latex input: mmd-article-header Title: Bash Notes Author: Ethan C. Petuchowski Base Header Level: 1 latex mode: memoir Keywords: Bash, Unix, Linux, Shell, Command Line, Terminal, Syntax CSS: http://fletcherpenney.net/css/document.css xhtml header: copyright: 2016 Ethan C. Petuchowski latex input: mmd-natbib-plain latex input: mmd-article-begin-doc latex footer: mmd-memoir-footer

What is *not* in this set of notes

- Notes on Bash's **syntax** can be found in **Bash Syntax Notes.md**
- Notes on **Sed and Awk** can be found in **Sed and Awk Notes.md**
- Notes on some standard UNIX commands that I have found to be particularly useful can be found in **Bash Commands Notes.md**

tmux

Commands before you've entered a session

- **tmux** -- create an unnamed session
- **tmux new -s sessionName** -- create a named session
- **tmux ls** -- list detached sessions
    - *Or* **tmux list-sessions**
- **tmux attach -t sessionName** -- re-attach to session named `session_name`

These are prefixed with ctrl-b

- **c** -- create a new window
- **,** -- rename window
- **p** -- previous window
- **n** -- next window
- **w** -- list open windows and select which one to enter
- **%** -- split vertically (creates new pane *within* this window)
- **: split-window** -- split horizontally
- **d** -- *detatch* from current session (leaves it running)
    - Now, if you log out of your `ssh` session, that detached session will *still* be left running

Other notes

- When you close the last `tmux` window, `tmux` will **exit**

Handy full commands

Delete all non-pdf files recursively from directory

```
find . ! -name '*.pdf' -delete
```

File Descriptors

- **0** -- STDIN
- **1** -- STDOUT
- **2** -- STDERR

Examples

```
make check 2>&1 | tee make-check.log
```

- We can think of **2>&1** as "point STDERR to where STDOUT points"
- `| tee afile` means "and also print it to the log-file" (in *addition* to STDOUT)

Settable Options

```
set -o <OPTION> [-o <OPTION> ...]
```

- **errexit** -- if a command doesn't return 0, `exit 1` from this script
  - This can also be accomplished with `set -e`
  - This can be disabled with `set +e`
  - If it's normal for a command to fail, append `|| true` after it so that the line will always return 0
- **nounset** -- treat use of unset variables as a fatal error
  - helps avoid mistakes with e.g. `rm -rf $chroot/usr/share/doc`
  - This can also be accomplished with `set -u`
  - This can be disabled with `set +u`
  - If you use this, and you want to use a variable that may be unset, use the synax `${MAYBE_EMPTY:-}`, which means "if the variable is unset, use the default string *emptystring*"
- **pipefail** -- a pipeline normally reports the exit code of its last command. With this option, the exit code of the pipeline is the exit code of the last program to return a non-zero exit code
- **set -f** -- disable filename globbing
- **shopt -s failglob** -- glob characters that get passed to a command cause fatal errors

POSIX Regex Character Classes

- These can help you be international/multicultural
- They are only recognized *inside* a *square-bracket sub-expression*. So if you are using one of these standalone, you must wrap it in square-brackets

```
[[:alnum:]]
```

- They only count for a single character, so for e.g. multiple spaces, you'd have to do

```
[[:space:]]+
```

- What follows is likely not a complete list for *your* system because GNU added more, etc.

But here are the basics

Class	Matches
`[:alnum:]`	alphanumeric
`[:alpha:]`	alphabetic
`[:blank:]`	space & tab
`[:cntrl:]`	control chars
`[:digit:]`	numeric
`[:graph:]`	non-space
`[:lower:]`	lowercase
`[:print:]`	printable
`[:punct:]`	punctuation
`[:space:]`	whitespace
`[:upper:]`	uppercase
`[:xdigit:]`	hexadecimal

Things one may want to do

Repeat a command *N* times

Version with cleanest syntax

```
for i in {1..10}; do command; done
```

Version is more flexible because you can go by 3's etc. (see `seq` command below)

```
for i in `seq 10`; do command; done
```

Iterate through files

```
find . -name '*.csv' | while read line; do
    echo "$line"
done
```

Some crazed bash commands

Coursera Lecture Time Aggregator

In a folder containing a bunch of videos with titles of the format "*Compilers 3.0 04-01 Lexical Specification (14m30s).mp4*", I'd like to count the total number of hours of video.

Algo

1. Morph list of filenames into just the time-pieces, e.g. 6m29
2. Split those strings on `m`
3. Add up the hours and minutes separately
4. Add up the total number of minutes and divide it into hours
5. Print it out

```
$ ls \
  | sed 's/[^(]*(\([^s]*\)s).*/\1/' \
  | awk -Fm '{s+=$1;t+=$2} END {printf "%.2f hrs\n", (t/60+s)/60}'

#=> 19.50 hrs
```

Bootleg Spellchecker

August 11, 2015

Look for spelling errors in some crappy Mac-provided dictionary file. Based on the UNIX command given in the AT&T Archives Video, but many of the commands have

disappeared since then and new ones have been introduced.

```
$ echo "The quick brown Fox jumped over." \
  | tr ' ' '\n' \                    # replace spaces with newlines
  | tr '[:upper:]' '[:lower:]' \     # convert to lowercase
  | sed 's|[^[:alpha:]\n]||g' \      # only keep letters
  | sort -u \                        # sort and dedup
  | comm -23 - /usr/share/dict/words  # keep only words not found in

=> jumped
```