COMS 6500 Notes

Ashlin Harris

30 August 2016

1 General Remarks

- MT vs. Alabama A&M this Saturday (3 September)
- If you experienced problems connecting to `https://www.cs.mtsu.edu/` on Monday, this may have been due to an outdated certificate. The CS Department has updated the certificate, so this issue should be resolved.

2 Course Reading: *Code Complete*

2.1 Quiz

1. What is code construction?
    - metaphor that expresses an abstract, unfamiliar concept (coding) in terms of a concrete, familiar one (construction)
    - puts emphasis on coding and debugging, less emphasis on planning and unit testing, and very little emphasis on architecture and maintenance
    - aguably, a better metaphor for coding than letter writing, farming, etc.
2. What software metaphor did you like the best?
    - The class preferred the construction metaphor.
    - Letter writing may be a poor metaphor, since it is not uncommon to throw away manuscripts and start again. This is not (purposefully) common with software coding.

Important work ought to be backed up off-site. This way, a disaster that would ruin the working copy and on-site backups will not destroy the project. If this advice ever helps you, credit goes to Dr. Hyrum Carroll.

3 LaTeX

3.1 General Remarks

To compile LaTeX files, `pdflatex` is recommended. This may have to be repeated, as some LaTeX operations require an additional pass. For important documents, go ahead and compile a third time. Output can also be generated using `latex example.tex; dvips example.dvi`, but unless your department or publisher requires a `.dvi` or `.ps` file, it is best not to bother.

The tilde (~) is a special character in LaTeX. It designates a space that should not be enlarged or broken between lines. It can be reproduced in LaTeX using `\textasciitilde` (~) or even `$\sim$` (∼). `\~` is typically not intended. Good examples of usage include `$\sim$hcarroll` for ∼hcarroll and `Dr.~Carroll` for Dr. Carroll.

All LaTeX examples and other files shown in class are publicly available on the course website, `https://www.cs.mtsu.edu/~hcarroll/6100/calendar.html`. For files on Dr. Carroll's website, you are encouraged to modify the ends of the URLs and search the parent directories for useful files.

> Dr. Carroll used a similar technique when considering the purchase of a salvaged car. The merchant only gave him web links to a select few pictures of the car before it was refurbished, but by altering the number in the URL, he was able to find about a dozen more.

If a link is broken, feel free to snoop around in search of the correct file.

3.2 Review

The preamble (section between `\documentclass{...}` and `\begin{document}` may contain `\title{...}`, `\author{...}`, `\date{...}`, etc. Depending on the document class, these are displayed when `\maketitle` is invoked. Package calls are also found here.

3.3 Simple Structures

With LaTeX, lists can be constructed easily, and `\item`s can be inserted effortlessly. Two types of lists (`itemize` and `enumerate`) are shown below. The following example is available at `https://www.cs.mtsu.edu/~hcarroll/6100/lectures/latex/simpleStructures.tex`.

.tex input

```
\documentclass{article}
\begin{document}

  Itemized list:
  \begin{itemize}
    \item Item one
    \item Item two
  \end{itemize}

  Enumerated list:
  \begin{enumerate}
    \item Item one
    \item Item two
  \end{enumerate}

\end{document}
```

.pdf output

Itemized list:

- Item one
- Item two

Enumerated list:

1. Item one
2. Item two

Since numbers are not explicitly specified, a list of type `enumerate` will be renumbered during compilation after an `\item` is inserted or deleted. No manual renumbering is ever required.

3.4 Centering

Text is left delimited by default in LaTeX. Text can be centered as in the following snippet:

.tex input

```
Text is left delimited \\
by default in \\
\LaTeX.
\begin{center}
Text can be centered \\
as in the following\\
snippet:
\end{center}
```

.pdf output

Text is left delimited
by default in
LaTeX.

Text can be centered
as in the following
snippet:

3.5 Verbatim

LaTeXwill interpret special symbols and macros unless you tell it not to. This is a common source of confusion for those who are new to LaTeX. If you get a strange error message upon compilation, check for the following characters:

```
# $ % ^ & _ { } ~ \
```

With one exception, these characters can be escaped by a backslash (e.g., `\#` for #). Since the sequence `\\` is reserved for line breaks, use `\textbackslash` in-

stead. With some macros, whitespace that immediately follows is not displayed; if space is desired, add an empty set of curly braces.

Consider the following examples:

.tex input

```
Here is the macro for \LaTeX: \\
\verb+\LaTeX+ \\
\verb|2+2| \\
For larger passages,
try this format:
\begin{verbatim}
$4.00, {}, my_file.txt, \\
~hcarroll, this & that \\

end{verbatim}
```

.pdf output

Here is the macro for LaTeX:
`\LaTeX`
`2+2`
For larger passages, try this format:

```
$4.00, {}, my_file.txt, \\
~hcarroll, this & that \\
```

For individual lines, it is simple to use `\verb`. Some delimiter character must be provided. Of course, that character must not be present inside the line. For longer passages, it is appropriate to use `\begin{verbatim}` and `\end{verbatim}`.

3.6 Equations

Arguably, one of the best features of LaTeX is its treatment of mathematical notations. Equations can be expressed easily and professionally. To embed mathematical characters in the text, use `$ ... $`. For equations that stand outside the paragraph, use `\begin{equation} ... end{equation}`. Equations of this second type are numbered sequentially and can be referred to using labels. This way, there is never a need to renumber equations and redo references. Take, for instance, the following:

.tex input

```
Energy-momentum relation:
\begin{equation}
E^2=(pc)^2+(m_0c^2)^2
\label{e-m r}
\end{equation}
where $p$ is momentum.
Of course,it follows
from \eqref{e-m r} that
\begin{equation}
E=m_0c^2\text{ when }p=0
\end{equation}
```

.pdf output

Energy-momentum relation:

$$E^2 = (pc)^2 + (m_0c^2)^2 \qquad (1)$$

where p is the system's momentum. Of course, it follows from (1) that

$$E = m_0c^2 \text{ when } p = 0 \qquad (2)$$

3.7 Images

Images can be included using `\includegraphics[...]{file}`. In the square brackets, options such as `height` can be assigned. In general, it is better to make the size of figures proportional to `\textheight` rather than a fixed value. With `\begin{figure}` ... `\end{figure}` notation, a `\caption{...}` can be added, as well as a `\label{...}`. As with equations, figures can be referenced by their labels, avoiding tedious renumbering and potential misreferences.

3.8 Tables

Tables in LaTeX are invoked using the following format:

`\begin{tabular}{`[vertical lines and alignment]`}` ... `\end{tabular}`

A vertical bar (`|`) designates a vertical line in the table and the justification of text in each cell is to the left (`l`), the right (`r`), or the center (`c`). The ampersand (`&`) moves to the next cell on the right, and a double backslash (`\\`) moves to the next row below. Horizontal lines are made using `\hline`. The following table shows these in action:

.tex input

```
\begin{tabular}{|l|cc|r|}
\hline
$n$ & $n^2$ & $n^3$ & $n^2+n^3$ \\
\hline
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 2 \\
2 & 4 & 8 & 12 \\
3 & 9 & 27 & 36 \\
\vdots & \vdots & \vdots & \vdots \\
10 & 100 & 1000 & 1100 \\
\vdots & \vdots & \vdots & \vdots \\
100 & 10000 & 1000000 & 1010000 \\
\hline
\end{tabular}
```

.pdf output

n	n^2	n^3	n^2+n^3
0	0	0	0
1	1	1	2
2	4	8	12
3	9	27	36
$\vdots$	$\vdots$	$\vdots$	$\vdots$
10	100	1000	1100
$\vdots$	$\vdots$	$\vdots$	$\vdots$
100	10000	1000000	1010000

4 Linux Commands

4.1 Linux Commands I Review

`man` show the *man*ual for a command

- `man man` - read manual page for `man`
- `man ls` - read manual page for `ls`
  - *Enter* - skip down 1 line in manual page

  - *Space* - skip down 1 screen in manual page
  - Arrow keys may also be used to navigate the page.
  - `q` - exit manual page

`ls` *li*st directory contents

- `ls` - list the contents of the current directory
- `ls -a /bin` - list *a*ll the contents of `/bin`
- `ls -lh` - display contents in *l*ong-listing format with *h*uman-readable file sizes

`cd` *c*hange *d*irectory

- `cd`, `cd ~` - go to home directory
- `cd my_dir` - go to `my_dir` if it exists in current directory
- `cd ..` - go to parent directory, if it exists

`touch` update a timestamp to current machine time

- If the file doesn't exist, a blank file will be made.

`cp` *cop*y a file

- `cp ./a/b/this that` - copy and rename `this` into the current directory
- `cp -R ./important_files ./test_directory` - recursively copy contents of `important_files` into `test_directory`

`mv` *mo*ve a file

- `mv good/idea.txt bad` - move the file `ideas.txt` to directory `bad`, if it exists; otherwise, move the files to the current directory and rename it `bad`
- `mv today.txt yesterday.txt` - rename the file `today.txt` to `yesterday.txt`

`rm` *rem*ove a file

- `rm old.txt` - remove the file `old.txt`
- `rm -Rf temp_dir/` - remove `temp_dir`, *R*ecursively deleting its contents and *f*orcefully deleting without confirmation
- `rm *` - delete all the files in a directory, but only if the user has permission to do so

`mkdir` *mak*e a *dir*ectory

- `mkdir Programs` - make a new directory `Programs`
- `mkdir Programs/1 Programs/2 Programs/3 Programs/Final` - make several new directories

`rmdir` *rem*ove an empty *dir*ectory

- `rmdir ./COMS/testing` - remove `testing` if it is an empty directory
- `rmdir *` - remove all empty directories in the current directory

4.2 Linux Commands II Exercise and Review

1. Display the name of the current directory.
   - `pwd`
   - We will learn how to put this information in the prompt.
2. Display the path of `mkdir`.
   - `which mkdir`
     - This command looks through the variable `$PATH` and returns the first directory that contains `mkdir`.
   - Tab completion can show if the command is present on your machine.
3. Execute the last command that started with "mak".
   - `!mak`
4. Execute the third to last command used.
   - `!-3`
   - Press the *Up Arrow* three times, then *Enter*.
5. Print to the screen the contents of all of the .txt files in the current directory.
   - `cat *.txt` - con*cat*enates the contents of all files in the directory that match the regular expression, sending it all to standard output
6. Display the filename of just the last file from the previous command.
   - !$
7. View the contents of the file `researchIdeas.txt` one screen full at a time.
   - `more researchIdeas.txt`
   - `less researchIdeas.txt`
   - `less` is like `more`, but there are some differences. For instance, `less` supports backwards scolling, while `more` does not. As a general rule, `less` is more.

Students should be familiar with all the commands covered in class and be able to use them without resorting to a reference.

4.3 Linux Commands III

`ssh` *s*ecure *sh*ell

- `ssh abc1d@herschel.cs.mtsu.edu` - connect to a remote MTSU server using account `abc1d`

`scp` *s*ecure *cop*y

- `scp abc1d@herschel.cs.mtsu.edu:some/remote/directory/file some/local/directory` - copy `file` to the local machine

`sftp` *s*ecure-shell *t*ransfer *f*ile *p*rotocol

- `sftp abc1d@herschel.cs.mtsu.edu` - open sftp connection to remote account with interactive prompt
  - `?`, `help` - see list of commands for the sftp prompt
  - `pwd`, `cd`, `ls` - same as before, but apply to remote machine
  - `lpwd`, `lcd`, `lls` - *l*ocal versions of the above commands
  - `get remote_file.txt` - copy `remote_file.txt` to the local machine
  - `put local_file.txt` - copy `local_file.txt` to the remote machine
  - `exit` - exit the remote connection

`du` *d*isk *u*sage

- `du` - show disk usage for all directories in the current directory
- `du | sort -n` - same as above, but *sort*ed *n*umerically by size
- `du -h` - sams as `du`, but with *h*uman-readable file sizes

`sort` *sort*s a list alphabetically by line

- `sort names.txt` - sort the lines in the file `names.txt`
- `sort numbers.txt` - sort the lines in `numbers.txt` alphabetically (e.g., 80 comes before 9)
- `sort -n numbers.txt` - same as above, but sorted by number

`uniq`

- `uniq` does not search the entire file for matching lines. It only detects matches in subsequent lines. For this reason, unsorted input is almost invariably piped from `sort` first. Alternatively, use `sort -u` instead of `uniq`.
- `uniq -c file.txt` - *c*ount the instances of each line and append the number to the front of the line
- `uniq -d file.txt` - print only *d*uplicate lines
- `uniq -i file.txt` - ignore case (e.g., `word`, `Word`, `wORD`, and `WOrD` are considered duplicates)

`echo` display input to standard output

- `echo Hello` - print `Hello` to the screen
- `echo $PATH` - print the variable `$PATH` to the screen
- `echo *.txt` - print the names of all `.txt` files in the current directory