Electronic Theses and Dissertations at Pitt (a LaTeX 2ε class)

Federico Garcia *

2004/08/17

Contents

*University of Pittsburgh, Electronic Theses and Dissertations (ETD) Working Group.

Pitt ETD Webpage: `http://www.pitt.edu/~graduate/etd/`

pittetd's Webpage:
`http://www.pitt.edu/~graduate/etd/latextemplate.html/`

TEX Users Group: `http://www.tug.org/` (with a link to CTAN)

1 Introduction

This is the guide to the pittetd LaTeX 2_ε document class, designed for the preparation of electronic theses and dissertations (ETD) at the University of Pittsburgh. It is recommended that users read this entire documentation before starting using pittetd, so that they will have an idea of the different possibilities and options, some of which are particular to pittetd and therefore not usual in standard LaTeX classes.

Users will find below a description of pittetd usage, extended with an introduction to some of the most relevant features of the `hyperref` package. In addition, when this document has been produced by running LaTeX on the file `pittetd.dtx`, it also contains a commented transcript of the code, so that users can modify things if they need to (and know what they are doing). In the version downloadable from the Pitt ETD webpage, this latter part is omitted. A separate document, *Comments on using LaTeX for theses*, also prepared for Pitt ETD authors, describes some standard tools of LaTeX that may or may not be known to the reader but can certainly prove useful when writing the thesis or dissertation. Touched upon are topics such as inclusion of graphics and the handling of large, book-length documents [1].

Throughout this text reference is made to the *Format Guidelines for Electronic Thesis and Dissertation Preparation at the University of Pittsburgh*, downloadable from the Pitt ETD webpage. The abbreviation FG is used to refer to it; page numbers are indicated in parenthesis.

Section 2 describes the creation of interactive PDF files through LaTeX, introducing the two main tools for that effect, the programs PDFLaTeX and dvipdfm. Section 3 explains how to install pittetd and the main LaTeX packages needed for its proper working. Also, cursory information for the installation of PDFLaTeX and dvipdfm is given.

In section 4 some general considerations are given about the best ways to use pittetd (and to cope with its restrictions). Use of packages and the issues it might bring about (notably incompatibility)is treated in a special subsection.

Detailed information about the options, available commands, and use of pittetd can be found in section 5. Typesetting of the preliminary pages is described in subsection 5.4.

Section 6 describes in an introductory way the basic features of `hyperref`, the package that implements interactivity into LaTeX documents.

Finally, section 7 gives some suggestions for a final format review before submitting an ETD written with pittetd. It warns about those problems that are most likely to occur because they lie beyond pittetd's control.

2 PDF creation through LaTeX

In principle, pittetd is equipped to fulfill the basic interactivity requirements of the FG, namely the creation of bookmarks from the entries in the Table of Contents, the List of Figures, and the List of Tables, and the implementation of these entries themselves as links to the corresponding page. This is done by means of invoking the formidable `hyperref` package,[1] which offers the basic functions for interactive handling (section 6 below offers an introductory guide to other features from this package that users can take advantage of). Thus, `hyperref` has to be—and it usually is in standard distributions of LaTeX—installed in the system for pittetd to be able to fulfill these tasks (section 3 offers immediate help on the installation of `hyperref` and other tools, including pittetd itself).

pittetd has been written under the assumption that the user will create the final `.pdf` file through one of two tools, namely PDFLaTeX, or the program dvipdfm. The user indicates which of the two ways is to be used as an option to the pittetd class, namely `pdftex` or `dvipdfm` (on the way to load the class and specify options, see section 5.1). The following sections explain the particularities of each way. A third related option, `nohyperref` option, will be discussed in section 2.4. Note that it is possible to switch back and forth between the three ways just by modifying the relevant option; as far as pittetd is concerned, nothing else is necessary to effect the change.[2]

2.1 PDFLaTeX

The most direct way to obtain a `.pdf` output file is running PDFLaTeX[3] instead of LaTeX. Naturally, PDFLaTeX has to be installed in the system (again, it is usually included in standard distributions of LaTeX; see section 3). The user has to indicate `pdftex` as an option to pittetd, and

[1] Written by Sebastian Rahtz.

[2] However, when going to/from `nohyperref`, it is always good to delete any auxiliary files before running. Also, some of the `hyperref` package's commands discussed in section 6 are of course disabled when `nohyperref` is used.

[3] Created by Hàn Thé Thàn.

this latter will pass that option to other packages that need it, including `hyperref` but also `graphicx` and `color` (this latter used by `hyperref`).

Note that if this is the chosen method, a regular LaTeX (i.e., not PDFLaTeX) run will result in an error message ('`Why not use pdf(e)TeX binaries?`'). This could affect user's habits, batch files, etc.

There is one more significant drawback to the use of PDFLaTeX: the running time is sometimes clearly longer than regular LaTeX. This depends, to be sure, on one of the configuration options of PDFTeX, namely `\pdfcompresslevel`, and it could be modified. But in that case, the resulting file is incomparably larger.[4]

2.2 dvipdfm

As the name indicates, dvipdfm[5] is one of the programs available to convert `.dvi` files into `.pdf`. The procedure then consists in running LaTeX as usual while the document is in preparation, having loaded pittetd with `dvipdfm` option, thus obtaining (more quickly) the usual (and smaller) `.dvi` output. Only optionally, at strategic points in the development of the document (notably at the end), has the user to worry about PDF, and apply dvipdfm to the `.dvi` file. This is usually as simple as typing

```
dvipdfm doc.dvi
```

in the command line. The file `dvipdfm.pdf` is the user's manual for the program and explains the switches that can be used in the command line.

In some `.dvi` viewers the bookmarks (and even the links) are lost. But the relevant information is recovered by dvipdfm when creating the corresponding `.pdf` file.

Another significative advantage of dvipdfm is that it tries to solve inclusion of PostScript graphic files, so it is not always necessary to convert them (see also [1]). To do the job, however, dvipdfm uses GhostScript, and therefore this program must also be installed in the system.

2.3 Other ways to get a PDF file and bookmarks

There are other ways to obtain a final PDF output file, but they are all discouraged to use alongside with pittetd. For example, a common method

[4] In general, a `.pdf` is much larger—much less efficient in all respects—than the `.dvi`. This tendency is reinforced if PDFLaTeX is configured to run faster.

[5] Written by Mark A. Wicks.

is to use `dvips` to convert a file to the PostScript format, and then apply Acrobat Distiller on it. This method involves two conversions; links and especially bookmarks tend to have an erratic behavior.

Acrobat PDF Writer, a 'printer emulator' that 'prints' PDF files will, of course, ignore anything that cannot be printed, including bookmarks. And the other `.dvi⇝.pdf` converter in existence, `dvipdf`,[6] is not as widely available as dvipdfm.

On the other hand, `hyperref` is not the only way to create bookmarks with LaTeX. Older packages and systems exist, like VTeX, but since those are much less widely used, pittetd does not support them.

2.4 No hyperref

There is a third option concerning the creation of bookmarks and links in pittetd. Option `nohyperref` will prevent pittetd from taking care of almost all interactivity requirements, and the user is left the freedom (and the burden) to fulfill them by him- or herself.

This option might be more useful than it seems, because it allows users to use the `hyperref` package itself *their way*, not pittetd's. There are in the latter's code a series of minor, but substantial, modifications to `hyperref`, and some of the options with which the package is loaded are fixed. As a security measure, pittetd will not allow the user manually to load the package, unless `nohyperref` is specified. Thus, if a user wants to control `hyperref`'s behavior, this option will be necessary. Section 6.2 gives some directions on how to do this.

Also, if pittetd cannot run normally due to some complication in installation or configuration of `hyperref`, the `nohyperref` option provides a way to keep working on the contents of the document and worry about requirements later.

3 Installation

3.1 pittetd

The pittetd bundle is made of the following files:

[6] By Sergey Lesenko.

`pittetd.dtx`	Source for the class and this documentation.
`pittetd.ins`	Batch file for installation.
`pittetd.cls`	The pittetd class itself.
`pit10pt.clo`	Definitions for 10pt-size option.
`pit11pt.clo`	Definitions for 11pt-size option.
`pit12pt.clo`	Definitions for 12pt-size option.
`pitthesis.pit`	Patch for pitthesis class
`pittdiss.pit`	Patch for pittdiss class
`achicago.pit`	Patch for `achicago` package
`pittetd.dvi`	This documentation
`pittetd.pdf`	

All these files are individually downloadable from pittetd's webpage. It is only the two first files, however, that are necessary, for the rest can be extracted from them. To do this, the file `pittetd.ins` has to be processed with TEX (*not* LATEX); the documentation results from running LATEX (*not* TEX) on `pittetd.dtx`.[7]

It is the `.cls` and `.clo` files that conform the class itself, i.e., what LATEX needs to have access to. Under a system that, like most TEX implementations today, use the standard TEX Directory Structure (TDS), LATEX files are put in subdirectories of the `.../texmf/tex/latex` directory (for example, the standard classes are in `.../texmf/tex/latex/base`). So the best thing under such a system is to create a subdirectory for pittetd:

`.../texmf/tex/latex/pittetd`

and place there the `.cls` and `.clo` files. Likewise, the documentation (the file you are reading, `pittetd.dvi`) should be placed in

`.../texmf/doc/latex/pittetd`

and the source files (`pittetd.dtx` and `pittetd.ins`) in

`.../texmf/source/latex/pittetd`

The 'patches' should be placed in the same directory as the actual document's input files.

After placing the files in those directories, you might need to 'refresh' the database, i.e., to make TEX aware that a new class is loaded. This usually

[7] To get the index right, you have to run `makeindex` with `gind` style, saying, in the command line (and after a LATEX run on `pittetd.dtx`), `makeindex -s gind.ist pittetd`. Then a final `latex pittetd.dtx` produces the document with a well-formatted index.

appears as a command (or button, or window, etc.) of the implementation.[8]

For non-TDS systems, the suggestion is 'put the files where TEX can find them.' For example, search your disk for the standard classes (e.g., `article.cls`), and put the pittetd files where they are. Alternatively, you can simply put the pittetd files in the directory that contains the input files of your document.

3.2 Installation of other required packages

In addition, you will need at least the `hyperref` and `color` packages, and PDFTEX if you use `pdftex` option. Most likely, you already have those packages installed. Even so, it is possible that you do not have the file `pdftex.def`, which is part of only relatively recent distributions. This file, available from the pittetd's webpage, should be copied to the same directory where the file `color.sty` is (`.../texmf/tex/latex/graphics` in a TDS system).

`hyperref` is a package used by pittetd (unless, of course, the `nohyperref` option is used), so it has to be in the system. In the very unlikely case it is not already installed, you will need to download it from either CTAN (through http://www.tug.org) or the pittetd's webpage, and install it by running TEX (*not* LATEX) on the file `hyperref.ins`. This will extract the files and instruct you on where to place them (which, in any case, is analogous to the placement of pittetd files).

Likewise, `hyperref` uses other packages from the standard distribution of LATEX (notably, `color`). Installation of those packages is analogous.

3.3 PDFTEX, dvipdfm

Installation of PDFTEX and dvipdfm is a more complex matter. Again, several implementations, including TEXLive, MIKTEX, TETEX, FPTEX, and CMACTEX, have both tools pre-installed. In case your system does not have either or both of them, you can download the relevant files, and obtain installation directions, at CTAN (through www.tug.org). The PDFTEX manual, file `pdftex-s.pdf`, is available from the pittetd's webpage, and contains information on the installation of the program.

[8] With MIKTEX, for example, you should run the program 'MiKTeX Options.'

4 To keep in mind

The pittetd class has been designed to fully comply with the format guidelines for Pitt ETDs. Due to this, there are some particularities that might create conflict with LaTeX users' habits. This section warns and advises about those particularities. Decisions have been made with two priorities: to discourage uses that go against the FG, and to facilitate conversion from standard LaTeX classes.

4.1 Headings and captions

pittetd will automatically capitalize the title of the document and those of the chapters. However, section titles have to be capitalized by the user.[9]

On the other hand, since both sectional headings and captions for tables and figures must have entries in the bookmarks panel, they are subject to two substantial limitations: they cannot be long, and must consist only of ASCII characters.[10]

When building the bookmarks, `hyperref` will convert some simple LaTeX commands, but in general will ignore most of them. It also will crop everything that goes beyond the maximum length of a bookmark (that varies among PDF viewers; Acrobat Reader makes it 64 characters). There are two tools to handle these limitations in LaTeX: the `hyperref` command `\texorpdfstring` (section 6.1), and the optional argument to `\caption` (section 5.6.1).

4.2 Preliminaries

The series of preliminaries in a Pitt ETD differs substantially from a paper-based thesis/dissertation. The committee page has changed, and dedication and acknowledgements pages have been eliminated (under the assumption and recommendation that these should be part of the preface). The order was modified accordingly (so that the preface goes immediately before the text of the thesis itself).

Thus, the preliminaries are in principle limited to the following:

[9] This is because modification of the `\section` command in order to capitalize not only the title itself, but also the bookmark, although possible, would highly increase the probability of incompatibilities with other packages.

[10] That is why the FG recommend using words (not formulas) in titles and keeping captions "to one line if possible" (p. 9).

Title page
Committee Membership page
Copyright page (optional)
Abstract
Table of contents
List of Tables
List of Figures
Preface (optional)

Since all these preliminaries have their own commands in pittetd (see section 5.4), there is in principle no need nor place for non-numbered chapters (`\chapter*` commands). In fact, the starred version behaves exactly as the regular one. If there is a need for additional preliminary pages, the (on purpose) cumbersome command `\preliminarychapter` is available (see section 5.4.7).

4.3 Use of packages

Almost certainly authors of Pitt ETD's will need to load a wide and unpredictable variety of packages. Although pittetd has been coded with the premise not to 'invite' incompatibilities, it is possible that some of these packages will create clashes, for there is simply no way to claim universal compatibility with the hundreds of packages already available and with those to come.

However, *partial* compatibility can be (and supposedly has been) achieved. A survey carried out in April–May 2003 gave us a list of packages that are of common use in the Pitt community, and those have been taken into account in the writing of pittetd.

For the handling of possible incompatibilities arising in the future, the following policy has been designed. The user who suspects he or she has found a clash should contact the Pitt ETD Working Group and explain the problem, ideally e-mailing a copy of the input file(s). Hopefully in a reasonable amount of time, a 'patch' will be created that solves the problem. The patch takes the form of a file with extension `.pit`, downloadable from pittetd's webpage.

After the file has been downloaded and put where LaTeX can find it (the easiest way is to put it in the same folder as the document itself), it should
`\patch` be accessed. The command `\patch`, that takes the name of the package

as its argument, reads any patch that exists for it. For example, there is already a patch for the `achicago` package; to ensure the proper behavior of this package, the user should type, *after* `\usepackage{achicago}`, the command `\patch{achicago}`.

`\usewithpatch`

Alternatively, the command `\usepackage` can itself be replaced by `\usewithpatch`. When a package is invoked by means of `\usewithpatch`, pittetd will search the system for the corresponding patch; if it exists, it loads it; if not, nothing happens. Options to the package, as usual, are indicated by the optional argument `[`⟨*options*⟩`]`. The drawback of this mechanism is that several packages *cannot* be loaded at once (i.e., by comma-separating them, as in `\usepackage{color,graphicx}`); each must receive its own `\usewithpatch`. But using `\usewithpatch` ensures that pittetd will always look for a patch when loading a package.

The following paragraphs mention some LaTeX packages and tell whether they are supported or not by pittetd. For information on compatibility with the bibliographical styles and packages, see section 5.8.

4.3.1 Unsupported packages

Many popular LaTeX packages provide formatting features that either go against the FG or are already incorporated into pittetd. Therefore, it is assumed that such packages will *not* be loaded. These include `setspace`, packages for the handling of floating objects (such as `float`, `floatflt`), for variations of layout (`fancyhdr`, `fncychap`, `multicol`), and sectioning (`titlesec`, `tocbibind`). Using any of those packages might result in error messages, anomalies, and unpredictable output. Before reporting or trying to solve these problems, keep in mind that departmental approval is needed to include the features.

4.3.2 Supported packages

Some packages provide features that are legitimate in a Pitt ETD. Font packages, such as those in the PSNFSS collection (`times`, `bookman`, `palatino`, `newcent`, etc.) are perfectly compatible with pittetd. In fact, if CM fonts are desired, it is recommendable that the `ae` package is loaded.[11]

The packages of the American Mathematical Society (`amsmath`, `amsthm`, etc.) are supported. Likewise, all the packages in the standard distribution

[11] The AE fonts emulate CM, but are PostScript, not bitmap, fonts, which gives them a better quality for screen display.

of LaTeX (`color`, `graphicx`, `xspace`, `verbatim`, etc.) are supposed to work. `caption2` works miraculously fine. In general, packages that *provide*, as opposed to *override*, features, should work fine.

4.4 Related classes

Currently there are two LaTeX classes that produce theses and dissertations for Pitt, namely pitthesis (by Wonkoo Kim, 1999) and pittdiss (by Will Slaughter, 2003). The former was designed for paper-based documents, following requirements somewhat different from those of an ETD; the latter, on the contrary, was created with ETD in mind.[12] Many features are shared by those classes and pittetd—notably the creation of preliminaries—but the detailed mechanisms (command names and things like that) are different. This release of pittetd includes two 'patch' files that allow using pittetd with conventions from the other two classes, so that the user does not have to change every command (some will require handling, though; pittetd will warn or complain).

The patches are called through either

```
\patch{pittdiss}
```

or

```
\patch{pitthesis}
```

(the latter only two t's.) Having read the corresponding patch, pittetd will try to interpret pittdiss- or pitthesis-commands. Hopefully, most times it will succeed; in any case, it will issue warnings (or error messages in the `final` option) for things that *have* to be changed. For example, if acknowledgements are created with pitthesis's `acknowledgements` environment, pittetd will warn that now there is no separate preliminary for that, and that this section should be part of the preface.

As an extra safety measure when going from pitthesis or pittdiss to pittetd, all auxiliary files should be deleted before the first pittetd run. Also, it is very much recommendable to change `\bibliography` to `\safebibliography` since the beginning.

4.5 Stage and interaction

In order to facilitate the process of converting files from standard classes into pittetd, annoying error messages due to the particularities of pittetd

[12] It does not produce links or bookmarks, however.

have been avoided as much as possible. The 'stage' of the document is used to decide whether or not the differences should make stop the LaTeX run. The idea is that when the `draft` option is used, most problems are reported as 'Class Warnings' that do not stop the process. But if `final` is used (and eventually it *should* be used), more prominent error messages appear instead.

However, many packages (including the seminal `hyperref`, `color`, and `graphicx`) themselves operate differently when `draft` is specified. So for example, `hyperref` does not create links or bookmarks, and `graphicx` does not import external graphic files. But the user might want to see these features, still not worrying about detailed pittetd concerns. That is why an intermediate stage `semifinal` is introduced. The packages will work as usual, but pittetd will issue mostly warnings, not error messages. This is the default option.

Both `semifinal` and `draft` issue a final warning at the end of the job, reminding the user to run the document with `final`. As usual, moreover, `draft` makes overfull boxes visible.

The 'stage' option also governs pittetd's complaints about the preliminaries when information for the different pages is missing, when the order is wrong, etc.: with `draft` and `semifinal`, there will be a warning, while with `final` there will be an error message.

4.6 Auxiliary files

In addition to the regular auxiliary files (`.aux`, `.toc`, `.lot`, `.lof`, etc.), a run of pittetd involving all its features will produce two files: one with extension `.out` (written by `hyperref` for the 'outlines,' or bookmarks), and one with extension `.etd`, used by pittetd to decide some details. Input or other files created by the user should avoid these extensions. Also, for some implementations of LaTeX that provide a quick erasing of auxiliary files, it is advisable to configure this tool to include `.out` and `.etd` files.

4.7 PDF Document Info

pittetd offers the possibility of filling in the fields of Acrobat Reader's 'Document Info' dialog box. 'Title' and 'Author' are filled in with the data of the title page. 'Subject' and 'Keywords' are handled by additional commands (see section 5.5).

5 Using pittetd

5.1 Loading the class

The pittetd class is loaded by typing

`\documentclass[`⟨*options*⟩`]{pittetd}`

at the very top of the input file. Table 1 shows all the ⟨*options*⟩ available. Most of the options for conventional classes (i.e., the standard classes article, book, report, and similar ones like amsart and amsbook) have been disabled in pittetd. The document will always be typeset letter paper (8.5 × 11 inches), portrait, and one column.[13]

Characteristic	Available Options

Font size	`12pt` (default)
	`11pt`
	`10pt`

Stage
(see section 4.5)
	`final`
	`semifinal` (default)
	`draft`

Bibliography layout	`openbib` ('open' bibliographies).
Equations	`leqno` (equation numbers on the left) `fleqn` (flush-left displays)
Type	`phd` (dissertation, default) `ms` (M.S.'s thesis) `ma` (M.A.'s thesis)
Section numbering (see section 5.3.2)	`sectionnumbers` (default) `sectionletters`

PDF production
(see section 2)
	`dvipdfm`
	`pdftex`
	`nohyperref`

Table 1: Available options for pittetd

[13] Thus, options to modify these parameters, namely those for *a*) paper size (`letterpaper`, `legalpaper`, etc.); *b*) paper orientation (`portrait`, `landscape`); *c*) number of columns (`onecolumn`, `twocolumn`); and *d*) pagination (`oneside`, `twoside`; `openright`, `openany`; `titlepage`, `notitlepage`), are not implemented.

5.2 Font sizes and spacing

\Small \SMALL

The usual LaTeX commands are defined according to the font size option selected. In addition, the `\Small` and `\SMALL` commands work as in the classes amsart and amsbook, i.e., are equivalent to `\footnotesize` and `\scriptsize` respectively. See figure 1.

The text of a Pitt ETD has to be at least "one half-spaced, with the exception of long quotations, footnotes, bibliographical references, and the Index (if included), which may be single-spaced" (p. 7). A spacing of little more than one-half for regular text has been built in into pittetd; the text in footnotes and quotations has been set to single spacing. The user can always adjust the spacing in the usual way, `\renewcommand`'ing the

\baselinestretch

command `\baselinestretch`, so that

```
\renewcommand\baselinestretch{1.3}
```

increases the built-in spacing by a 30%—for *all* the text, footnotes included.

The spacing-scheme is achieved in pittetd by building it into the font sizes. Normal-size font (`\normalsize`) is one-half spaced, while all other sizes are single-spaced. The `quote` and `quotation` environments, as well as `\footnote`, all of which set a smaller font, produce thus single-spaced

\singlespace

text. An additional 'size' has been implemented, namely `\singlespace`, which produces regular-size, but single-spaced, text.

\smallskip \medskip \bigskip

The spacing command `\smallskip` is set to an amount of a single space; `\medskip` is a line (a little more than one and a half space); and `\bigskip` a double space.

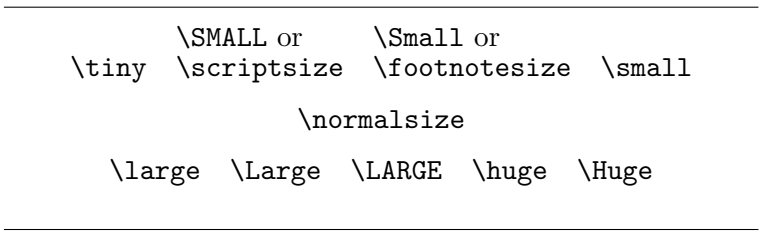


Figure 1: Font sizes

\acro

Adapted from the ltugboat class, pittetd implements the command `\acro`, that typesets its argument in a font smaller than the surrounding text. It is useful for all-uppercase acronyms like ETD (`\acro{ETD}`), UNICEF (`\acro{UNICEF}`), etc., which would be too large in regular size (compare ETD, CTAN, UNICEF); it is better than the direct `\small`, which is not good in contexts of font size other than normal.

5.3 Sectioning

5.3.1 Sectioning commands

\chapter
\section
\subsection
\subsubsection

The sectioning of a pittetd document is done through the usual commands `\chapter`, `\section`, `\subsection`, and `\subsubsection`. Note that `\part`, `\paragraph`, and `\subparagraph` are *not* implemented. The `\chapter` command takes care of capitalization of the title *both* in the text and in the bookmarks; however, since `\section` capitalizes in the text but not in the bookmarks, it is always advisable to capitalize manually.

Within preliminaries, the subdivisions `\section`, `\subsection`, and `\subsubsection` will produce neither a number nor a bookmark entry (`\chapter` is reserved for chapters in the body of the text; about additional preliminary 'chapters' see section 4.2). The starred variants `\chapter*`, `\section*`, etc., work exactly as the non-starred counterparts, although producing a warning.

All four sectioning commands have the usual optional argument, that contains the alternate version of the heading that appears in the table of contents. This, however, is implemented only for compatibility reasons, for the FG require that the table of contents lists the headings exactly as it appears in the text. The main reason why the optional argument could be used at all is that it permits to cope with the conversion of the text into ASCII text for the bookmarks, but that is best handled by the command `\texorpdfstring` (section 6.1).

On the other hand, there might be cases in which some letters must appear in lowercase even in headings (chemical elements is such a case). The command `\lowercase` works within the arguments to sectioning commands, and can be used for those cases.

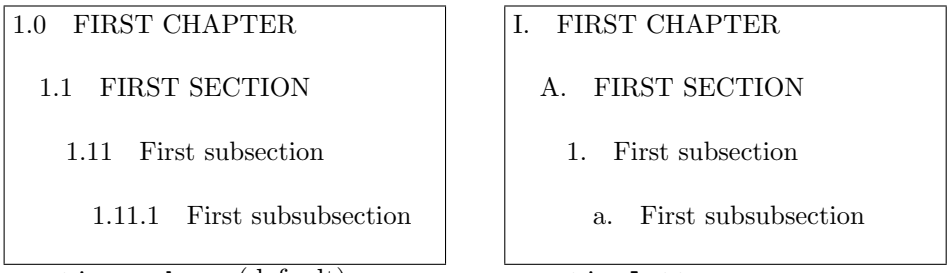
5.3.2 Numbering

The divisions of a Pitt ETD can be numbered in two ways, depicted in Figure 2. The first one is the one used by default (`sectionnumbers`); the user can specify pittetd's option `sectionletters` to use the second one. In this case, in addition, the labels for successive levels of the `enumerate` environment are also changed from their default appearance, to agree with the section numbering: the first level will be an uppercase roman numeral,

\regularenum

the second an uppercase letter, and so on. The user has the command `\regularenum` to revert to the usual appearances (namely arabic, letter,

roman, Letter).



`sectionnumbers` (default) `sectionletters`

Figure 2: The two possibilities for section numbering

5.4 The preliminaries

The first part of a Pitt ETD is made of the 'preliminaries.' They are created in pittetd with special commands that are the subject of the present section.

pittetd will keep track of the order in which the user typesets the preliminaries, and will warn or complain according to the 'stage' of the document (see section 4.5).

5.4.1 Title Page

`\maketitle` The title page is produced, as usual, by the command `\maketitle`, but involves several pieces of information in addition to `\title`, `\author` and `\date`, so it is only *deceivingly* similar to the same command in standard LaTeX classes. All efforts have been taken to prevent the differences to ruin the LaTeX run (making easier the conversion from other classes), but the user will eventually have to check it carefully.

`\title` The macro `\title` has an optional argument that sets the title of the document as it will appear in the 'Document Info' dialog box of Acrobat Reader. If no optional argument is given, the required argument will be used (however, bear in mind that only a limited portion of it will be visible). In order for this feature to work properly, `\title` must be issued in the preamble of the document. In the text, the title will be typeset uppercase. **Ex.:**

```
\title[An Anatomy of the World]{An Anatomy of %
  the World on texts by John Donne, %
for soprano and six instrumentalists}
```

\author The \author command works much the same as in standard LaTeX classes. Again, it should be issued in the preamble for the author's name to appear in the 'Document Info' dialog box. On the other hand, \thanks and \and are disabled. **Ex.:** \author{Federico Garcia}

The following macros set other information needed by pittetd to build the preliminaries. None of them is required unless final option is used. With semifinal and draft, a warning is issued informing of any missing commands.

These commands are analogous, but not identical, to additional commands in the classes pitthesis and pittdiss. Patches are available to facilitate conversion from those classes to pittetd.

\year The title page does not include the whole date, but only the year. By default, this is set to the current year; the user can optionally specify it with the command \year. **Ex.:** \year{2002}

\degree The information of the author's previous degrees is provided by the \degree command, and should contain the degree, institution, and year of the each degree. Several lines or degrees can be separated with \\.
Ex.: \degree{B.S. in Music (Composition),\\Bogot\'a, 2001}

\school The title page includes the text 'submitted to the graduate faculty of', followed by the school name. The user sets this name with the command \school. By default, the article 'the' is appended to the school name, but the user can change it with the optional argument.
Ex.: \school{Department of Mathematics}
Ex.: \school[]{FAS}
Ex.: \school[certain]{Other department}

\degreesought According to the option used (phd, ms, or ma), pittetd sets the value of \degreesought to either 'Doctor of Philosophy,' 'Master of Sciences,' or 'Master of Arts'. If desired, \renewcommand can be used to modify it.
Ex.: \renewcommand\degreesought{M. A. in Composition and Theory}

Figure 3 is the title page produced by the examples above.

5.4.2 Committee membership page

\makecommittee The \makecommittee command builds up the committee membership page. The author is typeset as it was in the Title Page (i.e., as is provided by the \author command); the school name comes initially from the \school command, although capitalized (see section 5.4.1 for these two commands). In some cases, the name of the school in the committee membership page

AN ANATOMY OF THE WORLD ON TEXTS BY JOHN DONNE FOR SOPRANO AND SIX INSTRUMENTALISTS

by

Federico Garcia

B.S. in Music (Composition)

Bogotá, 2001

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

M. A. in Composition and Theory

University of Pittsburgh

2002

Figure 3: Example of title page

should be different from the one that appears on the Title Page (for example, when it starts with 'Faculty'); the user can insert a new `\school` command right before `\makecommittee` (and after `\maketitle`) to control the second appearance.

In addition to that information, the committee membership page takes also the date and the committee members, which are provided with the next commands.

`\date` The `\date` command is intended for the date of the thesis/dissertation defense, which will appear after the text 'It was defended on'. The default value is `\today`. It can be omitted with `\date{}` (in whose case there will be no 'It was defended'), but a warning will be issued.

Ex.: `\date{May 15, 2003}`

`\committeemember` The list of committee members is typeset with information from one or more `\committeemember` commands (one for each member). The argument of `\committeemember` cannot contain more than one line.

Ex.: `\committeemember{N. Chimpsky, Ph.\ D., Professor}`

The *first* name will be treated as the thesis/dissertation advisor. When there are two advisors, the second one should be provided with the `\coadvisor` command.

In master's theses, inclusion of the rest of the committee is optional, but in Ph. D. dissertations it is required. Accordingly, if pittetd has been loaded with the `phd` option, it will require at least two `\committemember` commands, the requirement taking the form of a warning for `draft` and `semifinal` options, an error for `final`.

In any case, if just one member (the advisor) is listed, pittetd will omit the text 'approved by', only typesetting the advisor's name at the bottom of the page.

5.4.3 Copyright page

`\copyrightpage` Optionally, a copyright page can be appended immediately after the committee membership page, through the command `\copyrightpage`.

5.4.4 Abstract

`abstract` An abstract of no more than 350 words is required for every Pitt ETD. It is created as usual with the `abstract` environment:

```
\begin{abstract}
  ⟨text of the abstract⟩
\end{abstract}
```

The page will start with the title, the author, and the year of the document, followed by the text of the abstract.

Optionally, a list of keywords or descriptors can be appended at the end of the abstract. The keywords themselves have to be set in the preamble by the command `\keywords` (section 5.5). Then, an optional argument to the `abstract` environment sets the title of the list. For example, the command `\begin{abstract}[Keywords:]` produces, after the text of the abstract, the expression '**Keywords:**' followed by the contents of the previous `\keywords` command.

Some schools (including the School of Engineering) recommend that the word 'ABSTRACT' appears on the abstract page. pittetd provides for that requirement in the form of a starred version for the `abstract` environment:

```
\begin{abstract*}
  ⟨text of the abstract⟩
\end{abstract*}
```

Keywords can be appended to this kind of `abstract` in the same way.

5.4.5 Table of Contents, and Lists of Figures and Tables

`\tableofcontents` `\listoffigures` `\listoftables`

The table of contents and the lists of figures and tables are created with the usual LaTeX commands. If `hyperref` is used, the entries in these lists are links pointing to the corresponding page, and are included as bookmarks.

5.4.6 Preface

`\preface`

The preface is optional. If one is desired, the user needs only to type `\preface` followed by the text itself. Acknowledgements, dedication, etc., should be included in this preliminary. The preface is the only preliminary that is included in the table of contents.

5.4.7 Additional preliminaries

`\preliminarychapter`

As has been said, preliminaries in a Pitt ETD are in principle limited to those described above. Just for the sake of completeness, however, a command for additional preliminaries is implemented (and its use is discouraged) in pittetd:

`\preliminarychapter{`⟨*heading*⟩`}`

The ⟨*heading*⟩ will be both typeset and bookmarked, but not included in the table of contents). Sections within the additional preliminary will be unnumbered.

5.5 PDF Document Info

The 'Document Info' dialog box of Acrobat Reader includes information for title, author, subject, and keywords. pittetd will fill in these fields (if `hyperref` is used) with, respectively: the optional argument to the command `\title`; the `\author`; the `\subject` command; and the `\keywords` command. All four commands must be issued in the preamble for the information to go to the Document Info (although there is no error message if any or all are missing).

`\subject` `\keywords`

For example, the commands `\subject{Musical Composition}` and `\keywords{Music \& Text, John Donne, Vocal Music}` define the contents of the 'subject' and 'keywords' fields. The latter will, optionally, also be typeset at the end of the abstract (see section 5.4.4).

5.6 Main body

The way the main body of the document is typeset by LaTeX is very little modified by pittetd. As has been said, footnotes and quotations appear in a smaller font, and single-spaced. Within the `table` and `figure` environments, moreover, `\singlespace` is declared, so their contents appears single-spaced. To resort to one-half spacing, the declaration `\normalsize` is enough.

5.6.1 Numbering and captions for tables and figures

`\chapterfloats`

By default, figures and tables are numbered consecutively (1, 2, etc.), independently from the chapter. This can be changed with the `\chapterfloats` command, that has to appear before `\begin{document}`. In that case, figures and tables will be numbered within chapters (1.4, 2.5, etc., or I.4, II.5, etc.); pittetd reserves enough space for the figure or table number in the list of figures or tables (that might be something long like 'VIII.14') , but this requires several runs.

`\caption`

As has been mentioned, captions are subject to the limitations of bookmarking: they must be short and contain only ASCII text. In case this

poses problems, the optional argument to the `\caption` command is the best tool to deal with them:

`\caption[`⟨*alternate caption*⟩`]{`⟨*caption*⟩`}`

When present, it is ⟨*alternate caption*⟩, instead ⟨*caption*⟩, what is actually typeset in the list of tables or figures, and into the corresponding bookmark. So, if a long caption is necessary, it can be handled as in the following example (note the avoidance of `\cite` in the optional argument):

```
\caption[A modern ‘wave model’ of the Indo-European
languages according to Raimo~Antilla~(1972).]{A
modern ‘wave model’ of the Indo-European languages
according to \cite{r-a}. The numbers indicate 24
isogglosses (similarities) shared among different
Indo-European languages. Isogloss 1 indicates the
centum:satem split...
}
```

Refer also to section 6.1 for more details on `hyperref` conversion of TEX into ASCII text.

5.6.2 Cross references

When using `hyperref`, cross references created with the `\ref` and `\pageref` commands are interactive links. The package offers, as an alternative, the command `\nameref`, that is used exactly as `\ref`, but typesets the *name* of the chapter or section, instead of its number.[14] This kind of reference seems to be more consistent with interactivity (for, when a click is enough, the main motivation for an ordered numbering is called into question).

In any case, with pittetd, the `\nameref` command is slightly modified when it refers to an appendix: it does not produce the appendix's title, but its label (‘APPENDIX’, or ‘APPENDIX A’, etc.).

5.7 Appendices

`\appendix` The `\appendix` command tells pittetd that the following chapters (i.e., the following `\chapter` commands) are appendices. If there is only one appendix, its heading will be ‘APPENDIX’; if there are more, they will be numbered with capital letters, ‘APPENDIX A’, etc. pittetd needs a second run to know which way to follow.

[14] `hyperref` achieves this by means of invoking the `nameref` package.

5.8 Bibliography

5.8.1 BibTeX styles

This section applies only to documents whose bibliography is generated through BibTeX. Manually-created bibliographies (i.e., produced with the `thebibliography` environment) need no special warning to work properly with pittetd, which handles spacing after the FG (single space within entries; entries separated by one-half space).

As far as pittetd is concerned, there are three kinds of BibTeX styles (`.bst` files). The first kind includes the styles that limit themselves to ordering and formatting the different pieces of information within the bibliography entries (without modifying the appearance of the list as a whole). The vast majority of BibTeX styles, including the standard ones (`plain`, `unsrt`, `alpha`, `abbrv`), fall in this category. These styles pose no problem to pittetd, and nothing special is needed to fulfill the requirements of the FG.

The second group comprises those styles that, in addition to the individual entries, format the list as well. In general, styles that do not use bracketed labels ('[1]' or '[Cas44]') are part of this group, for they need to redefine the `thebibliography` environment to conform to the absence of such labels. They usually come with an associated package (`.sty` file) that takes care of this task. To ensure proper behavior when using these packages, pittetd offers the command `\safebibliography`. Its use is identical to that of `\bibliography`, and it tries to make a compromise between the style's conventions and the FG.[15]

`\safebibliography`

The last kind of bibliography styles is that of systems that modify aspects of formatting other than the final list of bibliographical references. All these systems have not only `.bst` files, but also substantial packages (`.sty`). `harvard`, `natbib` and `achicago` are common instances. When pittetd has a close encounter with packages of the third kind, there can be erratic behavior. It is recommended that `\safebibliography` is used instead of `\bibliography`, but this will probably not be enough. Since there is no general solution, the problems have to be treated individually, with patches, as explained in 4.3 above.

The three mentioned systems have already been tackled: `natbib` is an extremely well-written program, so that conciliating it with pittetd is easy

[15] It of course is not guaranteed that it will always succeed in doing so. If it does not, it is possible that the bibliography style is actually of the third kind.

and does not merit a separate patch file. No special treatment (other than using `\safebibliography`) will normally be needed.

On the other hand, `harvard` is a more complicated case, for the package creates interactive links. The `hyperref` package has support for `harvard`, but there is no way to foresee potential problems. It is strongly recommended if possible not to use this package, replacing it with `natbib`.

`achicago` poses other kinds of problems. It is an ambitious package that modifies things other than bibliography-related functions. For example, using this package, the effect of `\emph` will not be *italic*, but *slanted* shape; the `quote` and `quotation` environments are also modified, so that pittetd cannot set single spacing within them. Again, it is recommended to avoid this package, but in any case there is a patch available at pittetd's webpage, the file `achicago.pit`. It should be loaded saying `\patch{achicago}`.

Thus, through the means just explained, a broad range of bibliographical usages is supported by pittetd. Bracketed-labels referencing, being what LaTeX is designed for, can generally be used without restriction; for author-year referencing, `natbib` and `achicago` are supported; and for footnote referencing, the package `opcit` (available from CTAN) works fine if `\safebibliography` is used.

5.8.2 Citation packages

There are some packages that handle the way bibliographical references are handled within the text, rather than the way the entries of the final list are typeset. It is unfortunate that the package `cite`, that sorts the numbers of a multiple `\cite`, creates deep and quite un-traceable conflicts with `hyperref`. The package can be loaded, but it will have no effect. As a result, the `overcite` package will *not* sort the numbers either, although it will typeset them as superscripts (which, in addition, will be interactive links). `achemso` also causes problems, and it is recommended not to use it at all.[16] `chapterbib`, going against FG, is not supposed to be loaded.

[16] In `hyperref`'s documentation, Sebastian Rahtz admits not having been able to make hyper-bibliography robust, "since many styles redefine these things... Any or all of *achemso*, *chapterbib*, and *drftcite* may break." For the case of `cite`, I tried to make a compromise, sacrificing the interactivity of the bibliographical references to keep the effects of the package. But I got completely lost in the attempt... As Rahtz says, "life is too short," and I am not going to understand all the workings of `\@cite`, `\@citex`, `\@citen`, ... Sigh.

On the other hand, support for the `multibib` package, that allows multiple lists of references in the same document, is in progress. For the time being, the recommendation is to plan on writing one general bibliography if possible. In any case, several reference lists can be manually created (i.e., without using BibTeX).

5.9 The index

The code of pittetd defines the environment `theindex` to suit the FG, but otherwise exactly as standard classes define it. This means that the production of the index, be it manually or through *MakeIndex*, remains the same. `hyperref` offers an option to create a 'hyper-index,' whose page numbers are interactive links. However, the option is not very robust, and therefore pittetd uses `hyperref` but turns hyper-indexing off.

At the moment there is no support for multiple indexes to be generated automatically by *MakeIndex*, although several indexes can be manually created.

6 Using the `hyperref` package

This section is a very brief and incomplete guide to some extra features of the `hyperref` package that have not been explained before. Unfortunately, if something is missing to `hyperref`, it is documentation. Useful information is to be found in [2] and [3], but those documents are not intended for the average user. The present section is a translated adaptation of the relevant section in [4], to my knowledge the most complete (but still not comprehensive) user's guide on the package.

Section 6.2 provides a starting point to use `hyperref` in a way different of pittetd's default.

6.1 New user's commands

Certain character strings (notably the text of the bookmarks) are converted by `hyperref` into ASCII text, ignoring most LaTeX commands. In general, macros that expand into a piece of text (such as the `\LaTeX` command itself, the italic correction `\/`, or things like ‘ and ’) are appropriately handled. But math mode, for example, is completely ignored. The process

leads virtually never to an error message; warnings, however, are issued for every ignored token.

\texorpdfstring In any case, the user has a way to 'help' `hyperref` in the conversion, namely the command

`\texorpdfstring{`⟨*TEX text*⟩`}{`⟨*PDF text*⟩`}`

that can be used in sectioning commands or captions for figures and tables. For example, a caption with the text 'An H_2O molecule,' that would produce a bad bookmark entry, can be fixed by typing

```
\caption{\texorpdfstring{An H$_2$O molecule}{A water molecule}}
```

After this, the caption for the figure will feature 'H_2O' (both in the figure and the list of figures), but its bookmark will substitute '`water`'.

To create links other than those produced by the LaTeX commands `\ref`, `\pageref`, and `\cite`, `hyperref` makes available other commands. Only some of them will be mentioned here. See [2] for the rest.

\nameref The `\nameref` command works like `ref`, but creates a link with the chapter or section *name*. It is only applicable to sectioning commands.

\url The command `\url{`⟨*URL address*⟩`}` prints the ⟨*URL address*⟩ as a link that launches the local Internet surfer and leads to the corresponding page.

\hypertarget Analogous to `\label`, the command `\hypertarget{`⟨*key*⟩`}{`⟨*text*⟩`}` makes the ⟨*text*⟩ to be the target of a cross reference.

\hyperlink Analogous to `\ref`, the command `\hyperlink{`⟨*key*⟩`}{`⟨*expression*⟩`}` sets up an internal link whose target has been previously defined with `\hypertarget`.

\Acrobatmenu Through the command `\Acrobatmenu{`⟨*menu function*⟩`}{`⟨*text*⟩`}`, a ⟨*text*⟩ is typeset as a link that activates the ⟨*menu function*⟩ of Acrobat Reader (or Exchange). For a list of the available functions, see section 4 of [2].

6.2 Overriding pittetd's preferences

As has been said, pittetd loads `hyperref` with a fixed set of options. In order to access the package keeping control of it, it is needed to specify the `nohyperref` option for pittetd and then load `hyperref`:

```
\documentclass[nohyperref]{pittetd}
```

`\usepackage[`⟨*personal options*⟩`]{hyperref}`

This procedure is of course recommended only to users experienced with `hyperref`. A comprehensive list of `hyperref`'s options is given in [5]. Here is the list of options that pittetd uses by default (when allowed to):

```
letterpaper, colorlinks,
hyperindex=false
bookmarks, bookmarksnumbered, bookmarksopen,
citecolor=blue, urlcolor=blue
```

An option not used by pittetd that might be relevant is `backref`, that makes the bibliographical entries produce links to the sections in which the corresponding `\cite` appear (there is also the alternative `pagebackref`, with links leading to the *page* of the `\cite`'s).

In any case, it is always good to indicate the driver for `hyperref`, for example `pdftex` or `dvipdfm`, as an option to this package. In fact, when such an option is given to pittetd, all that is done by the latter is to pass it on to packages that need it, including `hyperref`, `graphicx` and `color`.

By loading `hyperref` manually, some automatic features of pittetd are lost: the bookmarks for the bibliography, the index, and the appendices; and the filling in of the 'Document Info' dialog box of Acrobat Reader. Figures and tables, however, will still create bookmarks. To get those bookmarks created was the thorniest issue in the writing of pittetd, and we have decided to keep this working even if the user has chosen to override pittetd's preferences about `hyperref` (see the code for `\listoffigures` and `\listoftables`).

`\pdfbookmark`

To create bookmarks additional to those that come from sections in the table of contents (or from the lists of figures and tables), `hyperref` provides the `\pdfbookmark` command:

`\pdfbookmark[`⟨*level*⟩`]{`⟨*bookmark text*⟩`}{`⟨*key*⟩`}`

where ⟨*level*⟩ is `0` for chapters, `1` for sections, and so on. The ⟨*key*⟩ is a unique name chosen by the user. The bookmark will be appended to the panel in the current position, and will point to the page of the text, in which `\pdfbookmark` appears. For more complicated instances (bookmarks that lead to a different location in the document, or that lead to different documents), see sections 5.2.4 and 7.2 of [3].

7 Before submitting

The pittetd LaTeX class is programmed to follow closely and consistently the FG. In general, the author of a thesis or dissertation needs not to be concerned about most of the formatting requirements (for example, checking the bookmarks and links one by one is unnecessary). However, this creates the danger of implying that nothing can go wrong. There are in fact some things beyond pittetd's control, and those things must be checked by the authors themselves (and will probably be checked closely by format reviewers). This section highlights the most common and likely problems.

Captions of tables and figures. Captions for tables should appear at the top of the table, while those for figures go at the bottom. pittetd does not force nor check this requirement.

Captions as bookmarks. Very long captions for tables and figures tend to be truncated when converted to bookmarks. Also, LaTeX constructions (like formulas, cite commands, etc.) are lost. Sections 5.6.1 and 6.1 show two ways of dealing with these limitations.

Capitalization of sections. The section titles are capitalized by pittetd in the text, but *not* in the bookmarks. The best thing is to provide `\section` with an already-all-capitals argument.

The `final` option. Before submitting it is always very important to run the document with `final` option (i.e., adding '`final`' to the list of options to `\documentclass`). This will catch and make evident any problems in the preliminary pages. See section 4.5.

Bad line breaks. Sometimes LaTeX cannot break a paragraph into lines satisfactorily. The result is one (or more) 'overfull' lines, that stick to the right of the margin. LaTeX always gives a warning about each and every overfull, and these can be seen in the `.log` file. This file, a plain-text file, can (and should) be read for overfull and other kinds of warnings. Overfull warnings start with the text '`Overfull \hbox in paragraph`'.

Bad page breaks. Similarly, LaTeX issues an 'underfull' warning for bad page breaking—when it is able to recognize it. But sometimes LaTeX will break a page just after a heading, which is wrong. The best way to check page breaking is to make the pages fit the screen and go scanning quickly page by page (PgDn).

Warnings. There are also warnings about other things, such as incomplete cross references, undefined `\cite`'s, etc., which are important to fix. The warnings are all collected in the `.log` file, and usually reveal at least one problem that had not been noticed before. It is not good to neglect reading this file; getting it to report no problems should be the crowning, final step in the thesis/dissertation production.

References

[1] Federico Garcia, *Comments on using LaTeX for theses*, July 2003, file `comments.dvi` or `comments.pdf`, available at pittetd's webpage.

[2] Sebastian Rahtz, *Hypertext marks in LaTeX: the hyperref package*, June 1998, file `manual.pdf`, part of the `hyperref` package distribution. Available at pittetd's webpage.

[3] Heiko Oberdiek, *PDF information and navigation elements with hyperref, pdfTeX, and thumbpdf*, paper at EuroTeX'99. File `paper.pdf`, part of the `hyperref` package distribution. Available at pittetd's webpage.

[4] Rodrigo De Castro, *El Universo LaTeX*, 2nd. edition, Bogotá, Universidad Nacional de Colombia, 2003.

[5] Sebastian Rahtz, *hyperref package options*, October 1999, file `options.pdf`, part of the `hyperref` package distribution. Available at pittetd's webpage.

8 The program

DOCSTRIP modules: `class`, `10pt`, `11pt`, `12pt`, `pittdiss`, `pitthesis`, `achicago`

8.1 Identification

```
1 ⟨*class⟩
2 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
3 \ProvidesClass{pittetd}[2004/08/17 v1.618
4     University of Pittsburgh ETD
5     (Electronic Thesis & Dissertations) class]
```

8.2 Declarations

```
\newwrite\@etdaux
\newif\if@errors
\newif\if@keywords
\newif\if@tables
\newif\if@figures
\newif\if@mainmatter\@mainmattertrue
\newif\if@hyper@ref\@hyper@reftrue
\newif\if@secletters
\newif\if@yeargiven\@yeargivenfalse
\newif\if@dategiven\@dategiventrue
\newlength\@chapterl
\newlength\@sectionl
\newlength\@subsectionl
\newlength\@subsubsectionl
\newlength\@singleline
\newlength\@presubs
\newlength\@presubsub
\newlength\@floatl
\newdimen\bibindent
\newlength\abovecaptionskip
\newlength\belowcaptionskip
\newcounter{@members}\setcounter{@members}{0}
\newcounter{@addprel}\setcounter{@addprel}{8}
\newcounter{@appno}\setcounter{@appno}{0}
\newcounter{chapter}
\newcounter{section}[chapter]
\newcounter{subsection}[section]
\newcounter{subsubsection}[subsection]
\newcounter{paragraph}
\newcounter{subparagraph}
\newcounter{figure}
\newcounter{table}
\newtoks\@expectedprelim
\newtoks\@committee
\newtoks\@coadv
```

8.3 Options

These come pretty unchanged from the standard classes:

```
\newcommand\@ptsize{}
\DeclareOption{10pt}{\renewcommand\@ptsize{0}}
\DeclareOption{11pt}{\renewcommand\@ptsize{1}}
\DeclareOption{12pt}{\renewcommand\@ptsize{2}}
```

```
\DeclareOption{leqno}{\input{leqno.clo}}
\DeclareOption{fleqn}{\input{fleqn.clo}}
\DeclareOption{openbib}{%
    \AtEndOfPackage{%
        \renewcommand\@openbib@code{%
            \advance\leftmargin\bibindent
            \itemindent -\bibindent
            \listparindent \itemindent
            \parsep \z@
            }%
        \renewcommand\newblock{\par}}}
```

Now for pittetd's own options.

```
\newcommand\t@or@d{}
\newcommand\@this{}
\newcommand\@@@degree{}
\DeclareOption{phd}{\renewcommand{\t@or@d}{phd}%
    \renewcommand\@this{Dissertation}%
    \renewcommand\@@@degree{PhD}%
    \def\@degreesought{\textbf{Doctor of Philosophy}}%
    \def\@advisor{Director}}
\DeclareOption{ma}{\renewcommand{\t@or@d}{ma}%
    \renewcommand\@this{Thesis}%
    \renewcommand\@@@degree{M.A.}%
    \def\@degreesought{\textbf{Master of Arts}}%
    \def\@advisor{Advisor}}
\DeclareOption{ms}{\renewcommand{\t@or@d}{ms}%
    \renewcommand\@this{Thesis}%
    \renewcommand\@@@degree{M.S.}%
    \def\@degreesought{\textbf{Master of Sciences}}%
    \def\@advisor{Advisor}}
\DeclareOption{sectionnumbers}{\@secletterstrue}
\DeclareOption{sectionletters}{\@secletterstrue}
```

The next are the 'stage' options. By declaration, `\if@errors` is false, so only `final` is required to mention it explicitly. See section 8.4 for the actual effect of this switch.

```
\DeclareOption{draft}{\setlength{\overfullrule}{5pt}}
\DeclareOption{semifinal}{\setlength{\overfullrule}{5pt}}
\DeclareOption{final}{\setlength{\overfullrule}{0pt}%
    \@errorstrue}
```

The following `\let`'ting of `\nameref` to `\ref` provides for the case that a document that has been processed all the time with `hyperref` is suddenly

changed to `nohyperref`. If `hyperref` (the package) is to be loaded, it will redefine the command (fortunately it is defined with `\def` in `nameref.sty`; for a not-as-lucky case, see the handling of `\texorpdfstring` below).

```
\DeclareOption{nohyperref}{\let\nameref\ref\@hyper@reffalse}
```

Note that the 'options' `pdftex` and `dvipdfm`, advertised so much in the documentation above, are not actually defined in `pittetd.cls`. Because of LaTeX options mechanism, they will be simply passed on to any package that is loaded.

Okay. Default options, and processing of any non-default:

```
\ExecuteOptions{12pt,semifinal,hyperref,phd}
\ProcessOptions
```

8.4 Stage and error handling

`\@linemessage` Through `\@linemessage` and `\@nolinemessage`, whose actual performance depends on the 'stage' option, most problems will only warn, unless the `final` option has turned `\if@errors` true.

```
\newcommand\@linemessage[2]{%
    \if@errors
        \ClassError{pittetd}{#1}{#2}
    \else
        \ClassWarning{pittetd}{#1}
    \fi}
```

`\@nolinemessage`

```
\newcommand\@nolinemessage[2]{\if@errors
    \ClassError{pittetd}{#1}{#2}\else
    \ClassWarningNoLine{pittetd}{#1}\fi}
```

8.5 Initialization

First, let's get the font size done: one file for each of the three available sizes. The length `\@singleline` is an important one: it is the space between two lines of text, which is exactly 1.5 of the regular spacing in a standard LaTeX class (according to size). `\normalsize` is the only size that really sets everything up, and the only whose spacing is more than single line. The others modify only the font size itself, and are just a little more than

single-spaced (as are all sizes in standard classes). Other things that depend on the font size are set in these files.

```
92 ⟨/class⟩
93 ⟨*10pt⟩
94 \setlength{\@singleline}{18\p@}
95 \renewcommand\normalsize{%
96     \@setfontsize\normalsize\@xpt{18}%
97     \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
98     \abovedisplayshortskip \z@ \@plus3\p@
99     \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
100    \belowdisplayskip \abovedisplayskip
101    \let\@listi\@listI}
102 \newcommand\small{\@setfontsize\small\@ixpt{11}}
103 \newcommand\footnotesize{\@setfontsize\footnotesize\@viiipt{9.5}}
104 \newcommand\scriptsize{\@setfontsize\scriptsize\@viipt\@viiipt}
105 \newcommand\tiny{\@setfontsize\tiny\@vpt\@vipt}
106 \newcommand\large{\@setfontsize\large\@xiipt{14}}
107 \newcommand\Large{\@setfontsize\Large\@xivpt{18}}
108 \newcommand\LARGE{\@setfontsize\LARGE\@xviipt{22}}
109 \newcommand\huge{\@setfontsize\huge\@xxpt{25}}
110 \newcommand\Huge{\@setfontsize\Huge\@xxvpt{30}}
111 \newcommand\singlespace{\@setfontsize\singlespace\@xpt\@xiipt}
112 \setlength\footnotesep{6.65\p@}
113 \setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
114 \setlength\parindent{15\p@}
115 ⟨/10pt⟩

116 ⟨*11pt⟩
117 \setlength{\@singleline}{20.4\p@}
118 \renewcommand\normalsize{%
119     \@setfontsize\normalsize\@xipt{20.4}%
120     \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
121     \abovedisplayshortskip \z@ \@plus3\p@
122     \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
123     \belowdisplayskip \abovedisplayskip
124     \let\@listi\@listI}
125 \newcommand\small{\@setfontsize\small\@xpt\@xiipt}
126 \newcommand\footnotesize{\@setfontsize\footnotesize\@ixpt{11}}
127 \newcommand\scriptsize{\@setfontsize\scriptsize\@viiipt{9.5}}
128 \newcommand\tiny{\@setfontsize\tiny\@vipt\@viipt}
129 \newcommand\large{\@setfontsize\large\@xiipt{14}}
130 \newcommand\Large{\@setfontsize\Large\@xivpt{18}}
131 \newcommand\LARGE{\@setfontsize\LARGE\@xviipt{22}}
132 \newcommand\huge{\@setfontsize\huge\@xxpt{25}}
133 \newcommand\Huge{\@setfontsize\Huge\@xxvpt{30}}
```

```
\newcommand\singlespace{\@setfontsize\singlespace\@xipt{13.6}}
\setlength\footnotesep{7.7\p@}
\setlength{\skip\footins}{10\p@ \@plus 4\p@ \@minus 2\p@}
\setlength\parindent{17\p@}
⟨/11pt⟩

⟨*12pt⟩
\setlength{\@singleline}{21.75\p@}
\renewcommand\normalsize{%
    \@setfontsize\normalsize\@xiipt{21.75}%
    \abovedisplayskip 12\p@ \@plus32\p@ \@minus7\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
    \belowdisplayskip \abovedisplayskip
    \let\@listi\@listI}
\newcommand\small{\@setfontsize\small\@xipt{13.6}}
\newcommand\footnotesize{\@setfontsize\footnotesize\@xpt\@xiipt}
\newcommand\scriptsize{\@setfontsize\scriptsize\@viiipt{9.5}}
\newcommand\tiny{\@setfontsize\tiny\@vipt\@viiipt}
\newcommand\large{\@setfontsize\large\@xivpt{18}}
\newcommand\Large{\@setfontsize\Large\@xviipt{22}}
\newcommand\LARGE{\@setfontsize\LARGE\@xxpt{25}}
\newcommand\huge{\@setfontsize\huge\@xxvpt{30}}
\let\Huge\huge
\newcommand\singlespace{\@setfontsize\singlespace\@xiipt{14.5}}
\setlength\footnotesep{8.4\p@}
\setlength{\skip\footins}{10.8\p@ \@plus 4\p@ \@minus 2\p@}
\setlength{\parindent}{19\p@}
⟨/12pt⟩
⟨*class⟩
```

The right file is loaded next. For all sizes, the following `\let`'s are true.

```
\input{pitetd1\@ptsize.clo}
\let\Small\footnotesize
\let\SMALL\scriptsize
\let\bibindent\parindent
```

Now for some initial values. The first five are needed to format the table of contents and the lists of tables and figures, and are to be changed by the `\jobname.etd` file below. Oh!, `\normalfont` must come before for `em` to make sense as a length unit.

```
\normalfont
\setlength\@chapterl{1.5em}
```

```
\setlength\@sectionl{1.5em}
\setlength\@subsectionl{1em}
\setlength\@subsubsectionl{1em}
\setlength\@floatl{2.3em}

\newcommand\@phd{phd}
\newcommand\contentsname{CONTENTS}
\newcommand\listfigurename{LIST OF FIGURES}
\newcommand\listtablename{LIST OF TABLES}
\newcommand\bibname{BIBLIOGRAPHY}
\newcommand\indexname{INDEX}
\newcommand\figurename{Figure}
\newcommand\tablename{Table}
\newcommand\chaptername{Chapter}
\newcommand\appendixname{APPENDIX}
\newcommand\abstractname{}
\newcommand\convname{CONVENTIONS}
```

For the table of contents and related:

```
\setcounter{secnumdepth}{3}
\setcounter{tocdepth}{3}
\newcommand\@pnumwidth{1.55em}
\newcommand\@tocrmarg{2.55em}
\newcommand\@dotsep{4.5}
```

And here we go with the `\jobname.etd` file. It will tell pittetd whether there are tables and/or figures—switches `\if@figures` and `\if@tables`, both initially false—and the lengths `\@chapterl`, `\@sectionl`, etc., which hold the width of the widest label of each sectioning level and the widest float number (plus some extra space). The indentation of the sections in the table of contents is equal to `\@chapterl`; that of the lesser levels (lengths `\@presubs` and `\@presubsub`) is then calculated as the sum of the widths of higher levels.

These calculations are necessary for potentially long Roman numerals of chapters. For example, in pittesis, the number **VIII** collides with the title of the chapter. And once we have to fix this, why not do it with all the levels, in all the cases? Likewise, if figures and tables are numbered within chapters, it is not that unlikely that expressions like 'VIII.10' have to appear in the corresponding list. The width has to be kept track of.

```
\InputIfFileExists{\jobname.etd}{}{%
    \if@errors
        \ClassWarningNoLine{pittetd}{%
```

```
            No \jobname.etd file. Entering semifinal mode}%
        \@errorsfalse
    \fi}
\if@errors\else\AtEndDocument{\ClassWarningNoLine{pittetd}{%
    Remember to run the document with 'final' option}}\fi
\setlength\@presubs{1\@sectionl}
\addtolength\@presubs{1\@chapterl}
\setlength\@presubsub{1\@presubs}
\addtolength\@presubsub{1\@subsectionl}
\newcommand*\l@chapter{%
    \@dottedtocline{0}{\z@}{\@chapterl}}
\newcommand*\l@section{%
    \@dottedtocline{1}{\@chapterl}{\@sectionl}}
\newcommand*\l@subsection{%
    \@dottedtocline{2}{\@presubs}{\@subsectionl}}
\newcommand*\l@subsubsection{%
    \@dottedtocline{3}{\@presubsub}{\@subsubsectionl}}
```

Other parameters:

```
\setlength\arraycolsep{5\p@}
\setlength\tabcolsep{6\p@}
\setlength\arrayrulewidth{.4\p@}
\setlength\doublerulesep{2\p@}
\setlength\tabbingsep{\labelsep}
\skip\@mpfootins=\skip\footins
\setlength\fboxsep{3\p@}
\setlength\fboxrule{.4\p@}
\@addtoreset{equation}{chapter}
\renewcommand\theequation{\ifnum \c@chapter>\z@
    \thechapter.\fi\@arabic\c@equation}
```

Page style and initialization:

```
\pagestyle{plain}
\pagenumbering{roman}
\setcounter{footnote}{0}
\let\@openbib@code\@empty
```

And just in case some LaTeX commands need to know this:

```
\@twosidefalse
\@mparswitchfalse
\@twocolumnfalse
```

8.6 Counters and numbers

The issue here is the two kinds of section numbering, 1.1.1.1 or I.A.1.a. In the second case, the headings for each section is *not* to include the 'number' of the previous (I. First Chapter; but A. First section). This involves two subtleties concerning cross references in the text: firstly, a reference to 'section B' would mean nothing without the chapter 'number;' secondly, the 'number' bears a final period in headings, but we do not want the period to appear within the text.

`\period@or@not` `\gobble@or@not`

This is handled with the `\period@or@not` and `\gobble@or@not` mechanisms. In principle, both commands expand to nothing, but when `\@withperiod` or `\@withgobble` are declared (always locally in a group), they expand to period or to gobble.

```
\let\period@or@not\relax
\def\@withperiod{\def\period@or@not{.}}
\let\gobble@or@not\relax
\def\@withgobble{\let\gobble@or@not\@gobble}
```

`\thechapter` `\thesection` `\thesubsection` `\thesubsubsection`

Now section numbers have to be defined accordingly. A regular call to `\thesubsection`, such as that done by a `\ref` command, for example, will not produce a period nor gobble the previous stuff.

```
\renewcommand\thechapter{\if@secletters
    \expandafter\@Roman\c@chapter\period@or@not
    \else\@arabic\c@chapter\fi}
\renewcommand\thesection{\if@secletters
        \gobble@or@not{\thechapter.}\expandafter\@Alph
        \c@section\period@or@not
    \else\thechapter.\@arabic\c@section\fi}
\renewcommand\thesubsection{\if@secletters
        \gobble@or@not{\thesection.}\@arabic
        \c@subsection\period@or@not
    \else\thesection.\@arabic\c@subsection\fi}
\renewcommand\thesubsubsection{\if@secletters
        \gobble@or@not{\thesubsection.}\@alph
        \c@subsubsection\period@or@not
    \else\thesubsection.\@arabic\c@subsubsection\fi}
```

`\pdfstringdefPreHook`

We also have to instruct `hyperref` how to handle these commands when building a PDF string. We take the chance to avoid a series of warnings for `\ignorespace`, and also tell about `\acro`, which is defined below.

```
\newcommand\pdfstringdefPreHook{%
    \let\acro\relax
    \let\ignorespaces\relax
    \let\gobble@or@not\@gobble
    \def\period@or@not{.}}
```

\thefigure
\thetable
\chapterfloats

Next, floating-objects numbering. In principle it is consecutive, but the user might want to make it chapter-dependent. In implementing that option (as `\chapterfloats`), `\thechapter` cannot be used, since the `\period@or@not` would wrongly expand into a period in the bookmarks, creating double periods. And there is no way around it, so we have to use `\c@chapter` directly.

On the other hand, with `\chapterfloats`, floats in the appendices should be numbered 'A1,' not 'A.1.' This is achieved by using `\if@mainmatter`, since only `\appendix` sets `\@mainmatterfalse`.

```
\renewcommand\thefigure{\@arabic\c@figure}
\renewcommand\thetable{\@arabic\c@table}
\newcommand\chapterfloats{\@addtoreset{figure}{chapter}%
    \@addtoreset{table}{chapter}%
    \renewcommand\thefigure{%
        \if@mainmatter
            \if@secletters
                \expandafter\@Roman\c@chapter
            \else\@arabic\c@chapter
            \fi.%
        \else\expandafter\@Alph\c@chapter\fi
        \@arabic\c@figure}%
    \renewcommand\thetable{%
        \if@mainmatter
            \if@secletters
                \expandafter\@Roman\c@chapter
            \else\@arabic\c@chapter
            \fi.%
        \else\expandafter\@Alph\c@chapter\fi
        \@arabic\c@table}}
\@onlypreamble\chapterfloats
```

\regularenum

Finally, the labels for the four levels of the `enumerate` environment are set. In case the second numbering is used for sections, these labels are set to Roman, Alpha, arabic, alpha.

```
\newcommand\regularenum{%
    \renewcommand\theenumi{\@arabic\c@enumi}%
```

```
    \renewcommand\theenumii{\@alph\c@enumii}%
    \renewcommand\theenumiii{\@roman\c@enumiii}%
    \renewcommand\theenumiv{\@Alph\c@enumiv}}
\if@secletters
    \renewcommand\theenumi{\@Roman\c@enumi}
    \renewcommand\theenumii{\@Alph\c@enumii}
    \renewcommand\theenumiii{\@arabic\c@enumiii}
    \renewcommand\theenumiv{\@alph\c@enumiv}
\else\regularenum\fi
```

8.7 Layout parameters

This follows closely the standard classes.

```
\setlength\lineskip{1\p@}
\setlength\normallineskip{1\p@}
\setlength\parskip{0\p@ \@plus \p@}
\setlength{\smallskipamount}{%
    .25\@singleline \@plus 1\p@ \@minus 1\p@}
\setlength{\medskipamount}{%
    .5\@singleline \@plus 2\p@ \@minus 2\p@}
\setlength{\bigskipamount}{%
    1\@singleline \@plus 4\p@ \@minus 4\p@}
\@lowpenalty 51
\@medpenalty 151
\@highpenalty 301
\clubpenalty 250
\widowpenalty 250
```

But page layout, being fixed here, is set directly and not through options.

```
\setlength\paperwidth{8.5in}
\setlength\textwidth{6.5in}
\setlength\oddsidemargin{\z@}
\setlength\paperheight{11in}
\setlength\textheight{9in}
\setlength\topskip{1\@ptsize \p@}
\setlength\maxdepth{.5\topskip}
\setlength\footskip{.5in}
\setlength\topmargin{\z@}
\setlength\headheight{\z@}
\setlength\headsep{\z@}
\setlength\marginparwidth{\z@}
\setlength\marginparsep{\z@}
```

By the way, if the `showidx` package is loaded, we should allow room for `\marginpar`'s (lest a forest of overfull messages populates the `.log` file).

```
\AtBeginDocument{\@ifpackageloaded{showidx}{%
    \setlength\marginparwidth{1in}
    \setlength\marginparsep{2pt}}{}}
```

More layout: this follows the standard classes, with adapted values, and omitting the double-column parameters.

```
\setcounter{topnumber}{2}
\renewcommand{\topfraction}{.5}
\setcounter{bottomnumber}{1}
\renewcommand\bottomfraction{.3}
\setcounter{totalnumber}{3}
\renewcommand\textfraction{.2}
\renewcommand\floatpagefraction{.5}
\setlength{\@fptop}{0\p@ \@plus 1fill}
\setlength{\@fpsep}{8\p@ \@plus 1fill}
\setlength{\@fpbot}{0\p@ \@plus 1fill}
\setlength{\floatsep}{.5\@singleline \@plus \z@ \@minus 10.2\p@}
\setlength{\textfloatsep}%
    {1.5\@singleline \@plus \z@ \@minus 10.2\p@}
\setlength{\intextsep}{.5\@singleline \@plus \z@ \@minus 10.2\p@}
\setlength\abovecaptionskip{\@singleline}
\setlength\belowcaptionskip{\@singleline}
```

8.8 Font commands

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
\DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
\DeclareOldFontCommand{\sl}{\normalfont\slshape}{\@nomath\sl}
\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\@nomath\sc}
\DeclareRobustCommand*\cal{\@fontswitch\relax\mathcal}
\DeclareRobustCommand*\mit{\@fontswitch\relax\mathnormal}
```

Here comes the first thing we have to deal with for the `achicago` package, that makes `\emph` behave as `\textsl`, rather than `\textit`. For the use of patches, see section 8.10.

```
⟨/class⟩
⟨*achicago⟩
\makeatletter
```

```
\@ifundefined{achicago@patch}\relax\endinput
\renewcommand{\emph}[1]{{\itswitch #1}}%
⟨/achicago⟩
⟨*class⟩
```

\acro The \acro command is an adaptation from the one in the ltugboat class:

```
\@ifundefined{acro}{%
    \DeclareRobustCommand\SMC{%
    \ifx\@currsize\normalsize\small\else
    \ifx\@currsize\singlesize\small\else
    \ifx\@currsize\small\footnotesize\else
        \ifx\@currsize\footnotesize\scriptsize\else
        \ifx\@currsize\large\normalsize\else
        \ifx\@currsize\Large\large\else
        \ifx\@currsize\LARGE\Large\else
            \ifx\@currsize\scriptsize\tiny\else
            \ifx\@currsize\tiny\tiny\else
            \ifx\@currsize\huge\LARGE\else
            \ifx\@currsize\Huge\huge\else
                \small\SMC@unknown@warning
    \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
    }
    \newcommand\SMC@unknown@warning{\ClassWarning{pittetd}{%
        \string\SMC: unrecognized text font size
        command---using \string\small}}
    \newcommand\textSMC[1]{{\SMC #1}}
    \newcommand\acro[1]{\textSMC{#1}\@}}{}
```

8.9 Hyperref

We begin by loading hyperref; at the end of the preamble, the proper options are set. Here, the user will already have provided the information for the 'Document Info' dialog box (\@pdftitle, \@author, etc.). We take pride on pittetd having created the PDF file, too.

```
\if@hyper@ref
    \AtEndOfClass{%
        \usepackage[bookmarks,hyperindex=false]{hyperref}%
        \def\@pdfcreator{LaTeX, pittetd class, hyperref package}}
    \AtBeginDocument{%
        \hypersetup{letterpaper,bookmarksnumbered,%
            colorlinks,bookmarksopen,%
            linkcolor=blue,citecolor=blue,urlcolor=blue,%
            pdftitle=\@pdftitle,pdfauthor=\@author,%
```

```
            pdfkeywords=\@keywords,%
            pdfsubject=\the\subject}}
\fi
```

`hyperref` redefines `\maketitle` to implement the special footnotes for `\author` and so on. This is disabled here. Also, the switch `\@hyper@ref` is set to true `\AtBeginDocument`, in case the user has loaded it manually. If `hyperref` is not loaded, `\texorpdfstring` is not defined. Since it could be still used in the document (if it was ever typeset with `hyperref` in mind), we provide a definition it here.

```
\AtBeginDocument{\@ifpackageloaded{hyperref}{%
    \let\maketitle\HyOrg@maketitle
    \@hyper@reftrue}{%
    \@hyper@reffalse
    \let\texorpdfstring\@firstoftwo}}
```

8.10 Packages and patches

`\patch` `\usewithpatch`

A difficult decision about the file extension for the patches. For one thing, an automatic search mechanism required the file name to be the same as the package's. But then extensions like `.clo` or `.def`, already in use by LaTeX, could very well confuse pittetd, for it is possible that files called that way, but not designed for the class, are already installed in the system.

```
\newcommand*\patch[1]{\InputIfFileExists{#1.pit}{%
            \ClassInfo{pittetd}{Patch for #1 loaded}}{%
            \ClassInfo{pittetd}{No patch found for #1}}}
\newcommand*\usewithpatch[2][\@empty]{%
    \ifx\@empty#1\usepackage{#2}\else\usepackage[#1]{#2}\fi%
    \patch{#2}}
\@onlypreamble\patch
```

The patches for the pitthesis and pittdiss classes are given below (section 8.11.14), and serve as illustration.

8.11 Preliminaries

8.11.1 Commands for the information

`\title`

First of all, `\title` has to be redefined for the optional argument, but `\author` remains the same (it `\gdef`'s `\@author` to its argument). How-

ever, standard-classes command `\thanks` is disabled here: it gobbles its argument and issues a warning.

```
\renewcommand*{\title}[2][\@empty]{%
    \gdef\@title{#2}\ifx\@empty#1%
        \gdef\@pdftitle{#2}%
    \else\gdef\@pdftitle{#1}\fi}
\renewcommand{\thanks}[1]{\ClassWarning{pittetd}{%
    The argument of \string\thanks\space has been ignored.}}
```

`\date` For master's theses the committee may be omitted. I conclude that everything concerning the defense may be omitted too. So, pittetd needs a way to know whether it should typeset the text 'It was approved' or not, which means scanning the `\date` to see if it is void. In any case, default value is `\today`.

```
\renewcommand*{\date}[1]{%
    \sbox\@tempboxa{\normalfont #1}
    \ifnum\wd\@tempboxa>0\global\@dategiventrue
    \else\global\@dategivenfalse\fi
    \gdef\@date{\ #1\ }}
\edef\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}
\date{\today}
```

`\year` The year, on the contrary, must be present. Since it is not a standard LaTeX requirement, we provide for the user not having thought of it: it assumes the current year and makes sure to remember that it was not provided.

```
\edef\@year{\number\year}
\renewcommand{\year}[1]{%
    \gdef\@year{#1}\@yeargiventrue}
```

`\@degree` `\@school` And there are many other commands that regular LaTeX never defined. They have to be kept from stopping the run, so we initialize some of values to produce a warning (or error, according to stage).

```
\newcommand\@degree{\@nolinemessage{No \string\degree\space
    command}{It is required to list the author's previous
    degree(s). Use the \string\degree\space (or \string\previousdegree\space command.}}
\newcommand\@school{\@nolinemessage{No \string\school\space
    command}{It is required to name the school (and
    optionally the department). Use the \string\school\space
```

```
    command.}}
\newcommand\@@school{\relax}
```

\degree
\previousdegree

User's commands, if any, will override the previous definitions. `\previousdegree` is useful in case `\degree` is assigned other meaning by a package.

```
\newcommand{\degree}[1]{\gdef\@degree{#1}}
\let\previousdegree\degree
```

\school

`\school` defines one thing for the title page, another one for the committee page.

```
\newcommand\school[2][the]{\gdef\@school{#1\ #2}%
    \gdef\@@school{\uppercase{#2}}}
```

\degreesought
\subject
\keywords

Now the only information missing is the degree sought, already initialized, and 'Subject' and 'Keywords' for the Document Info dialog box. The former is best implemented as a token register (the user fills it in if types `\subject{...}`). Both commands should be in the preamble, before `hyperref` builds up the Document Info.

```
\newcommand\degreesought[1]{\def\@degreesought{#1}}
\newtoks\subject
\def\@keywords{}
\newcommand\keywords[1]{\gdef\@keywords{#1}}
\@onlypreamble\subject
\@onlypreamble\keywords
```

8.11.2 The order

\@enable
\@disable

The mechanism by which pittetd checks that the preliminaries are in the right order is the next: each preliminary 'disables' those that cannot be placed afterwards, and 'enables' the one or more than can follow. 'Disable' and 'enable' mean 'create the right warning/error:'

```
\newcommand\@enable[2]{%
    \global\expandafter\let\csname @#1message\endcsname\relax
    \global\@expectedprelim{#2}}
\newcommand\@disable[1]{%
    \global\expandafter\def\csname @#1message\endcsname{%
        \@linemessage{Something is missing or misplaced.
            I was expecting \the\@expectedprelim}{%
            The preliminaries should be placed in the next order:
            \MessageBreak title, committee, [copyright],
```

```
            abstract, [preface/foreword], tableofcontents,
            [listoftables], [listoffigures].\MessageBreak
            \MessageBreak I think something is either missing or
            misplaced in your document. I will typeset things in
            the given order, but you should double-check.}}}
```

When something is 'enabled,' the 'expected preliminary' (used by pittetd when complaining) is changed (it's the second argument to `\@enable`). Note that the enabled preliminaries and expected ones are not necessarily the same, because there are optional preliminaries.

`\@@disable` A second kind of 'disable' (`\@@disable`) is for lists of tables or figures with no tables or figures in it:

```
\newcommand\@@disable[2]{%
    \global\expandafter\def\csname @#1message\endcsname{%
        \@linemessage{There is no #2}{I have found no #2, so it
        makes no sense to me to include a list of them.
        Proceed, but expect an ugly heading followed by
        nothing...}}}
```

So, each preliminary has (and executes) its own `\@...message` command, which is a complaint or a relaxation, according to the enable-status. To initialize these commands, we need to:

```
\@enable{title}{\maketitle}
\@disable{committee}
\@disable{copyright}
\@disable{abstract}
\@disable{preface}
\@disable{table}
\@disable{lof}
\@disable{lot}
\@disable{conv}
```

8.11.3 Bookmarking the preliminaries

Every preliminary is bookmarked by default. This is done by placing an `\hypertarget` and writing directly into the bookmark file, `\@outlinefile` (the bookmarks of figures and tables are discussed below). Option `nohyperref` turns this off, but in a round-about way: the test is no `\if@hyper@ref`, but `\@ifundefined{@outlinefile}`, because if no driver (`pdftex` or `dvipdfm`) is loaded by the user, `hyperref` will not create bookmarks anyway (it does not know which convention to use).

8.11.4 The title

\maketitle

```
\newcommand\maketitle{\@titlemessage
    \if@dategiven
        \if@yeargiven\else
            \ClassWarningNoLine{pittetd}{%
                I assumed `\@year' for the year.
                You can use \string\year\space to change it}%
        \fi
    \fi
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.0}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.0}{TITLE PAGE}{}}}
    \thispagestyle{empty}%
```

I chose to ignore the '8 spaces' kind of stuff. That is more suited for typewriters than for TEX, and far less robust, for multiple-line things could cause the information to overflow the page if the directions are followed at face value. A large font is used, with an interline spacing of around '2 spaces,' which is what the FG ask the most for in the title page. But for the '2 spaces' from the the top, a `\vskip` is needed, and `1.3\@singleline` is about right (`\@singleline` $\simeq 3s/2 \Rightarrow 4/3$`\@singleline` $\simeq 2s$)!

```
    {\centering
        \fontsize\@xviipt{1.5\@singleline}\selectfont
        \null\vskip 1.3\@singleline
        {\bfseries\uppercase\expandafter{\@title}}\par
        \vfill
        by\par
        {\bfseries\@author}\par
        {\let\noexpand\relax\@degree}\par
        \vfill
        Submitted to the Graduate Faculty of\par
        \@school\ in partial fulfillment\par
        of the requirements for the degree of\par
        \@degreesought\par
        \vfill
        University of Pittsburgh\par
        \@year\par}
    \@enable{committee}{\makecommittee}
```

And, to save memory:

```
    \global\let\thanks\relax
    \global\let\maketitle\relax
    \global\let\title\relax
    \global\let\author\relax
    \global\let\and\relax}
```

8.11.5 The committee page

\@commline
\@committee

The interesting thing here is the handling of committee members. First of all, the user shouldn't be asked to count the members—that sounds like a task for LaTeX—and, second, the advisor could be just the first member mentioned.

```
\def\@commline{\ifhmode\\\else\relax\fi}
\def\committeemember#1{%
    \stepcounter{@members}
    \@committee\expandafter{\the\@committee#1\@commline}}
```

Thus the register `\@committee` is a list of members separated by `\@comline`. The latter expands into `\\` only when the list is actually typeset.

\committeepage

Now for the actual page (which is typeset only if the user provided at least one `\committeemember`).

```
\newcommand\makecommittee{\clearpage\@committeemessage
    \ifnum\value{@members}>0
        \@ifundefined{@outlinefile}{}{%
            \hypertarget{prelim.1}{}%
            \protected@write\@outlinefile{}{%
                \protect\BOOKMARK
                [0][]{prelim.1}{COMMITTEE MEMBERSHIP PAGE}{}}}
        \@committee\expandafter{\the\@committee\relax}%
        {\centering
        \normalsize\fontsize\f@size{1.3\@singleline}\selectfont
        \null\vskip 1.3\@singleline
        UNIVERSITY OF PITTSBURGH\par
        \@@school\par
        \vfill
        This \lowercase\expandafter{\@this} was presented\\
        by\\[.67\@singleline]\@author\\[.67\@singleline]
        \if@dategiven
```

```
            It was defended on\\
            \@date\\ \else\ClassWarningNoLine{pittetd}{%
            You really should provide the date of the defense
            (command \string\date)}\fi
        \ifnum\value{@members}>1
            \if@dategiven and \else It was \fi
            approved by\\
            \the\@committee
```

If more than one member is given, the advisor has yet to be found (the trick is done by `\@getadvisor`, defined below). If not, and the document is a dissertation, pittetd complains.

```
            \expandafter\@getadvisor\the\@committee
        \else
            \ifx\t@or@d\@phd\@linemessage{Just one committee
            member}{For Ph. D. dissertations, the whole committee
            must be included. For the moment, I will typeset only
            the advisor.}\fi
            \vfill\fi
```

However, support for two advisors has to be given.

```
        \@this\ \@advisor: \the\@committee\the\@coadv
        \vfill
        \newpage}
        \renewcommand\committeemember[1]{\ClassError{pittetd}{%
            Command \string\committeemember\space after
            \string\makecommittee}{The new name will just be
            ignored.}}
        \let\makecommittee\relax
    \else\@linemessage{No members for the committee}{%
        I will ignore the \string\makecommittee\space command
        until you give me some names.}
    \fi
    \@enable{copyright}{\begin{abstract}}%
    \@enable{abstract}{\begin{abstract}}}
```

`\@getadvisor` When `\@getadvisor` is called, it is followed by the whole committee, names separated by `\@commline`, and all followed by `\relax`. Thus this command can take only the first name. Note that by the moment `\@getadvisor` is called, the whole list of the committee has been typeset already, and that is why it can be redefined.

```
\def\@getadvisor#1\@commline#2\relax{\@committee{#1}}
```

\coadvisor The `\coadvisor` command simply redefines `\@coadv` (initially void) and `\@advisor`.

```
\newcommand\coadvisor[1]{%
    \renewcommand\@advisor{Advisors}%
    \@coadv{,\\#1}}
```

8.11.6 Copyright page

\copyrightpage

```
\newcommand\copyrightpage{\clearpage\@copyrightmessage
    \null\vfill
    {\centering\normalsize Copyright
        \textcopyright\ by \@author\\\@year\\}%
    \vfill\newpage
    \@enable{abstract}{\begin{abstract}}}
```

8.11.7 The abstract

The optional argument is the title for the list of Keywords. The construction has to be done through a switch `\@keywordstrue` because environment arguments can only be used for the opening part. pittetd appends a period to the list, if it is not already there.

```
\newenvironment{abstract}[1][\@empty]
    {\ifx\@empty#1\else\@keywordstrue\def\@tempcsa{#1}\fi
    \clearpage\@abstractmessage
    \@afterindentfalse
    \typeout{Abstract}%
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.2}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.2}{ABSTRACT}{}}}
    \null\nobreak\vskip 1in
    {\centering\normalfont
        {\bfseries\uppercase
        \expandafter{\@title}}\\[.3\@singleline]
        \@author, \@@@degree\\[.3\@singleline]
        University of Pittsburgh, \@year\par}
    \par\vspace{1\@singleline}\@afterheading}%
    {\if@keywords
        \medskip\begin{description}\item[\@tempcsa]\@keywords
```

```
        {\normalsfcodes\ifnum\spacefactor=\sfcode‘.\else.\fi}
        \end{description}\fi
    \@disable{copyright}%
    \newpage\@enable{table}{\tableofcontents}}
\newenvironment{abstract*}[1][\@empty]
    {\ifx\@empty#1\else\@keywordstrue\def\@tempcsa{#1}\fi
    \clearpage\@abstractmessage
    \@afterindentfalse
    \typeout{Abstract}%
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.2}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.2}{ABSTRACT}{}}}
    \null\nobreak\vskip 1in
    {\centering\normalfont
        {\bfseries ABSTRACT\\[.3\@singleline]
        \uppercase\expandafter{\@title}}\\[.3\@singleline]
        \@author, \@@@degree\\[.3\@singleline]
        University of Pittsburgh, \@year\par}
    \par\vspace{1\@singleline}\@afterheading}%
    {\if@keywords
        \medskip\begin{description}\item[\@tempcsa]\@keywords
        {\normalsfcodes\ifnum\spacefactor=\sfcode‘.\else.\fi}
        \end{description}\fi
    \@disable{copyright}%
    \newpage\@enable{table}{\tableofcontents}}
```

8.11.8 The table of contents

\tableofcontents

```
\newcommand\tableofcontents{\clearpage\@tablemessage
    \typeout{Table of Contents}%
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.4}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.4}{TABLE OF CONTENTS}{}}
    }
    \null\vskip 1in
    {\normalfont\bfseries\centering
        TABLE OF CONTENTS\par}
    \par\vspace{54\p@}%
    {\@withperiod\@withgobble\@starttoc{toc}}%
```

```
    \if@tables
        \@enable{lot}{\listoftables}%
    \else
        \@@disable{lot}{tables}
        \if@figures
            \@enable{lof}{\listoffigures}%
        \else
            \@@disable{lof}{figures}%
            \@enable{preface}{the first chapter}%
            \@enable{conv}{the first chapter}%
        \fi
    \fi\@disable{preface}}
```

8.11.9 Bookmarking tables and figures

So far things have been at most challenging. But when it came to bookmarking the tables and figures, research and hard thinking had to be done. The main problem is that the location (within the bookmarks panel) of the bookmarks for tables and figures does not reflect their location in the document. So the immediate approach, modifying the `figure` and `table` environments to make them append a bookmark, does not work (the bookmarks would be appended in the current position of the panel). On the other hand, modifying those environments clearly invites incompatibilities, so it is only the last resource.

`\prepare@bookmarks`

`hyperref` builds the bookmarks from information contained in the `.out` file. Each sectioning command, for example, appends a bookmark command to this file. So, the only way to get the bookmarks for floats in the right place is to make the `.out` file call another file where the table or figure bookmarks are compiled. At the beginning I thought that the figures and tables (actually, the `\caption` command) would append lines to these additional files. But that involved changing very basic LaTeX commands, and also having to deal with different `hyperref` drivers. And, after all, the information would be basically the same as that already compiled in the `.lot` and `.lof` files. So, the point is to redefine the commands there included to interpret those files in terms of bookmarks. That is what `\prepare@bookmarks` does:

```
\def\prepare@bookmarks#1{%
    \def\@prelim{#1}%
    \def\numberline##1{##1. }
    \let\\\relax
```

```
    \def\contentsline##1##2##3##4{%
       \pdfstringdef\@tempcsa{##2}%
       \@expandedBOOKMARK{##4}{\@tempcsa}{prelim.\@prelim}}}
\def\@expandedBOOKMARK#1#2{\edef\reserved@a{%
    \noexpand\BOOKMARK[1][-]{#1}{#2}}\reserved@a}
```

The `\let`'ting of `\\` to `\relax` is a due to a subtlety with strong bug flavor. When `hyperref` is converting TEX strings into PDF ones, `\escapechar` is defined to be the `\` character, by means of `\escapechar`\\`. But then, just before reading the bookmarks, `\escapechar` is `\let` to `\@gobble`. So `\escapechar`\\` expands into `\@gobble`\\`, which ultimately renders `\\`. LATEX then complains that "`there is no line here to end`." On the other hand, `\@expandedBOOKMARK` is needed because otherwise `#2` (`\@tempcsa`) is not expanded.

But, after this tricky preparation (which of course has to take place within a group), the `.out` line can be instructed to read and interpret the `.lot` and `.lof` files.

8.11.10 The list of tables

`\listoftables`

```
\newcommand\listoftables{\clearpage\@lotmessage
    \typeout{List of Tables}%
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.5}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.5}{LIST OF TABLES}{}%
```

Here:

```
            \protect\begingroup\protect\makeatletter
            \protect\prepare@bookmarks5
            \protect\@input{\jobname.lot}
            \protect\endgroup}}
    \null\vskip 1in
    {\normalfont\bfseries\centering
        LIST OF TABLES\par}
    \par\vspace{54\p@}%
    \@starttoc{lot}%
    \newpage
    \if@figures\@enable{lof}{\listoffigures}\else
```

```
        \@enable{preface}{the first chapter}%
        \@enable{conv}{the first chapter}%
    \fi}
```

The width for the table numbers in the list depends on the widest float number (`\@floatl`), which is calculated at the end of the job, and written in the `.etd` file. Here it is used:

```
\newcommand*\l@table{\@dottedtocline{1}{1.5em}{\@floatl}}
```

8.11.11 The list of figures

`\listoffigures` This is analogous.

```
\newcommand\listoffigures{\clearpage\@lofmessage
    \typeout{List of Figures}%
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.6}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.6}{LIST OF FIGURES}{}%
            \protect\begingroup\protect\makeatletter
            \protect\prepare@bookmarks6
            \protect\@input{\jobname.lof}
            \protect\endgroup}}
    \null\vskip 1in
    {\normalfont\bfseries\centering
        LIST OF FIGURES\par}
    \par\vspace{54\p@}%
    \@starttoc{lof}\newpage
    \@enable{preface}{the first chapter}%
    \@disable{lot}}
\let\l@figure\l@table
```

8.11.12 The preface

`\preface` The only difference with other preliminaries is that this makes it into the table of contents. `\addcontents` creates the bookmark automatically.

```
\newcommand\preface{%
    \clearpage\@prefacemessage
    \@afterindentfalse
    \typeout{Preface}%
    \addcontentsline{toc}{chapter}{\bfseries PREFACE}
```

```
    \null\vskip 1in
    {\normalfont\bfseries\centering
        PREFACE\par}
    \par\vspace{1.5\@singleline}
    \@enable{conv}{the first chapter}%
    \@afterheading}
```

8.11.13 Additional preliminaries

\preliminarychapter

```
\newcommand\preliminarychapter[1]{%
    \clearpage\@afterindentfalse
    \typeout{#1}
    \@ifundefined{@outlinefile}{}{%
        \hypertarget{prelim.\the\c@@addprel}{}%
        \protected@write\@outlinefile{}{%
            \protect\BOOKMARK
            [0][]{prelim.\the\c@@addprel}{#1}{}}
        }%
    \null\vskip 1in
    {\normalfont\bfseries\centering
        #1\par}
    \par\vspace{1.5\@singleline}
    \@afterheading}
```

8.11.14 Support for other Pitt classes

These two patches serve as an illustration of patches in general. The `\@ifundefined` line makes sure the patch is not loaded twice. Of course, the final `\let` is companion. `\makeatletter` and `\makeatother` are a needed security measure.

For pitthesis:

```
⟨/class⟩
⟨*pitthesis⟩
\makeatletter
\@ifundefined{pittthesis@patch}{\relax}{\endinput}
\let\degrees\degree
\def\degree#1{\renewcommand\degreesought{#1}}
\let\university\@gobble
\let\proposal\@gobble
\let\advisor\committeemember
```

```
\let\coadvisor\committeemember
\newenvironment{acknowledgements}{%
    \preliminarychapter{ACKNOWLEDGEMENTS}%
    \@linemessage{Acknowledgments should be included in the
    preface}{I'm complaining because this comes from the
    'pittthesis' class. You haven't been reading my
    warnings.}}{\relax}
\newenvironment{committeesignature}[1][5]{\relax}{\makecommittee}
\let\advisorname\@gobble
\let\chairpersonname\@gobble
\let\authortitle\@gobble
\newenvironment{appendices}{\appendix}{\relax}
\newenvironment{singleappendix}{\appendix}{\relax}
\let\pittthesis@patch\@empty
\makeatother
⟨/pitthesis⟩
```

For pittdiss:

```
⟨*pittdiss⟩
\makeatletter
\@ifundefined{pittdiss@patch}\relax\endinput
\let\disstitle\title
\let\dissauthor\author
\newcommand\diplomaone[1]{\gdef\@degree{#1}}
\newcommand\diplomatwo[1]{\xdef\@degree{%
    \@degree\noexpand\noexpand
    \noexpand\noexpand\noexpand\noexpand\noexpand\\#1}}
\newcommand\diplomathree[1]{\xdef\@degree{%
    \@degree\noexpand\noexpand\noexpand\\#1}}
\newcommand\diplomafour[1]{\xdef\@degree{\@degree\noexpand\\#1}}
\let\university\@gobble
\let\degree\degreesought
\let\degreeabr\@gobble
\let\dissyear\year
\newcommand\dissdate[1]{\date{#1, \@year}}
\let\doctype\@gobble
\newcommand\chair[1]{\@striptitle#1\relax}
\def\@striptitle#1:#2\relax{\committeemember{#2}}
\let\committeetwo\committeemember
\let\committeethree\committeemember
\let\committeefour\committeemember
\let\committeefive\committeemember
\let\committeesix\committeemember
\let\disstitlepage\maketitle
\let\committeepage\makecommittee
```

```
\newenvironment{nomenclature}{\preliminarychapter{NOMENCLATURE}
    \@linemessage{Nomenclature is not a recognized
    preliminary.}{I'm complaining because this comes from the
    'pittdiss' class. You haven't been reading my
    warnings.}}{\relax}
\let\appendices\appendix
\let\pittdiss@patch\@empty
\makeatother
```
⟨/pittdiss⟩
⟨∗class⟩

8.12 Sectioning

`\@startsection` Section commands in the preliminaries create non-numbered divisions. This initial definition makes it happen:

```
\def\@startsection#1#2#3#4#5#6{%
    \if@noskipsec \leavevmode \fi
    \par
    \@afterindentfalse
    \@tempskipa #4\relax
    \@tempskipa -\@tempskipa
    \if@nobreak
        \everypar{}%
    \else
        \addpenalty\@secpenalty\addvspace\@tempskipa
    \fi
    \@ifstar
        {\ClassWarning{pittetd}{Starred
            \expandafter\string\csname #1\endcsname*\space
            has been treated as
            \expandafter\string\csname #1\endcsname}%
        \@ssect{#3}{#4}{#5}{#6}}%
        {\@ssect{#3}{#4}{#5}{#6}}}
```

`\@startbody` `\chapter` When the first `\chapter` appears, it acts also as the starting point of the body of the document: the page numbering is set to arabic, and sectioning commands are redefined.

```
\def\@startbody{%
    \pagenumbering{arabic}%
    \global\def\@startsection##1##2##3##4##5##6{%
        \@getlength{##1}%
        \if@noskipsec \leavevmode \fi
```

```
        \par
        \@afterindentfalse
        \@tempskipa ##4\relax
        \@tempskipa -\@tempskipa
        \if@nobreak
            \everypar{}%
        \else
            \addpenalty\@secpenalty\addvspace\@tempskipa
        \fi
        \@ifstar
            {\ClassWarning{pittetd}{Starred
                \expandafter\string\csname ##1\endcsname*\space
                has been treated as
                \expandafter\string\csname ##1\endcsname}%
            \@dblarg{\@sect{##1}{##2}{##3}{##4}{##5}{##6}}}%
            {\@dblarg{\@sect{##1}{##2}{##3}{##4}{##5}{##6}}}}%
    \renewcommand\chapter{\clearpage\global\@topnum\z@
        \@afterindentfalse
        \@ifstar{%
            \ClassWarning{pittetd}{Starred
                \string\chapter*\space has been treated as
                \string\chapter}%
            \@dblarg{\@chapter}}{\@dblarg\@chapter}}}
\newcommand\chapter{\clearpage\@startbody\chapter}
```

8.12.1 Chapters and sections

`\@chapter` `\@@chapter`

The actual chapters invoke `\@chapter`. The deferring construction with `\@@chapter` makes sure the chapter's title gets capitalized in the table of contents.

```
\def\@chapter[#1]#2{%
    \refstepcounter{chapter}%
    \@getlengthchapter
    {\@withperiod\typeout{Chapter\space\thechapter}}%
    \uppercase{\@@chapter{#1}}%
    \null\vskip 1in
    {\@withperiod\normalfont\bfseries\centering
        \if@mainmatter\thechapter\if@secletters\else.0\fi\quad\fi
        \uppercase\expandafter{#2}\par}
    \par\vspace{2\@singleline}\@afterheading}
\def\@@chapter#1{\addcontentsline{toc}{chapter}{{\bfseries
    \protect\numberline{\thechapter\if@secletters\else.0\fi}#1}}}
```

\section
\subsection
\subsubsection

The rest of sectioning commands invoke `\@startsection`. Note that all `\@withperiod`'s and `\@withgobble`'s will be expanded inside a group.

```
\newcommand\section{\@startsection{section}{1}%
    {\z@}{-2\@singleline \@plus -1ex \@minus -.2ex}%
    {1\@singleline \@plus .2ex}{\@withperiod\@withgobble%
    \centering\normalfont\bfseries\expandafter\MakeUppercase}}
\newcommand\subsection{\@startsection{subsection}%
    {2}{\z@}{-1\@singleline\@plus -1ex \@minus -.2ex}%
    {.5\@singleline \@plus .2ex}%
    {\@withperiod\@withgobble\normalfont\expandafter\bfseries}}
\newcommand\subsubsection{\@startsection{subsubsection}%
    {3}{\z@}{-1\@singleline \@plus -1ex \@minus -.2ex}%
    {-1em}{\@withperiod\@withgobble
        \normalfont\expandafter\bfseries}}
```

The following are disabled:

```
\let\part\@gobble
\let\paragraph\@gobble
\let\subparagraph\@gobble
```

\@getlengthchapter
\@getlength

Now, `\chapter`'s and `\...section`'s keep track of the widest numbers by means of the following commands:

```
\newcommand\@getlengthchapter{\begingroup\normalfont
    \@withgobble\@withperiod
    \settowidth\@tempskipa{%
        {\bfseries\thechapter\if@secletters\else0\fi}\quad
        \mbox{}}%
    \ifnum\@tempskipa > \@chapterl
        \global\@chapterl =\@tempskipa
    \fi
    \endgroup}
\newcommand\@getlength[1]{\begingroup\normalfont
    \@withgobble\@withperiod
    \stepcounter{#1}%
    \settowidth\@tempskipa{\csname the#1\endcsname.\ \mbox{}}%
    \ifnum\@tempskipa > \csname @#1l\endcsname
        \global\csname @#1l\endcsname = \@tempskipa
    \fi
    \addtocounter{#1}{-1}
    \endgroup}
```

8.12.2 Appendices

\appendix
\@appendix
\@@appendix

Only important thing: keeping track of the number of appendices, done through `@appno`. Minor details: redefinition of `\@currentlabel` to make `\nameref` refer to 'APPENDIX' instead of the title; and redefinition of `\thefigure` and `\thetable` to omit period after the chapter number, in case `\chapterfloats` has been issued.

```
\newcommand\appendix{%
    \@mainmatterfalse
    \setcounter{chapter}{0}%
    \setcounter{section}{0}%
    \ifnum 2>\value{@appno}
        \gdef\thechapter{}%
        \gdef\thesection{A.\@arabic\c@section}%
    \else
        \gdef\thechapter{ \@Alph{\c@chapter}}
    \fi
    \renewcommand\chapter{\clearpage\global\@topnum\z@
        \@afterindentfalse
        \protected@write\@auxout{}%
            {\string\stepcounter{\string @appno}}%
        \@ifstar{%
            \ClassWarning{pittetd}{Starred
                \string\chapter*\space has been treated as
                \string\chapter}%
            \@dblarg{\@appendix}}{\@dblarg\@appendix}}
    \let\appendix\relax}
\def\@appendix[#1]#2{\refstepcounter{chapter}%
    \xdef\@currentlabelname{\appendixname\thechapter}%
    \typeout{Appendix\thechapter}%
    \uppercase{\@@appendix{#1}}%
    \null\vskip 1in
    {\normalfont\bfseries\centering
        APPENDIX\thechapter\\[2\@singleline]
        \uppercase\expandafter{#2}\par}
    \par\vspace{2\@singleline}\@afterheading}
\def\@@appendix#1{\addcontentsline{toc}{chapter}{%
    {\bfseries APPENDIX\thechapter. #1}}}
```

There are no `\...matter` commands in pittetd.

```
\let\frontmatter\relax
\let\backmatter\relax
\let\mainmatter\relax
```

8.13 Floating objects

The two-column (starred) versions are implemented just for completeness. They might be used if the document is converted from other classes.

```
\def\fps@table{tbp}
\def\ftype@table{1}
\def\ext@table{lot}
\def\fnum@table{Table\nobreakspace\thetable}
\newenvironment{table}
    {\global\@tablestrue
        \singlespace\@float{table}}{%
```

The following is the main difference with standard classes: the widest label has to be tracked down. This could have been implemented as part of `\@makecaption`, instead of repeated four times here (in addition, there might be figures with no caption, and the test will still run). But I prefer to change here and not there, `\@makecaption` is probably redefined by many packages. `caption2` for one.

```
        \end@float
        \settowidth\@tempskipa{%
            \thetable\ \mbox{}}%
        \ifnum\@tempskipa > \@floatl
            \global\@floatl = \@tempskipa
        \fi}
\newenvironment{table*}
    {\global\@tablestrue
        \singlespace\@dblfloat{table}}{%
        \end@dblfloat
        \settowidth\@tempskipa{%
            \thetable\ \mbox{}}%
        \ifnum\@tempskipa > \@floatl
            \global\@floatl = \@tempskipa
        \fi}

\def\fps@figure{tbp}
\def\ftype@figure{1}
\def\ext@figure{lof}
\def\fnum@figure{Figure\nobreakspace\thefigure}
\newenvironment{figure}
    {\global\@figurestrue
        \singlespace\@float{figure}}{%
        \end@float
        \settowidth\@tempskipa{%
            \thefigure\ \mbox{}}%
```

```
        \ifnum\@tempskipa > \@floatl
            \global\@floatl = \@tempskipa
        \fi}
\newenvironment{figure*}
    {\global\@figurestrue
        \singlespace\@dblfloat{figure}}{%
        \end@dblfloat
        \settowidth\@tempskipa{%
            \thefigure\ \mbox{}}%
        \ifnum\@tempskipa > \@floatl
            \global\@floatl = \@tempskipa
        \fi}
```

`\@makecaption` This modified version of `\@makecaption` can be lost if some package redefines it, but it would not hurt much. Only the warning would be lost.

```
\def\@makecaption#1#2{%
    \vskip\abovecaptionskip
    \sbox\@tempboxa{#1: #2}%
    \ifdim \wd\@tempboxa >\hsize
        {\let\nobreakspace\space\ClassWarning{pittetd}{%
            The caption of #1 is longer than one line}}%
            #1: #2\par
    \else
        \global\@minipagefalse
        \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
    \fi
    \vskip\belowcaptionskip}
```

8.14 Lists

```
\setlength\leftmargini\parindent
\leftmargin\leftmargini
\setlength\leftmarginii{2.2em}
\setlength\leftmarginiii{1.87em}
\setlength\leftmarginiv{1.7em}
\setlength\leftmarginv{1em}
\setlength\leftmarginvi{1em}
\setlength\labelsep{.5em}
\setlength\labelwidth{\leftmargini}
\addtolength\labelwidth{-\labelsep}
\setlength\partopsep{\@ptsize\p@ \@plus 1\p@ \@minus 1\p@}
\addtolength\partopsep{1.5\p@}
\@beginparpenalty -\@lowpenalty
\@endparpenalty -\@lowpenalty
```

```
\@itempenalty -\@lowpenalty

\def\@listi{\topsep\smallskipamount
    \leftmargin\leftmargini
    \parsep \z@ \@plus 2\p@
    \itemsep \z@ \@plus 2\p@}
\let\@listI\@listi\@listi
\def\@listii{\leftmargin\leftmarginii
    \labelwidth\leftmarginii
    \advance\labelwidth-\labelsep
    \topsep \z@ \@plus 2\p@
    \parsep \z@ \@plus\p@
    \itemsep \parsep}
\def\@listiii{\leftmargin\leftmarginiii
    \labelwidth\leftmariniii
    \advance\labelwidth-\labelsep
    \topsep \z@ \@plus\p@
    \parsep \z@
    \itemsep \topsep}
\def\@listiv{\leftmargin\leftmarginiv
    \labelwidth\leftmarginiv
    \advance\labelwidth-\labelsep}
\def\@listiv{\leftmargin\leftmarginv
    \labelwidth\leftmarginv
    \advance\labelwidth-\labelsep}
\def\@listiv{\leftmargin\leftmarginvi
    \labelwidth\leftmarginvi
    \advance\labelwidth-\labelsep}

\newcommand\labelenumi{\theenumi.}
\newcommand\labelenumii{\theenumii.}
\newcommand\labelenumiii{\theenumiii.}
\newcommand\labelenumiv{\theenumiv.}

\newcommand\labelitemi{\textbullet}
\newcommand\labelitemii{\normalfont\bfseries\textendash}
\newcommand\labelitemiii{\textasteriskcentered}
\newcommand\labelitemiv{\textperiodcentered}

\newenvironment{description}
    {\list{}{\labelwidth\z@ \itemindent-\leftmargin
        \let\makelabel\descriptionlabel}}
    {\endlist}
\newcommand*\descriptionlabel[1]{%
    \hspace\labelsep\normalfont\bfseries #1}
```

8.15 Miscellaneous environments

```
\newenvironment{verse}
    {\let\\\@centercr
    \list{}{\itemsep \z@
        \itemindent -1.5em
        \listparindent\itemindent
        \rightmargin\leftmargin
        \advance\leftmargin 1.5em}
    \item\relax}
    {\endlist}
```

Since `quotation` and `quote` are redefined (against FG) by the `achicago` package, we must include the following definitions in the patch.

```
\newenvironment{quotation}\relax\relax
\newenvironment{quote}\relax\relax
⟨/class⟩
⟨∗class | achicago⟩
\renewenvironment{quotation}
    {\list{}{\topsep\medskipamount
        \listparindent 1.5em\relax
        \itemindent \listparindent
        \rightmargin \leftmargin
        \parsep \z@ \@plus\p@}
    \item\small\relax}
    {\endlist}
\renewenvironment{quote}
    {\list{}{\topsep\medskipamount
        \rightmargin\leftmargin}%
        \item\relax\small}
    {\endlist}
⟨/class | achicago⟩
⟨∗class⟩
```

8.16 Footnotes

```
\renewcommand\footnoterule{%
    \kern-3\p@
    \hrule\@width.3\columnwidth
    \kern2.6\p@}
\@addtoreset{footnote}{chapter}
\newcommand\@makefntext[1]{%
    \parindent 1em%
    \noindent
    \hb@xt@1.8em{\hss\@makefnmark}#1}
```

8.17 Bibliography

The `thebibliography` environment calls for `\@safebibliography`, which takes care of the heading, bookmark, spacing, etc.

```
\newenvironment{thebibliography}[1]{%
    \@safebibliography
    \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}}%
    \sloppy
    \clubpenalty4000
    \@clubpenalty \clubpenalty
    \widowpenalty4000%
    \sfcode`\.\@m}
    {\def\@noitemerr{\@latex@warning{Empty
        `thebibliography' environment}}%
    \endlist}
```

`\@safebibliography` The spacing within entries is `\singlespacing`. Between entries, `\itemsep` applies.

```
\newcommand\@safebibliography{%
    \renewcommand\chapter{\@ifstar{\@gobble}{\@gobble}}%
    \renewcommand\section{\@ifstar{\@gobble}{\@gobble}}%
    \renewcommand\@mkboth[2]{\relax}%
    \clearpage\null\vskip 1in
    \typeout{\bibname}%
    \@ifundefined{@outlinefile}{}{\phantomsection}%
    \addcontentsline{toc}{chapter}{{\bfseries\protect\bibname}}%
    {\normalfont\bfseries\centering\bibname\par}%
    \par\vspace{2\@singleline}%
    \singlespace
    \let\@listi\@listI
    \setlength{\itemsep}{.5\@singleline \@plus 4\p@}%
    \def\@listi{\topsep\z@
        \leftmargin\leftmargini
        \parsep \z@ \@plus 2\p@
        \itemsep .5\@singleline \@plus 2\p@}
    \@afterheading
    \let\@safebibliography\relax}
```

\safebibliography `\@safebibliography` is also used by `\safebibliography`.

```
\newcommand\safebibliography{\@safebibliography\bibliography}
\newcommand\newblock{\hskip .11em\@plus.33em\@minus.07em}
```

The `natbib` package handles the spacing not with `\itemsep`, but with the new `\bibsep`. As far as I know, this is the only thing we need to make the package work, so it is done here rather than in a patch.

```
\AtBeginDocument{\@ifundefined{bibsep}{}{%
    \setlength{\bibsep}{.5\@singleline \@plus 4\p@}}}
```

Support for the `achicago` package, on the contrary, goes to a patch:

```
⟨/class⟩
⟨*achicago⟩
\renewenvironment*{thebibliography}{%
    \@safebibliography
    \begin{list}{}{%
        \leftmargin\z@
        \advance\leftmargin\labelsep
        \advance\leftmargin\bibindent
        \itemindent -\bibindent
        \listparindent \itemindent
        \parsep \z@
        \itemsep .5\@singleline \@plus 4\p@}
    \sfcode`\.=\@m
    \let\@biblabel\Gobble
    \sloppy
    \clubpenalty4000 \widowpenalty4000
    }{%
    \def\@noitemerr{\@latex@warning{Empty
        `thebibliography' environment}}%
    \relax\end{list}}
\let\achicago@patch\@empty
\makeatother
⟨/achicago⟩
⟨*class⟩
```

8.17.1 The index

```
\newenvironment{theindex}{%
    \clearpage\typeout{\indexname}%
    \@ifundefined{@outlinefile}{}{\phantomsection}%
    \addcontentsline{toc}{chapter}{{\bfseries\protect\indexname}}%
    \twocolumn[\null\vskip 1in{\normalfont\bfseries\centering
```

```
        \indexname\par}\par\vspace{2\@singleline}]%
    \thispagestyle{plain}\parindent\z@
    \columnseprule \z@
    \columnsep 35\p@
    \parskip\z@ \@plus .3\p@\relax
    \small
    \let\item\@idxitem}
    {\clearpage}
\newcommand\@idxitem{\par\hangindent 40\p@}
\newcommand\subitem{\@idxitem \hspace*{20\p@}}
\newcommand\subsubitem{\@idxitem \hspace*{30\p@}}
\newcommand\indexspace{\par\vskip10\p@ \@plus5\p@ \@minus3\p@
    \relax}
```

8.18 The `.etd` file

The LaTeX run of the document has carefully taken note of certain values for the next run. It is now time to write them down in the `.etd` file.

```
\AtEndDocument{\immediate\openout\@etdaux=\jobname.etd
    \if@tables
        \immediate\write\@etdaux{\string\@tablestrue}
    \fi
    \if@figures
        \immediate\write\@etdaux{\string\@figurestrue}
    \fi
    \immediate\write\@etdaux{%
      \string\setlength{\string\@chapterl}{\the\@chapterl}}
    \immediate\write\@etdaux{%
      \string\setlength{\string\@sectionl}{\the\@sectionl}}
    \immediate\write\@etdaux{%
      \string\setlength{\string\@subsectionl}{\the\@subsectionl}}
    \immediate\write\@etdaux{%
      \string\setlength{\string\@subsubsectionl}{\the\@subsubsectionl}}
    \immediate\write\@etdaux{%
      \string\setlength{\string\@floatl}{\the\@floatl}}
    \immediate\closeout\@etdaux}
```

And that is it.