LaTeX Primer

Randall R. Holmes

January 4, 2019

Note: For this to make sense it needs to be read with the code and the compiled output side by side. And in order for the compiling to be successful, the file "5320Quiz.sty" *needs to be in the same folder as this document.*

1 Command

A *command* (or *macro*) in LaTeX is a backslash followed by some letters (the name of the command). A command tells the compiler to do something. For instance, the command with the name "LaTeX" tells the compiler to produce the logo LaTeX. The command with the name "copyright" produces ©. The command with the name "quad" produces a large space, like between these words.

For some commands an input is required before the compiler knows what to do. For instance, the "emph" command needs to know what it is you want to emphasize, so you provide this as input by putting it between braces after the command: *This text is emphasized.* Similarly, **this text is in bold face**.

LaTeX comes with a long list of built-in commands, but you can also define your own using the "newcommand" command. For example, if you need to write the alphabet several times, you can save keystrokes by first defining a new command "AtoZ" as follows: (See the code.)

Then you can produce the alphabet like this: ABCDEFGHIJKLMNOPQRSTUVWXYZ. So the "newcommand" command makes the first thing in braces a substitute (a nickname) for the second thing in braces.

You can also define your own commands that use one or more inputs. You will not need to define commands in this course so I will say no more about this, but if you are interested you can read about it online at

https://en.wikibooks.org/wiki/LaTeX/Macros .

2 Math mode

Here is an example of typesetting using math mode: $z = x + y$. Math mode begins with the open parenthesis command, that is, a backslash followed by (. This tells the compiler to switch the way it is doing things. For instance, it says to start typesetting letters using italics, which is the standard practice for typesetting variables. It also tells the compiler to start using spacing that is appropriate for mathematics. Typing the close parenthesis command, that is, a backslash followed by), switches from math mode back to normal typesetting.

What you will be typesetting in this course will be proofs, which involve mostly words with an occasional use of symbols, like this:

Let x be a real number such that $x + 3 > 2$. Adding -3 to both sides of this inequality, we get $x > -1$.

Note how the symbolic parts are all set in math mode, even the number -3. (Without math mode, we get -3, which doesn't look quite right since the negative sign is too short.) As a rule of thumb, you should switch to math mode when you come to something that is not a word and stay in math mode until you encounter the next word (or punctuation).

The old method for notating math mode used dollar signs, like this $z = x + y$ (see the code). This still works, but the old method is being phased out since it causes parsing problems to have the same symbol for both starting and ending math mode. Old habits die hard, so you'll see the dollar signs a lot, but it's best not to get into the habit of using them.

A variation on math mode is *displayed* math mode, which you get by using square brackets instead of parentheses, like this:

$$z = x + y.$$

This is just the same as regular math mode except that the expression is centered and a little space is put above and below.

Since the compiler knows what is standard in the way of spacing in mathematics, it ignores any spaces it sees in math mode, which we can see here: $z = x + y$. Users of LaTeX take advantage of this feature when writing displayed math by coding it like this (see the code):

$$z = x + y.$$

This makes the code easier to read since it resembles the actual output.

Although the compiler is usually right about spacing choices, there are some situations where you might want to override the default spacing. For this, use $x\,x$ (small space), $x\ x$ (normal space), $x\quad x$ (large space), and $x\qquad x$ (even larger space).

The usual mathematics symbols are available for use within math mode. For instance,

$$\alpha, \quad \pi, \quad \sum, \quad \int, \quad \infty.$$

If you try to use these commands outside of math mode the compiler will complain.

I will be providing you with the code for the quiz problems, so you will be able to look at it to see the commands for the various symbols required in your solutions, but if you need to consult a list there is an extensive one in the following document:

http://tug.ctan.org/info/symbols/comprehensive/symbols-a4.pdf .

And at the Detexify site you can draw the symbol you want and it will tell you the command:

http://detexify.kirelabs.org/classify.html .

3 Common constructions

Here are some examples of superscripts:

$$x^2, \quad 3^{x+y}, \quad 3^x + y, \quad 3^\alpha.$$

In LaTeX, braces are used for grouping things, like in the second example. If braces are omitted, then the superscript is assumed to be just the first character following the caret (third example). A command counts as a single character in this regard (last example).

Subscripts are similar:

$$x_2, \quad x_{(1,2)}, \quad x_a bc, \quad x_i^2, \quad {x_i}^2, \quad x_{i^2}.$$

Here is a set given in list form:

$$\{1, 2, 3, \ldots, 100\}.$$

In LaTeX, we use braces to tell the compiler how we want things grouped. The braces themselves are not typeset (see?). So to get braces to appear, like in the set notation above, we need to use the corresponding command, that is, a backslash followed by a brace.

Use ... as above for items separated by commas, but use $\cdots$ for items separated by operations, as in $1 + 2 + 3 + \cdots + 100$. Get a single dot like this: $2 \cdot 3 = 6$.

Here is a set given in set-builder notation:

$$\{n \in \mathbf{Z} \mid n = 2m \text{ for some } m \in \mathbf{Z}\}.$$

The command "text" typesets its input in the normal way. This allows us to momentarily leave math mode for the words "for some." (Without this command, the compiler would think we wanted these letters as variables and would give us $forsome$.) Several spaces were included above for the sake of readability, but only three are essential: the one after "mid" (since the command "midn" is not defined), and the spaces before and after "for some." (Recall that in math mode the compiler controls the spacing, ignoring any spaces it sees.)

Here are some examples of fractions:

$$\frac{ab^2}{c+2d}, \quad \frac{a}{b}, \quad \frac{1}{\beta}, \quad \frac{x+\frac{y}{z}}{w}.$$

The command "frac" requires two inputs: the first becomes the numerator and the second becomes the denominator. If braces are omitted, a single character (or command) is used as the input.

Roots are done like this:

$$\sqrt{a+b}, \quad \sqrt{2}, \quad \sqrt[3]{xy}.$$

The command "sqrt" has one required input, namely, the thing that goes under the radical sign. But it also has an *optional* input, which can be used to specify higher roots (last example). In LaTeX, required inputs are enclosed in braces, but optional inputs are enclosed in square brackets. (It's a bit grating that the last example is not actually a square root, so the name of the command seems wrong, but that was somebody's decision.)

The "not" command works pretty well to negate the symbol that follows it:

$$\neq, \quad \not< , \quad \notin \ .$$

Purists think that this one-size-fits-all approach produces less than satisfactory results in some situations, so they prefer to use the commands that custom fit the strikeout to the symbol:

$$\neq, \quad \not< , \quad \notin \ .$$

It is sometimes difficult to tell the difference. If you want one of the custom commands it should come up as an option when you draw the symbol at the Detexify site.

4 Environment

This is an example of an environment:

This text is centered.

The name of the environment is "center." The first line (see the code) tells the compiler to switch the way it is doing things, namely, start centering the text instead of making it flush with the left margin. The final line ends the environment and switches things back to normal.

Math mode is an example of an environment (the "math" environment). In fact, the notation $z = x + y$ is a shorthand notation for $z = x + y$. And using the square bracket commands for displayed math is shorthand for

$$z = x + y.$$

Sometimes, displayed math is so long that it runs into the margin (or even off the page):

********** = ********** = ********** = ********** = ********** = ********** = ********** =

The best place to break a line of mathematics usually depends on the circumstances, so the compiler doesn't even try to break lines on its own in math mode. It does, however, issue a

warning that we have an "overfull hbox," meaning a horizontal box of stuff that is too wide to fit between the margins.

To fix this we can use the "align*" environment:

$$\begin{aligned} &********** = ********** = ********** = ********** \\ &\quad = ********** = ********** = ********** \\ &\quad = ********** . \end{aligned}$$

The ampersands mark the point of alignment. The double backslash starts a new line. Built into this environment is a switch to math mode. You will get an error if you try to start another math mode inside of it. The spaces and carriage returns used here are just for readability of the code. They are ignored by the compiler. (FYI: There is also an environment called simply "align" (without the *). It automatically numbers each line in the expression, which is usually not what we want for quiz solutions. Putting the * has the effect of suppressing the line numbering.)

Here is another example showing how the "align" environment works:

$$\begin{aligned} &123456789 \\ 123456789 & \\ &\qquad 123456789 \end{aligned}$$

If you have a series of steps and you want to justify them, use the "alignat*" environment:

$$\begin{aligned} c^2 &= a^2 + b^2 && \text{(Pythagorean theorem)} \\ &= 2a^2 && \text{(since } b = a\text{)}. \end{aligned}$$

The alignments occur at the ampersand signs. The "qquad" command in the first line (in combination with the ampersands) fixes the spacing for all subsequent lines, so it does not need to be repeated. Just as with "align*," a switch to math mode is built in. Note the math mode $b = a$ embedded in the text mode (which itself is embedded in math mode). The 2 in the first line, a required input, is a bit confusing. It needs to be half of one more than the number of ampersands in a line: $(3 + 1)/2 = 2$. There is no need to worry about such things right now since you can simply copy and paste the code and amend it to suit your needs.

The "cases" environment is useful for piecewise-defined functions:

$$f(x) = \begin{cases} 2x, & \text{if } x < 0, \\ x + 1, & \text{if } x \geq 0. \end{cases}$$

Again, simply copy, paste, and amend.

For problems that have multiple parts, use the "alphenum" environment:

(a) The first part.

(b) The second part.

(c) And so on.

This environment is not standard LaTeX. It is an environment I defined. So, yes, you can define your own environments just like with commands (and they can include inputs as well). It is unlikely that you will need to do anything this fancy for your quizzes, so I won't show how to do it here. If you're interested, you can easily find the method online.

Incidentally, LaTeX has a built-in environment called "enumerate," which produces a list with the automatically generated item numbers 1, 2, and so on. I prefer lettered items in quiz problems, so I altered this environment to get "alphenum."

5 Package

I mentioned in the last section that the "alphenum" environment is my own environment. But if you look through this document you won't see anything that looks like a definition of this environment. So how does the compiler know what to do when it encounters it?

The key is the second line from the top, that is, the "usepackage" command with input "5320Quiz." This tells the compiler to look for definitions of commands and environments in a file called 5320Quiz.sty. A file, such as this, with the extension ".sty", is called a *package* (or a *style file*). By default, the compiler looks only in the same folder for user-designed packages like this.

If you look in the file "5320Quiz.sty" you will see where I defined the environment "alphenum." You will also see definitions of some commands. For instance, you will see a newcommand statement that shortens the coding of the boldface R for the set of real numbers from $\mathbf{R}$ to just $\mathbf{R}$ (see the code).

Also in that file you will see at the top where I have further directed the compiler to look for definitions in some other packages, like "amsmath." There are thousands of packages freely available. No matter how you are using LaTeX (online, personal computer, computer lab) you will have access to the more popular packages. However, the compiler will look for definitions only in the packages that you tell it to look in.

So what do you do if you find a particular command online that you wish to use? First, just try using it since chances are it is defined in one of the packages like "amsmath" that I have already included for you. If it doesn't work (so the compiler issues the error "Undefined control sequence"), try to find out what package the command is defined in (Detexify gives this information) and then include that package at the top of your document with a "usepackage" statement. In the unlikely situation that the compiler complains that it cannot find the package, search for the package online (it will be a file with extension ".sty") and put a copy of it in the same folder that your document is in.

6 Errors

The compiler at the ShareLaTeX site is quite forgiving. If you make a coding error, the compiler will skip over it and keep on going, but the output will usually look wrong.

Look for an icon just to the right of the Recompile button. When errors occur the icon displays a red flag with the number of lines having errors. Click on the icon to read the error messages.

Here's a command that is undefined: . Note how it is just ignored in the output. See if you can find the error message.

Each message begins with a line number where the error occurs. Then it reproduces your text close to the error and breaks the line right at the point where the error occurs. Finally, it tries to explain the error. If you click on the error message a cursor will appear at the beginning of the line in the code where the error occurred (marked with a red X).

Please try to fix as many errors as you can so that I won't have to fix them for you. Sometimes the error messages can be quite cryptic, especially if you're new to LaTeX, so I'll understand if you can't fix everything.

7 Miscellaneous

When the compiler encounters a percentage sign, it ignores the sign as well as everything after it, all the way to the end of the line. For example, ... (see the code).

This provides a nice way to include comments in the code without messing up the output. Also, if you have second thoughts about some code you typed and you're thinking about deleting it, you can initially just use percentage signs to "comment it out" until you're sure you won't need it.

The easiest way to start a new paragraph is just to leave a blank line, as the code shows. An alternative is to use the "par" command, like this (see the code).

The default is for paragraphs to be indented, but I think that for mathematics it works better to have block paragraphs. You can see in the package 5320Quiz.sty where I have overridden the default behavior.

If you would like a large vertical space in your document, you can get it by using the "bigskip" command (see the code).

(This command requires a new paragraph. Otherwise it is ignored.) There are also the commands "medskip" and "smallskip."

Searching on the Internet is a good way to find out how to do things in LaTeX. The wikibook is also useful (especially the section on mathematics):

https://en.wikibooks.org/wiki/LaTeX .