Sistemska Programska Oprema - Latex Compiling

Rok Kos

January 15, 2019

Contents

1 Introduction **3**

1.1 Distributions and Compilers 3

1.2 Getting the desired output 3

2 Compiling process **4**

2.1 Log Files in LaTeX 4

3 Auxiliary Files in LaTeX **6**

3.1 Simple example 6

3.2 Labels and refs 7

3.3 Include and input files 8

4 Infinitely many compiler passes **9**

5 Build Tools **10**

5.1 Latexmk 10

References **11**

List of Figures

1 Latex compiling flow (from site [Ovea]) 4

2 Undefined Ref 5

3 First Run 5

4 Second Run 5

5 Citation Warning first run 5

6 Citation Warning second run 6

7 Label ref first run 7

8 Label ref second run 8

1 Introduction

So in this seminar, I will present how compiling in LaTeX works. This is the world's most popular typesetting system for scientific and technical documentation.

From now on when I will use word LaTeX I will refer to LaTeX2ε and not old LaTeX2.09 or new LaTeX3 which is currently in development.

But first, we need to clear some things first. LaTeX is built on top of TeX. TeX is low-level typesetting system/formatting engine created by Professor Donald Knuth. LaTeX provides all the macros (written in TeX) for formatting that are easy to use.

So basically you can imagine that LaTeX is more focused on content and TeX is more focused on formatting text and making it look the way you want.

1.1 Distributions and Compilers

TeX distribution is basically toolset that allows you to compile your LaTeX document. It consists of a set of programs and each major OS has it [pro] [Ovea]. Some popular and maintained distributions:

- **TeX Live** for Linux and other UNIX-like systems
- **MacTeX** redistribution of TeX Live for macOS
- **MiKTeX** for Windows
- **proTeXt** is based on MiKTeX

and there are also some that are not actively maintained [egr]:

- **teTeX** for Linux and other UNIX-like systems
- **fpTeX** for Linux and other UNIX-like systems
- **emTeX** for MS-DOS and OS/2
- **gwTeX** or macOS
- **OzTeXor** macOS
- **AmigaTeX** for the Amiga
- **PasTeX** for the Amiga

Each of these distributions supports compilers that can output three types of documents:

- **DVI** - Device independent file format. This is usualy output of **latex** compiler
- **PDF** - Portable Document Format. This is output of **pdflatex** compiler
- **PS** - PostScript format. We get this by compiling .tex file first to DVI or PDF and then converting it to PS.

Some other compilers that also exist are XeLaTeX and LuaLaTeX.

1.2 Getting the desired output

So how do we get our text from document.tex to PDF or DVI form? I will show an example on Linux with TeX Live distribution. It's easy as typing a command into the terminal (under the condition that you previously setup you distribution correctly) So we type in

```
$ pdflatex document.tex
```

for PDF output or

```
$ latex document.tex
```

for DVI output.

You might ask why I am doing this seminar if it's so easy. That I will explain this in the following chapters.

But before we continue I will just add one picture to show with which commands you can get which output. See the figure 1 below.

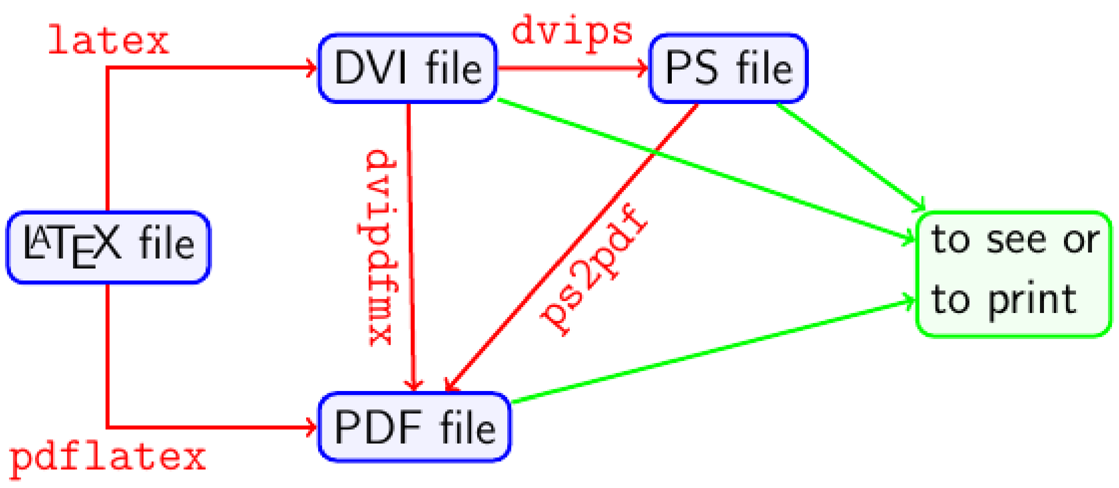


Figure 1: Latex compiling flow (from site [Ovea])

2 Compiling process

In this section, I will talk more about what happens when you compile your LaTeX document. By this I mean when you use commands like pdflatex, biber etc.

When you run pdflatex command compiler goes through the whole document and searches for tags/commands like `\label`, `\ref`, `\cite`, `\include`, `\input`, `\section`, subsection etc. (It also searches for other commands that define formatting, include different packages, create graphs and other things, but this is not so important for compiling). And when it gets to the end of the document (we call this page ship out) it produces several files [Boo]:

- **.log** file - this file will be created after each compilation. It holds information about which packages and file have been included (with information like version of package, date etc.). Log file will also record if there is any error or warning. Besides that, it will also contain some general messages and information about the compiled document. This is the file you should first look if anything goes wrong.
- **.aux** file - This is an auxiliary file which contains all the information for cross-referencing. In this file goes all the information about `\label`, `\ref` and other commands that I mentioned previously. You can look at this as **Symbol table** (SYMTAB) from SIC assembler. If you delete this file then you will need to run pdflatex command twice each time (but more about this later).
- **.toc** file - Table of content file
- **.lof** file - List of figure file
- **.lot** file - List of tables file

Additionally, if you use bibtex or biblatex you could also get some file associated with bibliography like .bbl, .bcf, .blg. They have same purpose as .log, .aux and .toc files.

I will focus on useful and most common file that you will get are .log and .aux files. Other files I will just briefly go through.

2.1 Log Files in LaTeX

Log files are a source of information about what happened during the compilations. I will present some of the information that you can find in log file [Werb].

1. **Compiler information** - This first line in .log file. It information about which compiler you are using, which version, which TeX distribution you are using and timestamp.
   One example from my laptop using pdflatex compiler:

   ```
   This is pdfTeX, Version 3.14159265-2.6-1.40.18 (TeX Live 2017) (preloaded format=pdflatex 2018.7.17) 14 JAN 2019
   12:54 entering extended mode
   ```

   and one other example using xelatex compiler from [Werb]:

   ```
   This is XeTeX, Version 3.1415926-2.3-0.9997.5 (Web2C 2011)
   ```

2. **Warnings** - This will usually occur with cross-referencing. Some of the cases would be:

- undefined `\ref` - This means that you have `\ref` that is pointing to non existing `\label`. The solution for this is that you add missing `\label`.
  You would warning like this:

  ```
  LaTeX Warning: Reference ‘fig:test_123’ on page 4 undefined on input line 121.
  ```

  The compiler, in this case, would output two questions marks in your PDF like this:

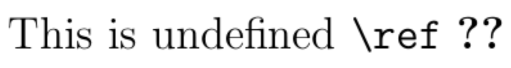
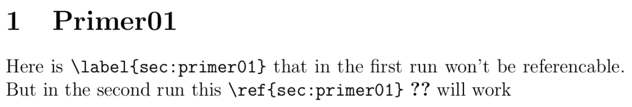
  
  This is undefined \ref ??

  Figure 2: Undefined Ref

  You can try this if you run compile Primer00.tex in the repository. Or you can use a handy script that I wrote and just type:

  ```
  $ ./CompilePrimer.sh Primer00
  ```

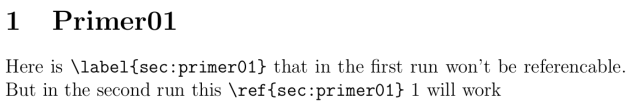
- `\label` not found in .aux file - This happens when we first run our compiler the .aux file is empty and it doesn't have any data. When trying to reference a label it doesn't find any reference in .aux file. The solution for this is to again run pdflatex command and by doing this compiler will find a reference. It will output same warning as with undefined `\ref` . I will show quick example here:
  This is the first run of pdflatex:

  
  1 Primer01

  Here is \label{sec:primer01} that in the first run won't be referencable.
  But in the second run this \ref{sec:primer01} ?? will work

  Figure 3: First Run

  This is the second run:

  
  1 Primer01

  Here is \label{sec:primer01} that in the first run won't be referencable.
  But in the second run this \ref{sec:primer01} 1 will work

  Figure 4: Second Run

  You can also try out with Primer01 that I attached with CompilePrimer.sh script.

- missing reference when citing bibliography - This is due too wrong naming in `\cite` command or in missing reference .bib file or not running bibtex/biber command before pdflatex. The solution is to either correct name in `\cite` command or to add an entry in .bib file or to run commands in this sequence pdflatex -> bibtex/biber -> pdflatex -> pdflatex.
  The warning message will look something like:

  ```
  LaTeX Warning: Citation 'John_Doe' on page 1 undefined on input line 13.

  .
  .
  .

  LaTeX Warning: There were undefined references.


  Package biblatex Warning: Please (re)run Biber on the file:
  (biblatex) LatexCompiling
  (biblatex) and rerun LaTeX afterwards.
  ```

  The output of compiled PDF after first will be:

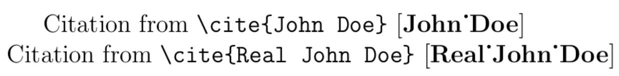
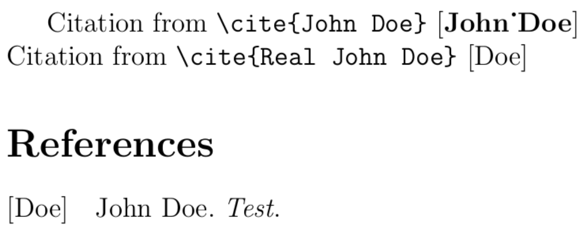
  
  Citation from \cite{John Doe} [John˙Doe]
  Citation from \cite{Real John Doe} [Real˙John˙Doe]

  Figure 5: Citation Warning first run

Here you can see that citations that are missing will be in brackets and the name of citations will be in this brackets (`[citation_ref]`). In the second run (after doing bibtex/biber -> pdflatex -> pdflatex) we got desired output.



Citation from `\cite{John Doe}` [**John˙Doe**]
Citation from `\cite{Real John Doe}` [Doe]

References

[Doe] John Doe. *Test.*

Figure 6: Citation Warning second run

We had the reference in .bib file about Real John Doe. So you can see that compiler found the reference. As for John Doe, it stayed the same because we didn't put in the reference in .bib file.

You can also try out with Primer02.

3. **Statistics** - At the end of each file there will be statistics on document and pdf that can look something like this:

```
Here is how much of TeX's memory you used:
28649 strings out of 493064
455396 string characters out of 6138159
930333 words of memory out of 5000000
31707 multiletter control sequences out of 15000+600000
52065 words of font info for 88 fonts, out of 8000000 for 9000
1137 hyphenation exceptions out of 8191

Output written on LatexCompiling.pdf (5 pages, 112356 bytes).
PDF statistics:
107 PDF objects out of 1000 (max. 8388607)
92 compressed objects within 1 object stream
19 named destinations out of 1000 (max. 500000)
46 words of extra memory for PDF output out of 10000 (max. 10000000)
```

3 Auxiliary Files in LaTeX

As we mentioned before in Auxiliary file (.aux) there is information for cross-referencing. This is probably the most interesting and crucial part of compiling. Latex compiler will open this file before it will start parsing main .tex file. By doing this it will extract all the info for cross-referencing. When parsing document it will try to resolve all the references and at the end of the file, it will write to .aux file all the new references and values of this references. It's similar to the process 2-phase SIC assembly.
So now we will look more deeply at the content of the .aux file.

3.1 Simple example

Let's start with a simple example where we will have only sections and some text. For example:

```
\documentclass[12pt]{article}
\begin{document}
\tableofcontents

\section{Primer01}
\subsection{Primer01 a}
\subsection{Primer01 b}
\subsection{Primer01 c}
\section{Primer02}

\end{document}
```

If we will look into .aux file we would see that there are just commands that instruct the compiler to write indexes of sections and subsection to the .toc file (in our case to Primer03.toc). See below:

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline {1}Primer01}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.1}Primer01 a}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.2}Primer01 b}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.3}Primer01 c}{1}}
\@writefile{toc}{\contentsline {section}{\numberline {2}Primer02}{1}}
```

3.2 Labels and refs

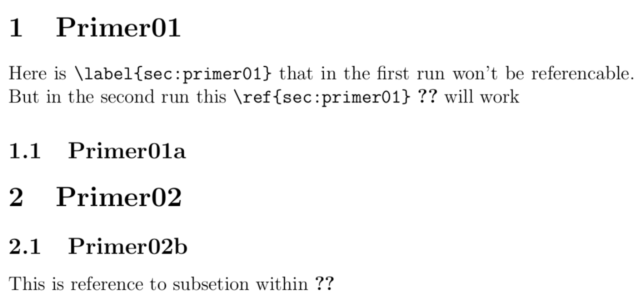
Now we will add some labels to these sections and reference them. We can see command `\label` as creating a variable on which we can reference. And `\ref` command like a function that gets value from that variable.

```
\documentclass[12pt]{article}
\begin{document}

\section{Primer01}
Here is \verb|\label{sec:primer01}| \label{sec:primer01}  that in the first run won't be referencable.\\
But in the second run this \verb|\ref{sec:primer01}|~\ref{sec:primer01} will work

\subsection{Primer01a}

\section{Primer02}

\subsection{Primer02b}
\label{sec:primer02sub}
This is reference to subsetion within~\ref{sec:primer02sub}

\end{document}
```

As we will see in .aux file only command `\label` adds an entry in this file and `\ref` command only reads (gets resolved from .aux file).

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline {1}Primer01}{1}}
\newlabel{sec:primer01}{{1}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {1.1}Primer01a}{1}}
\@writefile{toc}{\contentsline {section}{\numberline {2}Primer02}{1}}
\@writefile{toc}{\contentsline {subsection}{\numberline {2.1}Primer02b}{1}}
\newlabel{sec:primer02sub}{{2.1}{1}}
```

This is also a reason why we need to compile the document twice. Because in the first run we only write down this label in .aux file and we will get two question marks where reference value should be. See the figure 7 below.



1 Primer01

Here is `\label{sec:primer01}` that in the first run won't be referencable.
But in the second run this `\ref{sec:primer01}` **??** will work

1.1 Primer01a

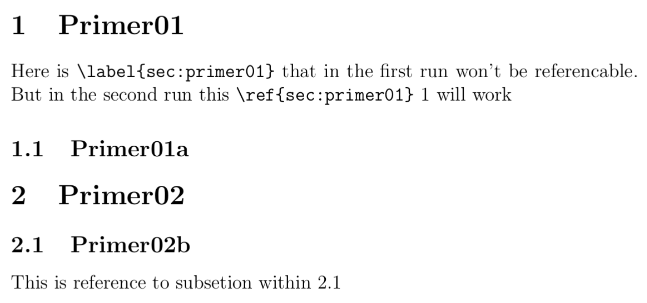
2 Primer02

2.1 Primer02b

This is reference to subsetion within **??**

Figure 7: Label ref first run

Later in the second run, this reference will get resolved and the question marks will disappear, but the .aux file will stay the same. You can see this in the figure 8 below.



1 Primer01

Here is \label{sec:primer01} that in the first run won't be referencable.
But in the second run this \ref{sec:primer01} 1 will work

1.1 Primer01a

2 Primer02

2.1 Primer02b

This is reference to subsetion within 2.1

Figure 8: Label ref second run

You can also try out with Primer04.

3.3 Include and input files

First I will explain the motivation behind using `\include{filename}` and `\input{filename}` command. It gives you an opportunity to break large files into a smaller part that you can edit. One huge benefit is that all the settings, formatting and section indexes will be transferred from base file to this included file [Wik]. This means that you don't need to write twice all the stuff at the top of the file (`\usepackage`, `\graphicspath`, `\newcommand` etc.). This is because when the compiler sees `\include{filename}` or `\input{filename}` command it pauses parsing of the main file and goes to this included file parses it and then continues parsing the main file.

The differences between these two commands are that you cannot nest `\include` command in files and that it will force a page break. This makes ideal for book chapters or large seminars. While with `\input` you can nest command in each other files and it will not force a page break. This is ideal for including pictures and making articles.

We have also a difference when it comes to .aux files. Let's look at the bottom example. This is Primer05.tex

```
\documentclass[12pt]{article}

\usepackage{graphicx}           % images
\graphicspath{ {../../img/} }
\begin{document}

\section{Input file}
\input{Primer05Input}

\include{Primer05Include}

\end{document}
```

This is Primer05Input.tex

```
This is imported text with inputed\\
It more useful for graphics and figures:
\begin{figure}[h]
    \centering
    \includegraphics[width=0.5\textwidth]{InputImage.png}
    \caption{Input image }
    \label{fig:InputImage}
\end{figure}
```

This is Primer05Include.tex

```
\section{Include file}
This is imported text with include\\
It can have:
\begin{itemize}
    \item a
    \item b
    \item c
\end{itemize}

or even subsection
\subsection{Include subsection}
or  section
\section{Include section 2}
```

Now if we look at Primer05.aux

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline {1}Input file}{1}}
\@writefile{lof}{\contentsline {figure}{\numberline {1}{\ignorespaces Input image }}{1}}
\newlabel{fig:InputImage}{{1}{1}}
\@input{Primer05Include.aux}
```

we will see that `\include` command makes reference to Primer05Include.aux file. While `\input` command makes direct entry in .aux file. We can see this with at new label.
Now let look at Primer05Include.aux file

```
\relax
\@writefile{toc}{\contentsline {section}{\numberline {2}Include file}{2}}
\@writefile{toc}{\contentsline {subsection}{\numberline {2.1}Include subsection}{2}}
\@writefile{toc}{\contentsline {section}{\numberline {3}Include section 2}{2}}
\@setckpt{Primer05Include}{
\setcounter{page}{3}
\setcounter{equation}{0}
\setcounter{enumi}{0}
\setcounter{enumii}{0}
\setcounter{enumiii}{0}
\setcounter{enumiv}{0}
\setcounter{footnote}{0}
\setcounter{mpfootnote}{0}
\setcounter{part}{0}
\setcounter{section}{3}
\setcounter{subsection}{0}
\setcounter{subsubsection}{0}
\setcounter{paragraph}{0}
\setcounter{subparagraph}{0}
\setcounter{figure}{1}
\setcounter{table}{0}
}
```

We see here that it's the same format as our previous examples. Just that it sets some parameters so that it has right referencing

4 Infinitely many compiler passes

When I was researching for this topic I stumbled upon interesting StackExchange question. It was how to write LaTeX document that would not stabilize (that means it would infinitely many compiles to compile this project). That also means that each compile would give different resoult which is what we don't want.
Someone actually answered this question with this document:

```
\documentclass{article}

\pagenumbering{Roman}
\begin{document}

a\clearpage b\clearpage c\clearpage

\begin{figure}[!t]
\framebox(200,430){}
\caption{a figure to take up space}
\end{figure}


Some interesting text about  something in Section \ref{x},
which starts on page \pageref{x}.
```

```
\section{zzz\label{x}}
The text of an interesting section.
\end{document}
```

The catch here is that when you count pages in Roman numerals, IV take more space than V. This causes to go to a new page. But when on new page it takes less space so it goes back to the old page and this process is cycling forever.

If you compile Primer06.tex you will see the result of this. You can also check answer on this link [Car]

5 Build Tools

Now we know how compiling works behind the scenes. Now we can use build tools, editors and online editors that will make our life easier. I will list some of them that are the most popular. And I will only demonstrate only one because in their core each build tool does the same as we would. It executes this chain of commands (pdf)latex -> bibtex/biber -> (pdf)latex -> (pdf)latex -> ... and on each step checks if the document is already stable. This means that it checks if there are some references to resolve and some bibliography to include.

Build tools[Blo]

- latexmk - Documentation home page [Lat]
- Rubber - Home page [Rub]
- arara - Github site [ara]

Editors *Disclaimer: these editors are specially made for LaTeX. Almost certainly you could also take your favourite editor and add some plugins and have a functional editor. By favourite editor I have in mind something like (VS Code, Sublime, Eclipse, Emacs, Vim etc.)*

- Texmaker - Home page[Tex] supported on Linux, Mac and Win
- TeXworks - Home page [TeXc] supported on Linux, Mac and Win
- TeXShop - Home page [TeXb] supported on Mac
- TeXnicCenter - Home page [TeXa] supported on Win

Online Editors

- Overleaf - Home page [Oveb]
- LaTeX Online Editor (XO) - Home page [XO]
- LaTeX Base - Home page [Bas]
- Papeeria - Home page [Pap]

5.1 Latexmk

Using Latexmk is as simple as running command:

```
$ latexmk -pdf
```

which will run pdflatex on all .tex file in current directory.
If you want specific file to be compiled you just add the name like this:

```
$ latexmk myfile.tex -pdf
```

and at the end you can clean you directory with this command:

```
$ latexmk -c
```

References

[ara] arara. *arara Github page*. URL: https://github.com/cereda/arara. (accessed: 15.01.2019).

[Bas] LaTeX Base. *LaTeX Base Home page*. URL: https://latexbase.com/. (accessed: 15.01.2019).

[Blo] Tex Blog. *Tex Blog - Tex Resources*. URL: https://texblog.org/tex-resources/. (accessed: 15.01.2019).

[boo] Wiki book. *Wiki book - Latex Home site*. URL: https://en.wikibooks.org/wiki/LaTeX. (accessed: 14.01.2019).

[Boo] Dickimaw Books. *Auxiliary and Log Files*. URL: https://www.dickimaw-books.com/latex/novices/html/auxiliary.html. (accessed: 13.01.2019).

[Car] David Carlisle. *Stackexchange Tex - Document requiring infinitely many compiler passes?* URL: https://tex.stackexchange.com/questions/30674/document-requiring-infinitely-many-compiler-passes. (accessed: 14.01.2019).

[egr] User: egreg. *Stackexchange Tex - Latex distributions*. URL: https://tex.stackexchange.com/questions/239199/latex-distributions-what-are-their-main-differences. (accessed: 13.01.2019).

[Gee] Egon Geerardyn. *Github Repo with Latex resources*. URL: https://github.com/egeerardyn/awesome-LaTeX. (accessed: 15.01.2019).

[Lat] Latexmk. *Latexmk home page*. URL: https://mg.readthedocs.io/latexmk.html. (accessed: 15.01.2019).

[Ovea] Overleaf. *Choosing a LaTeX Compiler*. URL: https://www.overleaf.com/learn/latex/Choosing_a_LaTeX_Compiler. (accessed: 12.01.2019).

[Oveb] Overleaf. *Overleaf Home page*. URL: https://www.overleaf.com/. (accessed: 15.01.2019).

[Pap] Papeeria. *Papeeria Home page*. URL: https://papeeria.com/. (accessed: 15.01.2019).

[Proa] Latex Project. *Latex 3*. URL: https://github.com/latex3/latex3. (accessed: 14.01.2019).

[Prob] Latex Project. *Latex Kernel*. URL: https://github.com/latex3/latex2e. (accessed: 14.01.2019).

[Proc] Latex Project. *Latex Official Documentation*. URL: https://www.latex-project.org/help/documentation/. (accessed: 14.01.2019).

[Prod] Latex Project. *Latex Project Home site*. URL: https://www.latex-project.org/. (accessed: 14.01.2019).

[pro] Latex project. *Getting LaTeX*. URL: https://www.latex-project.org/get/. (accessed: 13.01.2019).

[Rub] Rubber. *Rubber home page*. URL: https://launchpad.net/rubber/. (accessed: 15.01.2019).

[ST] Paul Stanley and Torbjørn T. *Stackexchange Tex - Question mark or bold citation key instead of citation number*. URL: https://tex.stackexchange.com/questions/63852/question-mark-or-bold-citation-key-instead-of-citation-number. (accessed: 14.01.2019).

[Tex] Texmaker. *Texmaker Home page*. URL: http://www.xm1math.net/texmaker/. (accessed: 15.01.2019).

[TeXa] TeXworks. *TeXnicCenter Home page*. URL: http://www.texniccenter.org/. (accessed: 15.01.2019).

[TeXb] TeXworks. *TeXShop Home page*. URL: https://pages.uoregon.edu/koch/texshop/. (accessed: 15.01.2019).

[TeXc] TeXworks. *TeXworks Home page*. URL: http://www.tug.org/texworks/. (accessed: 15.01.2019).

[TUG] TUG. *Tex User Group*. URL: https://www.tug.org/. (accessed: 14.01.2019).

[Wera] Werner. *Stackexchange Tex - Understanding how references and labels work*. URL: https://tex.stackexchange.com/questions/111280/understanding-how-references-and-labels-work. (accessed: 14.01.2019).

[Werb] User: Werner. *Stackexchange Tex - Understanding the Log file*. URL: https://tex.stackexchange.com/questions/32213/understanding-the-log-file. (accessed: 14.01.2019).

[Wik] Wikibooks. *Wikibooks/LaTeX - Modular Documents*. URL: https://en.wikibooks.org/wiki/LaTeX/Modular_Documents. (accessed: 14.01.2019).

[Wria] Joseph Wright. *Stackexchange Tex - How does the LaTeX compile chain work exactly?* URL: https://tex.stackexchange.com/questions/121383/how-does-the-latex-compile-chain-work-exactly. (accessed: 14.01.2019).

[Wrib] Joseph Wright. *Stackexchange Tex - What is the difference between TeX and LaTeX?* URL: https://tex.stackexchange.com/questions/49/what-is-the-difference-between-tex-and-latex. (accessed: 14.01.2019).

[XO] LaTeX Online Editor (XO). *LaTeX Online Editor (XO) Home page*. URL: http://latexonlineeditor.net/. (accessed: 15.01.2019).