }
146 \fi
```

5.10   Bibliographic notes

\savetreesbibnote   The bibliographies output by the `savetrees.bst` BIBTEX style (to be presented
in Section 6) surround all `NOTE` fields with a call to the `\savetreesbibnote`
macro. As a result, this must be defined for `savetrees.bst` to work. By default,
`\savetreesbibnote` is defined to do nothing. However, the author can specify
that `\savetreesbibnote` should instead output its argument with no additional
processing.

```
147 \if@st@normal@bibnotes
148   \newcommand{\savetreesbibnote}[1]{#1}
149 \else
```

To supppress a `NOTE` field, we also have to gobble the period following the note.
Otherwise, the bibliography will show two periods surrounding an empty note.

```
150   \newcommand{\savetreesbibnote}[1]{\@gobble}
151 \fi
```

This is the conclusion of `savetrees.sty`.

```
152 ⟨/package⟩
```

6   Implementation of `savetrees.bst`

In addition to a LATEX 2ε style, the savetrees package also includes a BIBTEX style,
`savetrees.bst`. `savetrees.bst` was generated with the help of Patrick W. Daly's
custom-bib package. The following options were provided to `merlin.mbs`:

        lang, nm-init, ed-au, nmdash, nmlm, x2, m2, isbn,
        issn, pp, ed, abr, ednx, xedn, jabr, nfss

Because `savetrees.bst` is a generated file—and can be regenerated using the
options listed above—it is unnecessary to list the complete source code in this
document. Rather, only the hand-modified parts are presented below.

```
153 ⟨*bibstyle⟩
```

$$\vdots$$

(460 lines of code omitted)

$$\vdots$$

The following function is the only one modified by Scott Pakin. The modification involves placing the entire note field within \savetreesbibnote{...}. This enables savetrees.sty to selectively define \savetreesbibnote to either output its argument as is or discard it (and the subsequent period). See Section 5.10 for savetrees.sty's definition of \savetreesbibnote.

```
154 FUNCTION {format.note}
155 {
156  note empty$
157     { "" }
158     { "\savetreesbibnote{"
159       note #1 #1 substring$
160       duplicate$ "{" =
161         'skip$
162         { output.state mid.sentence =
163           { "l" }
164           { "u" }
165         if$
166         change.case$
167         }
168       if$
169       note #2 global.max$ substring$ *
170       "}" * *
171     }
172   if$
173 }
```

$$\vdots$$

(941 lines of code omitted)

$$\vdots$$

174 ⟨/bibstyle⟩

7   Implementation of `makethin`

`makethin` is a Perl script that produces narrow variants of all of the fonts used in a document. The script is about five pages long and is not listed here. However, the following is an outline of `makethin`'s behavior:

175 ⟨∗makethin⟩

1. Read all of the TFM and VF fonts named in the given DVI file.

2. For each font, perform the following operations:

   (a) If the font is a virtual font (`.vf`), then add all of the VF and TFM fonts it names to the end of the font "to-do" list.

   (b) If the font contains TeX font metrics (`.tfm`), then convert the font to an ASCII property list (`.pl`), replace each "(CHARWD R $\langle width \rangle$)" expression with "(CHARWD R $\langle width \rangle \times \langle x\text{-}scale \rangle$)", and convert the result back to a `.tfm` file in the current directory.

3. Acquire a list of font-map files from `config.ps` and any other Dvips configuration specified on the command line.

4. Create a new font-map file, containing one line for each font named in the given DVI file, but modified to specify ""$\langle x\text{-}scale \rangle$ `ExtendFont`"" on each line.

5. Create a new Dvips configuration file that points to the new font-map file.

176 ⟨/makethin⟩

Change History

v1.0

    General: Initial version 1

v1.1

    `thebibliography`: Modified to test for `\chapter`, not `\bibname`. . 19

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.

24