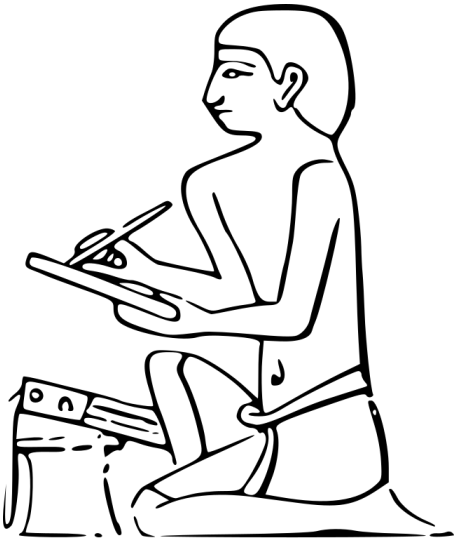
USER GUIDE



ESCRIBA

An efficient and automated LaTeX project layout

Abstract

This is the official ESCRIBA user guide. The tool allows for efficiently manage large LaTeXbased projects and automate several tasks such us cleaning of the working directory, code reformatting, linking of externally generated figures and bibliography compilation.

In this manual, all required information for downloading, installing, creating a new project and customizing it is presented.

The tool is still under heavy maintenance, so if you experience any issue or bug while working with it, please open a new issue in the official board: https://github.com/jorgepiloto/escriba/issues.

Keywords: LaTeX, automated template, project documentation, academical work

Contents

1 The ESCRIBA project

ESCRIBA is just an efficient and automated LaTeXproject layout:

- A project layout because it provides you with several directories for building documents making use of LaTeX.
- It is automated, as it ships with a Makefile for cleaning, formatting, compiling and rendering your work.
- And finally, it is efficient because it saves you time!

When working with large academical works, it might be difficult to manage large quantities of data and information. The only thing writer should care about is writing. This is the main goal of ESCRIBA .

In addition, this tool does not require from Ethernet access, meaning that you can fully work locally and keep track of the changes by using a version control system (VCS) such is Git.

Finally, ESCRIBA is an open-source software, released under Apache 2.0 LICENSE. This means that users are allowed to improve and contribute to the project or even create a new one! The source code is hosted in https://github.com/jorgepiloto/escriba.

2 Installation guide

This chapter is devoted to explain the user how to install the ESCRIBA tool. First all software dependencies are introduced and their own installation guide is provided. Then, the chapter focuses on how to download and install ESCRIBA .

2.1 Project dependencies

Before even downloading ESCRIBA , you require from some additional tools. These tools or dependencies are for code formatting of drawings compilation and rendering. Some of those are optional, but it is recommended that you install all of them.

The required dependencies by ESCRIBA are:

- Make: a tool for automating processes, originally designed for controlling the generation of executables and other non-source files. By making use of a Makefile, different rules are provided for managing all your LaTeXproject.

- LaTeX: of course, without it you will not be able to render any document.

- Ghostscript (optional): used for building drawings created by the Asymptote software.
- Asymptote (optional): a framework which allows the creation of vectorial drawings by making use of custom scripts. It is a very powerful tool for drawing beautiful scientific figures which would be complex of getting if using TkiZ.
- Python: because ESCRIBA is a cookiecutter template, you will need this programming language to create a new project.

From now on, it will be assumed that the operative system you are using is a Linux based one, in particular the Ubuntu flavor. The only thing which differs from this flavor with the rest is the package manager you use for installing the dependencies.

2.1.1 Installing the dependencies

For installing the different dependencies, use your favorite package manager. As said before, Ubuntu is the Linux distribution being used as example. Therefore, follow the command exposed by figure 2.1:

```
sudo apt-get install build-essential ghostscript asymptote texlive-full
```

Fig. 2.1: The command to be used for installing the dependencies.

Regarding Python installation, it is likely that your system already ships with it. Nevertheless, you can download it from the official Python webpage. To do so, use the link provided below these lines:

Download Python: https://www.python.org/downloads/

Once you have downloaded Python and installed it, it is time for installing the cookiecutter package. This package allows for building templates, so you can start a new ESCRIBA project whenever you want. Run the same command from figure 2.2:

```
python -m pip install cookiecutter
```

Fig. 2.2: The command to install cookiecutter package.

3 How to use ESCRIBA

This chapter presents to the user how to use ESCRIBA . At first, the procedure for creating a new project is presented together with the different available rules and commands.

3.1 Creating a new ESCRIBA project

After installing all project dependencies, you are ready to download and use ESCRIBA . Because this tool is simply a LaTeXtemplate it does not require from an installation but from a cookiecutter call. Hence, start a new ESCRIBA project by running the command from figure 3.1:

```
cookiecutter gh:jorgepiloto/escriba
```

Fig. 3.1: The command to create a new ESCRIBA project.

Previous command will ask you to input some parameters such us the name of your project, the title of your work, author, location and date among others. These parameters might become more complex as the project evolves.

3.2 Project structure

Any ESCRIBA project will generate the structure depicted by figure 3.2.

```
escriba_project
├── asy                 # Save here your Asymptote scripts
├── bib                 # All bibliography related files
├── bin                 # Any Python binaries
├── dat                 # A directory holding data files for your scripts
├── fig                 # Generated figures from Asymptote or Python will be placed in here
│   └── static          # Static figures will no be removed when cleaning project directories
├── main.tex            # The main project file
├── Makefile            # A make script for automating different ESCRIBA commands
├── README.md           # Introduce your readers to your project with a README file
├── src                 # All core files: organize those by chapters, then by sections
│   ├── ch00
│   │   ├── index.tex
│   │   ├── one_section.tex
│   │   └── another_section.tex
│   ├── ch01
│   │   ├── index.tex
│   │   └── ...
│   └── ...
└── tex                 # LaTeX customization: packages, commands, cover and preface
    ├── commands.tex    # Define your custom commands in here
    ├── cover.tex       # The cover of your project
    ├── packages.tex    # All require packages for your project
    └── preface.tex     # The preface or abstract goes in here
```

Fig. 3.2: Generic ESCRIBA project layout.

Each one of the directories is devoted to a particular task:

- The main goal of the asy/ directory is to store all the Asymptote scripts for generating the different vector figures of your project. ESCRIBA will automatically detect if any figure is present, compile it and move it temporary to the fig/ folder before compiling your LaTeXdocument.

- The bib/ directory is expected to store all the bibliography files, who's extension is *.bib. You will need to link manually

those in the main.tex file.

- Regarding the bin/ folder, it is devoted to the storage of binaries. These can be Python files or any other ones. However, for the moment, ESCRIBA is only expected to work with Python scrips.

- The dat/ directory can be used to save different data files used by the binaries.

- For the fig/ folder, every PNG file which is not stored in the fig/static/ directory will be assumed to be a temporary file. This is because the output of the Asymptote scripts or the figures developed using the Python binaries are expected to be saved in this directory.

- The main.tex is the critical file of the project. It links all the different *.tex and *.bib files, while declares the LaTeXclass of the document. User is expected to add the different index.tex files of every chapter within the main.tex.

- All the commands and rules are defined in the Makefile. Not only that, you can set up the tools and their configuration, such us the formatting options or the output name of the rendered PDF.

- The README.md is not included but user is encouraged to add one. This is a common file in software projects and its goal is to inform users about whatever the author considers important about the project.

- The src/ directory stores all the information files of the work. The idea is that you create a folder for each one of the chapters. Each one of these folders is expected to have an index.tex file and one TeX per section of the chapter. These

content files can be linked using the input LaTeXcommand in the index.tex. Remember to include the index.tex files in the main.tex one. Make sure to include the new chapter directories within the $(CHDIRS) variable in the Makefile.

- Finally, the tex/ folder stores some LaTeXfiles which are not related to the content of your work but to its style, such us the cover, preface or abstract and custom commands and packages.

3.3 Available commands

As introduced in previous section, the Makefile contains all the rules and automation commands. Among these rules you can find:

- **make clean**: this rule removes all the auxiliary files generated by LaTeX, so your project directory becomes clean.
- **make drawings**: this command will compile all the scripts stored in the asy/ directory but will not remove those from the fig/ one.
- **make binaries**: this command will compile all the binaries hosted in the bin/ folder.
- **make reformat**: by calling this rule, all your TeX files will be reformatted according to the options defined in the $(LATEXINDENTOPTS) variable, which you can modify as you need.
- **make pdf**: the command use to compile and render the whole project into a final PDF file.

4 How to customize

To be completed...