Latex Certificate Course Instructions

Jacob Antony

April 1, 2021

LaTeX supports lengths and counters for document configuration. The lengths decides the dimension of different elements of the document including headers, footers, paragraphs, images, tables, ... And counters are used for numbering chapters, sections, paragraphs, equations, tables, figures, definitions, theorems, items on a list, ...

1 Lengths

The following are default length commands in LaTeX,

- `\baselineskip`
- `\baselinestretch`
- `\columnsep`
- `\columnwidth`
- `\evensidemargin`
- `\linewidth`
- `\oddsidemargin`
- `\paperwidth`
- `\paperheight`
- `\parindent`
- `\parskip`
- `\tabcolsep`
- `\textheight`
- `\textwidth`
- `\topmargin`
- `\unitlength`

1.1 Length related Commands

And LaTeX has following commands for managing the lengths,

\newlength to create a new length

\setlength to assign a value to a length

\addtolength to add a value to a length

For example, `\setlength{parskip}{3pt}` changes the paragraph separation length to 3pt. The paragraphs will be separated by a vertical space of 3pt. The default value of the \parskip length is 1pt which is $\frac{1}{72}$ inches.

1.2 Unit of Measure

LaTeX support the following units of measure for lengths.

- in
- cm
- em
- pc
- dd
- sp
- mm
- pt
- ex
- bp
- cc

2 Counters

The following are default counters in LaTeX,

- `part`
- `chapter`
- `section`
- `subsection`
- `subsubsection`
- `paragraph`
- `subparagraph`
- `page`
- `figure`
- `table`
- `footnote`
- `mpfootnote`
- `equation`
- `enumi`
- `enumii`
- `enumiii`
- `enumiv`

2.1 Counter related Commands

And LaTeX uses the following commands for managing the counters,

\newcounter to create a new counter

\setcounter to assign a value to the counter

\stepcounter to increment the counter value

\addtocounter to add a number to the counter value

\value to print the value of the counter

\the`CounterName` to print the value of the counter in a suitable format
For example, `\thechapter`, `\thesection`, `\theequation`, ...

\usecounter to use a counter value for another counter

For example, `\thesubsection` prints 2.1 where 2 is the current section number and 1 is the current subsection number. And `\setcounter{enumiv}{5}` assign value 5 to the `enumeration` item counter at depth 4.

2.2 Printing Counter Value

Other than `\the`*`CounterName`* commands, there are a few commands for printing the value of these counters in different formats.

`\arabic` uses indo-arabic numerals

`\roman` uses roman numerals

```
\begin{enumerate}
\item Depth 1
\begin{enumerate}
\item Depth 2
\begin{enumerate}
\item Depth 3
\begin{enumerate}
\item First
\setcounter{enumiv}{5}
\item Sixth
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}
```

1. Depth 1
   (a) Depth 2
      i. Depth 3
         A. First
         F. Sixth

Figure 1: Manipulating Counters

`\Roman` uses roman numerals in uppercase

`\alph` uses english alphabets

`\Alph` uses english alphabets in uppercase

`\fnsymbol` uses footnote symbols

For example, `\roman{subsection}` gives ii which is the current value of the `subsection` counter in roman numerals.

In Figure 1, you can see in the output that LaTeX uses arabic, alpha, roman, Alph numerals for successively nested enumeration environments. And itemize environments and footnotes at different depth uses different symbols in the same fashion.

3 Creating Commands

LaTeX allows you to create new commands using the `\newcommand` command. The `\newcommand` commands takes two arguments, first argument is the name of the new command and second argument is its definition. The number of arguments and default arguments are available as optional arguments of this command. Hash character, # is used for referencing the arguments of the new command being defined.

For example, `\newcommand{\pde}[2]{\frac{\partial #1}{\partial #2}}` creates a command called `\pde`. Clearly, the new command is suppose to be used in math mode. And `$\pde{f(x)}{y}$` gives $\frac{\partial f(x)}{\partial y}$.

Warning : The mismatch of command modes is quite a grave mistake while defining new commands. All the commands, that are used to define a new

command should be either in text mode or in math mode. In above example, both commands used are available in math mode — \frac and \partial.

```
1  \usepackage{textcomp}
2  \newcommand{\group}[2]%
3  {\langle #1,#2 \rangle}
4  \newcommand{\textgroup}[2]%
5  {\textlangle #1,#2 \textrangle}
6  ...
7  \begin{document}
8  ...
9  We have $\group{G}{\ast}$
10 and \textgroup{H}{+}.
```

We have $\langle G, * \rangle$ and ⟨H,+ ⟩.

Figure 2: Creating new Commands

In Figure 2 at lines 2 and 3, \newcommand{\group}[2]{\langle #1,#2 \rangle} creates the \group command which takes two arguments and prints them in between left and right angle brackets, separted by a comma. Also note that these commands will work only inside a math mode. The $\langle G, * \rangle$ is printed using \group command.

And at lines 4 and 5, \newcommand{\textgroup}[2]{⟨#1,#2 ⟩} will create the \textgroup command which also does the same as former, but outside math modes. And ⟨H,+ ⟩ is printed using \textgroup command.

3.1 textcomp Package

The textcomp Package is used for accessing the commands for left and right angle brackets in the text mode — \textlangle and \textrangle. The documentation on latest version of this package is not available. Clearly, this package requires a command catalogue than a detailed documentation. We don't have a formal catalogue, but detexify can give you commands for different symbols and different charts of symbols are available online.

3.2 detexify Service

The detexify[1] is the frontend of LaTeX symbol classifier webservice. It is written in Ruby language. An android version is available at google playstore. The https://detexify.kirelabs.org/classify.html is the webpage featuring it. The detexity code is available on Github.

In detexify, you can draw symbols by hand and get the commands for matching symbols available in LaTeX. This is quite a useful tool, when you are searching for a specific symbol. However, you should use detexify only as a reference. And learn about the purpose of the command from respective package documentation before use.

[1] detexify ©2009 Daniel Kirsch, MIT License

4 Creating Environments

LaTeX allows you to create new environments using the `\newenvironment` command. This command takes three arguments, first argument is the name of the new environment, second is the actions to be performed before entering into the environment and third is the actions to be performed before leaving the environment.

For example, `\newenvironment{myFigure}{\begin{figure} \centering \begin{tikzpicture}} {\end{tikzpicture} \end{figure}}` will create a new environment `myFigure`. And `\begin{myFigure}` will get you into the new environment and `\end{myFigure}` will leave this new environment.

Before entering into the `myFigure` environment, the actions in the second argument are performed. That is, every time `\begin{myFigure}` occurs, LaTeX enters into `figure` environment, applies `\centering` to the image, and then enters into `tikzpicture` environment. And before leaving the environment it performs the actions in the third argument of the `\newenvironment` command.

Warning : The environment definitions should be crafted very carefully. For example, the nesting of environments follow LIFO *(Last In First Out)* order. That is, the last nested environment entered, is the first environment to leave. In above example, any attempt to leave `figure` environment before leaving `tikzpicture` environment will cause an environment delimiter mistmatch error. And error debugging is difficult as the LaTeX error report won't give much hint about the `\newenvironment` command causing the trouble.