Abstract

The programming side of LaTeX is often overlooked. In many cases, the use of programming capabilities of LaTeX can facilitate document editing and improve the quality of the compiled document. LaTeX3 is a set of modern programming interfaces provided by the LaTeX kernel. Compared to traditional LaTeX programming, LaTeX3 yields more standardized, readable, and robust code. In this tutorial, we present a series of examples that help readers understand the commonly used modules of LaTeX3.

Modern LaTeX programming: an example based LaTeX3 tutorial

Ziyue Xiang

Contents

List of Examples

1 Introduction

There is no doubt that LaTeX is viewed as a typesetting language by most of the users. The programming aspect of LaTeX is often overlooked by many. In practice, many large and structured documents can benefit from the programming capabilities provided by LaTeX. Even understanding the most basic programming principles in LaTeX can greatly facilitate the efficiency of generating figures or tables that are made up of similar and repeated sub-structures. The infrastructure provided by LaTeX 2_ε [2] is already Turing complete, which means its programming capabilities are identical to those of Python [12] and C [9]. However, the syntax and conventions of LaTeX 2_ε are nonstandardized and obsolete compared to the mainstream programming languages now. This makes learning LaTeX 2_ε programming more difficult for today's LaTeX users.

In order to modernize the programming interfaces in LaTeX, the LaTeX3 programming interfaces are introduced [5]. Unfortunately, the learning materials for this tool chain is scarce. One of the few resources available for new learners is The LaTeX3 Interfaces [11], which is an technical documentation that is difficult to for one to start with. Therefore, in this material we intend to provide an example based LaTeX3 tutorial for LaTeX3 learners with sufficient background in computer programming. That is, the reader is expected to understand basic structures (e.g., loops, conditional branches) as well as data types (e.g., integers, floating point numbers, strings) in programs. It would be helpful if the reader also understands the basic principles of the C programming language [9].

Since the LaTeX3 project has accumulated a huge code base over the years, it is infeasible to cover all of its functionalities in one tutorial. In this tutorial, we focus on the most frequently used components in LaTeX3. The complete documentation of LaTeX3 can be found in [11]. In the upcoming section titles, if parentheses are present, then the content in the parentheses is the corresponding section number in [11].

1.1 Motivations of LaTeX3

As mentioned above, LaTeX 2_ε is already Turing complete and serves as the building blocks of many existing packages. In this section, we describe the problems of traditional LaTeX programming, which justifies the reason why LaTeX3 is developed despite already having the powerful LaTeX 2_ε and many other existing packages.

Nonuniform interface Outside LaTeX3, the interfaces provided by traditional LaTeX are not standardized. They suffer from the following disadvantages.

Firstly, the mechanism of LaTeX can affect the readability of traditional LaTeX code. In LaTeX, functions and variables are all declared using control sequences (see Chapter 3 of [4]). When a function is invoked, it is expected to execute a series of predefined procedures. Variables are used to store values only. In LaTeX, we can declare functions that absorb one or multiple arguments. Sometimes arguments are stored in variables. Since both functions and its arguments can be both control sequences, it is difficult to distinguish between them. This is likely to make reading traditional LaTeX source code difficult.

Secondly, in traditional LaTeX the implementation of many fundamental programming capabilities are provided by external packages. For example, to compare the equality of two strings, we can use `\ifthenelse` and `\equal` from `ifthen` package [3]; we can use `\pdfstrcmp` from `pdftexcmds` package [7]; we can also use `\IfStrEq` from `xstring` package [10]. The use of multiple similar packages is likely to cause redundancy and compatibility issues. The lack of a centralized documentation and comparison for these similar packages also increases the learning cost of traditional LaTeX programming.

Expansion control Unlike generic programming languages, LaTeX does not have support for types. Programming components such as variables, functions and function arguments are all treated in the same way. Therefore, LaTeX programmers need to more precisely control how variables are defined and how functions are called. These techniques are known as expansion control. Expansion control in traditional LaTeX is achieved by using the `\expandafter` command, which is difficult to use because the number of `\expandafter` command calls required may scale exponentially [1].

LaTeX3 aims at mitigating these inconveniences in traditional LaTeX programming. It provides a uniform interface for LaTeX programmers, where functions and variables are separated from each other. It defines standardized infrastructure for many programming tasks such as string processing, numerical calculation, regular expression matching, etc. The new expansion control mechanism of LaTeX3 is easier and more straightforward to use.

1.2 Compiling Examples

This tutorial is based on examples. To compile the examples, the minimum preamble required is:

```
1 \documentclass{article}
2 \usepackage{tikz} % load TikZ for some TikZ examples
3 \usepackage{expl3} % load latex3 packages
                                                       1
```

The example code should be placed between `\begin{document}` and `\end{document}`. All examples were tested with TeXLive 2020 on Ubuntu 20.04. For newer versions of LaTeX compilers, there is no need to load the `expl3` package explicitly (i.e., Line 1:3[1] is optional).

The source code of this tutorial can be obtained from `https://github.com/xziyue/latex3-tutorial-latex-source`.

2 LaTeX3 Naming Conventions (I-1)

Unlike many programming languages that enclose the function arguments with parentheses, LaTeX does not require delimiters between the function and its arguments. In the example below, we define 6 control sequences, where `\ta` and `\td` are functions, and the rest are variables.

```
1 \newcommand{\ta}[2]{[arg1={#1}, arg2={#2}]}
2 \newcommand{\tb}{$\alpha$}
3 \newcommand{\tc}{$\beta$}
4 \newcommand{\td}[1]{[arg3={#1}]}
5 \newcommand{\te}{$\gamma$}
6 \newcommand{\tf}{$\delta$}
7 \ta\tb\tc\td\te\tf
```

[arg1=α, arg2=β][arg3=γ]δ

2

[1] Every listing has a unique index, which is shown at the bottom right corner. A line in the listing is referenced by <listing index>:<line number>.

On Line 2:7, we call functions `\ta` and `\td` with their respective arguments (stored in variables). We output the value of `\tf` next. In appearance, Line 2:7 is six control sequences placed next to each other. It is difficult to understand the code unless the programmer finds out which control sequences are functions and how many arguments each function absorbs. To improve readability, LaTeX3 introduces a special naming convention where functions and variables are clearly distinguishable. In addition, programmers can gather more information from function and variable names such as the number of function arguments, the type of variables, and the scope of variables.

2.1 Category Codes and Command Names

In LaTeX, every input character can be classified into 16 categories. Each category is identified with an integer ranging from 0 to 15, which is known as the *category code*. More detail about category codes can be obtained from [8]. We focus on one of the sixteen categories, which is known as *letter*. In most cases, command names in LaTeX can only be made up of characters from the letter category. Because the letter category only contains the lowercase and uppercase versions of the 26 English alphabets by default, command names are comprised of these 52 characters exclusively under the initial LaTeX setup. By extending the letter category, it is possible to add more permissible characters to command names. For example, the `\makeatletter` command changes the category of `@` to letter, which allows it to be used in command names [6]. Many packages use `\makeatletter` and include the `@` character in the defined command names. This technique can prevent LaTeX users from overwriting a command imported from the package accidentally, since one can only access commands whose names are made up of English alphabets under the default settings. To separate LaTeX3 from other LaTeX programming conventions, LaTeX3 introduces two new characters into the command names, namely `_` (underscore) and `:` (semicolon).

References

[1] S. V. Bechtolsheim. A tutorial on `\expandafter`. *TUGboat* 9(1):57–61, 1988.

[2] K. Berry, S. Gilmore, and T. Martinsen.

LaTeX 2ε: An Unofficial Reference Manual. 12th Media Services, 2017.

[3] D. Carlisle, The LaTeX Project Team, and L. Lamport. *The `ifthen` package.* `https://ctan.org/pkg/ifthen?lang=en`

[4] D. E. Knuth and D. Bibby. *The TeXbook.* Addison-Wesley Reading, 1984.

[5] F. Mittelbach and the LaTeX Project Team. Quo vadis LaTeX(3) team — a look back and at the upcoming years. *TUGboat* 41(2):201–207, 2020.

[6] A. Munn. What do `\makeatletter` and `\makeatother` do?, 1 2011. `https://tex.stackexchange.com/questions/8351/what-do-makeatletter-and-makeatother-do`

[7] H. Oberdiek. *The `pdftexcmds` package.* `https://ctan.org/pkg/pdftexcmds?lang=en`

[8] Overleaf. Table of TeX category codes. `https://www.overleaf.com/learn/latex/Table_of_TeX_category_codes`

[9] D. M. Ritchie, B. W. Kernighan, and M. E. Lesk. *The C programming language.* Prentice Hall Englewood Cliffs, 1988.

[10] C. Tellechea. *The `xstring` package.* `https://www.ctan.org/pkg/xstring`

[11] The LaTeX Project Team. The LaTeX3 Interfaces, October 2020. `http://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/interface3.pdf`

[12] G. VanRossum and F. L. Drake. *The Python language reference.* Python Software Foundation Amsterdam, Netherlands, 2010.

◇ Ziyue Xiang
Purdue University
`ziyue.alan.xiang (at) gmail (dot) com`
`https://www.alanshawn.com`
ORCID 0000-0001-6054-5801