

Folder structure of the project

July 8, 2022

Contents

Contents	1	
1	Introduction and documentation	4
1.1	Roles and use of the files	5
1.1.1	The thesis: <code>main.tex</code>	5
1.1.2	The class elements: <code>yorkthesis.cls</code> and <code>thesisoptions.tex</code>	5
1.1.2.1	Margins	6
1.1.2.2	Chapter and sections style	7
1.1.2.3	The <code>draft</code> option	7
1.1.3	The personal stuff: <code>mypreamble.tex</code>	8
1.1.4	The other files	9
1.2	Using subfiles	10
1.2.1	Including graphics	10
1.2.2	Cross-referencing	11
1.2.2.1	Using the auxiliary file in a L ^A T _E X editor	11

1.2.2.2	Using cross-ref in overleaf	11
1.3	Accessibility	12

Todo list

■ add bib files/graphics to examples	4
■ Create the accessibility section	12

— 1 —

Introduction and documentation

This serves as an introduction and a documentation to the file structure of this project and to the interaction between the different files present. This file hierarchy tree is as follows (only source files are shown, compiled files are hidden):

```
[folders are denoted by a *]  
* Project  
  |-- main.tex          <- thesis tex file  
  |-- yorkthesis.cls    <- the premade class  
  |-- thesisoptions.sty  <- package holding the settings for the class  
  |-- mypreamble.tex    <- packages and precious macros  
  |--* C-Intro  
    |   |-- beg.tex       <- introductory chapter in its folder  
  |--* C-Beg  
    |   |-- beg.tex       <- another chapter  
  |--* C-Middle  
    |   |-- mid.tex       <- another chapter  
  |--* C-End  
    |   |-- end.tex       <- last chapter  
  |--* Bibfiles  
    |   |-- firstbib.bib  <- some bib file  
    |   |-- secondbib.bib <- another bib file
```

As can be seen on the diagram, the principal file of the thesis is `main.tex` which lies at the root of the `Project` folder, together with the class used for the thesis' layout and some helper files (an additional file not listed in the tree for overleaf is

add bib
files/graphics
to examples

described in section 1.2.2.2). The actual content of the thesis is split in subfiles lying in multiple sub-folders. The next section will describe the role of each file and how they should be used, while section 1.2 describes the relation to each other and the code they need to work together.

1.1 ROLES AND USE OF THE FILES

1.1.1 THE THESIS: `MAIN.TEX`

Even though the content of this file appears to be very small, this is the file that has to be compiled in order to obtain the full thesis. The motto of the `main` is “Divide and conquer”, by means of delegating big tasks to other files and elements and bringing them all together in one place. This file calls the `yorkthesis` class to create the document with only basic options, and then delegates the handling of its options to `thesisoptions.sty`, with the exception of the title page values which are to be filled in the main document.

The overall structure of the thesis is described in the `\begin{document} ... \end{document}` by first creating the first pages of the thesis corresponding to the title page, abstract, table of contents (and other list of elements), acknowledgment and author’s declaration, before calling onto subfiles to fill in the body of the document (selecting the order in which to include the work), and finishing by including the appendices and the bibliography.

Remark 1. The subfiles are called in the example by relative path, but there is a possibility to ignore that and simply use file names by uncommenting a section in the preamble. See the details in section 1.1.3.

1.1.2 THE CLASS ELEMENTS: `YORKTHESIS.CLS` AND `THESISOPTIONS.TEX`

The `yorkthesis` class is built upon the `memoir` class and apart from forcing some options by default (such as setting A4), a title page layout is defined to follow the university regulations, giving access to the appropriate commands to do so. Additionally, it creates a header for all pages, and gives access to new commands to set the page layout. In order to use it, you need to call `\documentclass[<options>]{yorkthesis}`, where `<options>` is a list of options compatible with the `memoir` class that will be

passed to it. Currently the only option passed is the `draft` option, which has multiple implications (details in section 1.1.2.3). In order to create the title page the following must be called in the `main.tex`:

- `\title{..}`: [required] title of the thesis;
- `\subtitle{..}`: [optional] subtitle (if appropriate);
- `\author{..}`: [required] your name;
- `\department{..}`: [required] department name, according to the list in the “depositing your thesis” section of the GRS website;
- `\qualification{..}`: [required] name of the qualification for which this work is submitted (PhD, MSc,...);
- `\submitdate{..}`: [optional] the month and year of the first submission (if not included, will default to the month and year at compilation).

The other options for the class are set in the file `thesisoptions.tex` and concern the following elements: fonts, margins, chapter style, sections style, list spacing and draft version options.

1.1.2.1 Margins

It is strongly recommended to use the commands provided to set the margins of the document instead of trying to deal with it by calling the `geometry` package (even though it should work too, I cannot insure that the class will react well to that). Margins are set by pair of opposite edges, so that the spine and foredge margins will be set in one command, and the upper and lower margins by a second one. The sides of reference when setting the margins are the spine (for the binding) and the upper one.

The horizontal margins can be set either by giving one length, two lengths or a length and a ratio as follows:

- `\horizontalmarginvalue{x}`: [`x` is a length with its unit] sets the spine and foredge margins to the value of `x`;
- `\horizontalmarginvalue[y]{x}`: [`x` and `y` are lengths with their unit] sets the spine margin to the value of `x` and the foredge margin to the value of `y`;
- `\horizontalmarginratio[r]{x}`: [`x` is a length with its unit, `r` is a ratio] sets spine margin to the value of `x` and the foredge margin to the value of `x*r`;

- (`\horizontalmarginsratio{x}` is equivalent to `\horizontalmarginsratio[1]{x}` and `\horizontalmarginsvalue{x}`).

The corresponding commands `\verticalmarginsvalue` and `\verticalmarginsratio` are used to set the upper and lower margins.

Remark 2. If no command is called for either the horizontal or the vertical margins, the default value for these will be used, which is set to `3cm` for the spine and foredge margins, and to `4cm` for the upper and lower margins, giving the inside frame for the content a similar ratio width/height to the original A4 page.

Remark 3. Only one of the horizontal commands and one of the vertical ones should be called to insure an appropriate behaviour. If both `\horizontalmarginsvalue` and `\horizontalmarginsratio` are called, the spine value will be set to the last of the two commands called, whereas the foredge value will be set from the resulting operation of `\horizontalmarginsvalue{x}` (I just didn't want to make another nested condition, so this is what happens when things go wrong).

1.1.2.2 *Chapter and sections style*

The general layout of the sections is set by using the `\headstyle{..}` command, which gives a special format to the headers of each of the *Chapter*, *Section*, *Subsection*, *Subsubsection*, *Paragraph* and *Subparagraph* subdivisions. Examples can be found in the `memoir` documentation in section 6.9. It is possible to override such definitions by calling the `\setSheadstyle{..}` command (where `S` is one of the subdivision shortnames), as can be seen in the `thesisoptions.tex` example file. Additionally, more examples of chapter header exists (see appendix B of the `memoir` documentation), and they can be called upon by using the `\chapterstyle{..}` command. Since this only define the style for a chapter, it is necessary to call it after the `\headstyle{..}` command if you do not want this setting to be overridden. You can also take the general structure of a chapter style and only redefine the style of the text for the chapter title and number as is done with the `\renewcommand*{\chapttitlefont}{..}` in the option file for this example.

1.1.2.3 *The `draft` option*

As mentioned earlier, this `draft` option is given when calling the class in the `main.tex` document, but is used to determine the behaviour of some other packages. This option also allows to take actions conditionally on if it is present or not. This

is what the conditional operations located at the end of the `thesisoptions.tex` file are about:

```
\ifdraftdoc
  %% Some actions occurring when 'draft' is set
\else
  %% Other actions if 'draft' is not set
\fi
```

which is currently used to make a watermark on the front page in draft mode, and call the `microtype` package otherwise.

Additionally, this example calls the `hyperref` package with the `final` option to override the fact that hyperlinks are not created in draft mode; whilst the `todonotes` package is called with the `obeyDraft` option which means that the todo notes on the side will only appear if the `draft` option is set for the class, and they will be disabled otherwise.

1.1.3 THE PERSONAL STUFF: `MYPREAMBLE.TEX`

Apart from the important bit at the top of this file, its content is globally up to you. You should put in there all the packages and macros you will want or need to use.

Since I have decided to use the `subfile` package to simplify my life (more on that in section 1.2), the first thing to do is to include the packages needed to work with that. We first need to add the `xr-hyper` package which will take care of cross references and their correct behaviour with the `hyperref` package, and then add the `subfiles` package. In order to use the references between different files we need an additional command that will make them available to the subfiles (the main file will have them anyway). After these few inclusions there are two sets of commands that are commented out and serve different purposes.

The first one is as follows:

```
\makeatletter
\def\input@path{ {dir1/}{dir2/} }
\makeatother
```

which allows to use file names in inclusion, instead of relative paths as long as all the folders of the project are added to this list by respecting the syntax. It

is currently commented out since this slows down the compilation time a lot and provide only small benefit. But if you are lazy, then this can be helpful. The second part concerns the use of *Overleaf*, as additional commands are needed for cross-referencing purposes (see section 1.2.2.2 for the details). Only uncomment these lines if you are using that website to write your thesis.

On top of the usual packages that relates to maths, some additional ones are listed in the example such as the `hyperref` package which creates hyperlinks into the outputed pdf in order to jump between sections, and the `todonotes` which allows you to create sticker notes in the margins about what needs to be done as well as giving you a list of these after the table of content by calling `\listoftodos` (check its documentation for more info). Your bibliography needs to be setup somewhere, so if you are using `.bib` files, this is the right place to include the package with all the options you would like, and then to add the files needed as a ressource, so that referencing a book like Knuth's TeXbook [`texbook`] becomes very easy. Notice the invocation with the `\subfix{..}` command (or the relative path in the Overleaf project as the subfix didn't seem to work there unfortunately...).

The `mypreamble` file is also the place to create the styling of your theorems, definitions, remarks and other scholia.

Lastly, all your macros for new commands should go there, so that they are all in one place and available to use everywhere.

Remark 4 (Technical implementation). The choice has been made to make to have this document as a plain `.tex` file instead of making it a package file (as `.sty`). This is a personal choice to avoid any interpretation of the commands listed there and to make the packages placed there too.

1.1.4 THE OTHER FILES

All the other files of the structure correspond to the actual content of the thesis separated between chapters. These are pretty standalone files that one would write in a classical `article` class style, with only a few elements that will be covered in the following section.

1.2 USING SUBFILES

The reason why I chose to use the `subfiles` package is that it allows you to include files into the main document, while keeping the possibility to compile them separately. That way you don't have to move back to the main file every time you want to compile something, you can compile only the desired section instead of the whole document, and you don't need to delete all the stuff outside of the `\begin{document}.. \end{document}` in the file storing your chapter content in order to make it compile from the main document.

How to do it is pretty straightforward and can be found on p.2–3 of the package documentation. In short, the subfiles, are using a special document class with the path to the main file by calling upon `\documentclass[./main]{subfiles}`. This does most of the job, since this will allow the file to compile by using all the preamble of the main file. In turn, when in `main.tex` the command `\subfile{..}` is invoked, it will only pick the content located between the `\begin{document}` and `\end{document}` commands. If you want to run some actions depending on if the file is included or if it is compiled by itself there is a command for it that you can include in the body of the document:

```
\ifSubfilesClassLoaded{
    % actions to do if compiling this file alone
}[
    % else condition, i.e. if we are compiling the main file
}
```

which can be used to only create a local titlepage, toc or bibliography which should not appear when compiling the main file (you can check the difference between the result of the compilation of the `intro.tex` file on its own compared to when it is integrated into the main).

1.2.1 INCLUDING GRAPHICS

In order to make sure that the path will not get broken when compiling the main file, you might need to use the `\subfix{..}` command around the filename so that the `subfiles` class can deal with that problem for us by doing the following: `\includegraphics[<options>]{\subfix{image}}`.

1.2.2 CROSS-REFERENCING

Most of the job of cross referencing is done by the `xr-hyper` package, but we need to help it a bit. This is the reason of the presence of `\externaldocument{\subfix{main}}` in the preamble. After that, when you compile one of the subfile, it will always read the auxiliary file generated by the `main` document to find the references and to include the appropriate number in the right places. For example, if we create an equation here that we will reference in Chapter 2:

$$\Omega(I) = \{(\lambda, \rho) \in \Lambda(I) \times P(I) \mid s(\lambda t) = (s\rho)t \quad \forall s, t \in S\}, \quad (1.1)$$

(which denotes the translational hull of a subsemigroup $I \subseteq S$), we already have the labels appropriately introduced.

1.2.2.1 Using the auxiliary file in a *L^AT_EX* editor

In order for the cross referencing to work in subfiles, you first need to compile the `main.tex` file. This will create the `main.aux` file which stores all the labels and associated values. Then compiling any subfile will use that auxiliary file and should give you access to all these references as expected.

1.2.2.2 Using cross-ref in overleaf

If you intend to use Overleaf, some additional steps can be needed as described in the documentation on the overleaf website. This is due to the fact that we cannot directly access the `.aux` files which contains the references we need as they are only stored in the cache and not in the file system. In order to do that, there are three necessary steps to accomplish this (see [here](#)):

1. in the file `mypreamble.tex` you need to uncomment the line that relates to the overleaf cross-referencing at the top of the file;
2. a new file named `latexmkrc` needs to be created containing the following code:

```
add_cus_dep( 'tex', 'aux', 0, 'makeexternaldocument' );
sub makeexternaldocument {
    if (!($root_filename eq $_[0]))
    {
        system( "latexmk -cd -pdf \"$_[0]\\" );
    }
}
```

which will force the creation of the auxiliary file of a dependency file called upon;

3. add the command `\myexternaldocument{main}` to the preamble of each subfile, which will force the availability of `main.aux`.

This process will allow the cross-referencing to happen, but will make the compilation time much slower, because it will require the compilation of the `main` file every time, which in turns ask all the subfiles to be compiled. If this is ok for you, then you can leave the settings as such and never touch this again.

However, there are ways to speed up the compilation process by using the fact that Overleaf keep the compiled files available in the cache. To do this, simply unset all of the elements listed above (so keeping the lines commented out, and not creating the `latexmkrc` file). Then, compile the `main.tex` once, which will set the `.aux` file in the cache, and compile the subfile as usual. You should be able to keep this cache set for a while, but remember that your browser might clean it at any point without you knowing it, so you would need to recompile the main for it to be working again. It might also happen that the first compilations of the subfile produce errors but it should resolves itself fairly quickly.

Remark 5. In the log you will notice that Overleaf warns you about labels being defined multiple times. This is normal because it is reading the labels set both in the `main.aux` file as well as those in the auxiliary file of the current file you are compiling. For this reason, you need to be extra careful not to use the same name for labels. A way to check is to use the autofill of Overleaf to try a `\ref{..}` with the label you are going to set, and if it is not suggested, then you should be fine to set it with `\label{..}`.

1.3 ACCESSIBILITY

Find the tools necessary to allow graphics and tables to be accessible.