

Linux and Git Boot Camp

Roshan, Zack, Blair, Ian

Jan. 21, 2018

Connecting Clients

SSH

Windows users: MobaXterm, PuTTY, SSH Tectia

Mac & Linux users: Terminal (Just type `ssh`)

```
ssh andrewid@shark.ics.cs.cmu.edu
```

I Need You To Make A Directory

```
$ ls
$ cd private
$ mkdir 15-213
$ cd 15-213
```

File Transfers

- Download lab-handout.tar from Autolab
- Use MobaXTerm's file transfer dialog if you're on Windows
- On Linux or Mac OS X:

```
$ sftp andrew@shark.ics.cs.cmu.edu:private/15-213
sftp> help
(read help for 'cd', 'lcd', 'pwd, 'lpwd', 'get',
'put', etc.)
```

Also, you can use FileZilla! Here's a detailed guide:
http://cs.cmu.edu/~213/recitations/using_filezilla.pdf

Continue On...

```
$ ls
$ cd private
$ mkdir 15-213
$ cd 15-213
$ tar xvpf lab-handout.tar
$ cd lab-handout
```

Git


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Git Setup (User Information)

```
$ git config --global user.name "<Your Name>"
$ git config --global user.email <Your email>
```

Sample Git Commit

In a new folder (mkdir)

```
$ git init
$ echo “a sample file” > readme.txt
```

To save your progress:

```
$ git add readme.txt
$ git commit -m “my first commit”
```

Git Setup

```
$ echo "a second line" >> readme.txt
```

To save your progress:

```
$ git add -u
```

(includes changed files in the commit that were previously added to git)

```
$ git commit -m "second commit"
$ git log
```

*(shows previous commits - **commit hash**)*

```
$ git reset --hard HEAD~1
```

(deletes last commit and reverts back to the first commit - DO WITH CARE - use revert or something similar if you may want to keep the changes)

Git Ignore

For those who want to use

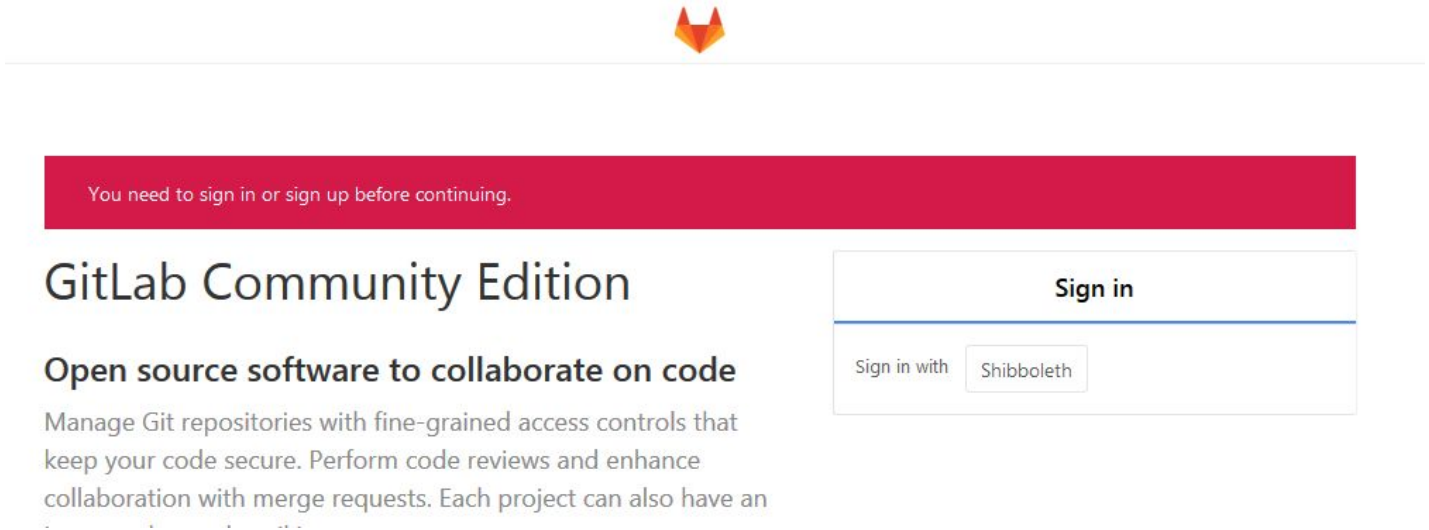
```
git add -all or git add *
```

Create a file `.gitignore` in your git repository and add files that you do not want to track

Further reading: https://git-scm.com/docs/gitignore

GitLab Setup

Sign into GitLab through Shibboleth



GitLab Setup

Create a new project in GitLab, call it “sample-lab”

Make sure the projects you create a private repository!

GitLab Setup

Create a new folder in your current directory (simulating copying over your lab-handout)

Follow the instructions in **Existing folder** to save your new folder to the Gitlab repository

```
...
$ git remote add origin
http://git.ece.cmu.edu/andrewid/sample-lab.git
...
$ git push -u origin master
```

Avoid authenticating every push to Gitlabs

How to generate ssh-keys for different systems:

https://docs.gitlab.com/ce/ssh/README.html

How to add your ssh-key:

https://docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html

Example ssh key set up on Linux

First check if you already have an ssh key. If not:

```
$ ssh-keygen -t rsa -C "GitLab" -b 4096
```

Use the default file path (press Enter), and optionally type in a password. (press Enter if you don't want one)

```
$ cat ~/.ssh/id_rsa.pub
```

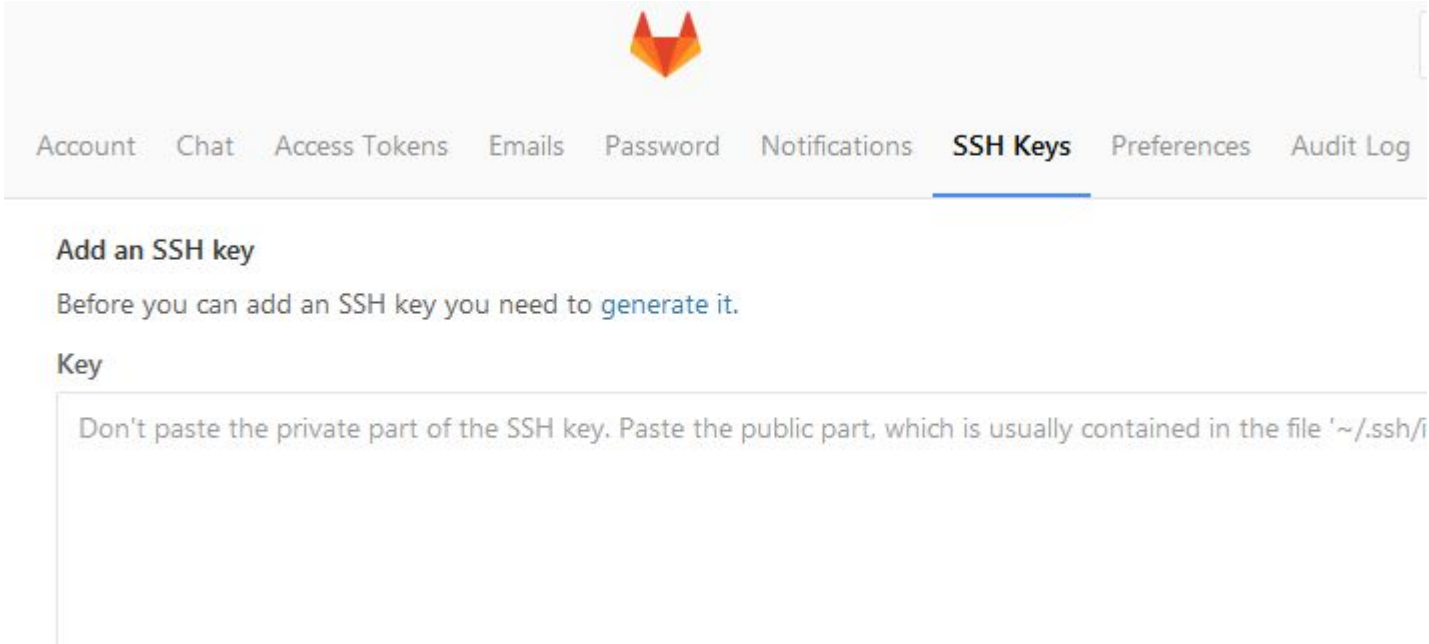
Your public key will be printed.

Highlight it with the mouse and copy

- Cmd+c if you are on a mac
- Ctrl+c if you are on windows

GitLab ssh Keys Setup

Paste the public SSH key into the text field here.



Git Commands

`add`	Stage new or changed files	`rebase`	Modify, combine, delete, ... previous commits
`commit`	Save current staged files	`merge`	Combine commits from specified branch into current branch
`push/pull`	Push/pull local index to/from the remote server	`checkout`	Examine a different commit/branch/file
`log`	Show history of git commits	`stash`	Temporarily save your current uncommitted changes
`status`	Shows working directory status (added/modified/deleted files)	`stash pop`	Restore previously stashed changes
`show`	Show a file from a different commit or branch	`diff`	Show changes between commits, files, unstaged changes, ...
`branch`	Create a new branch (use a new branch for experimenting safely)	`clone`	Clone a git repository (like a remote GitLab repo)

More Git

Getting help:

- `git help <command>`
- Piazza/Office hours

Git tutorials:

- https://www.atlassian.com/git/tutorials
- https://try.github.io

Terminal Shortcuts

The command line operates on one directory at a time (the “working directory”).

You can use these shortcuts whenever a directory or file path is expected.

	Meaning	Example
`~`	Home directory	`cp foo.txt ~`
`.`	Working (current) directory	`cp ~/foo.txt .`
`..`	Parent directory	`cp ~/foo.txt ..`
`-`	Previous directory	`cd -`
`*`	Match as many characters as possible	`cp ~/*.txt` `rm *.c`

- **Be very *very* ***very*** careful with rm!!!**
  - **There is no trash with `rm`. It is gone.**

More Terminal Shortcuts

- Pressing tab will autocomplete file/directory names.
- Use the up+down arrow keys to scroll through your previous commands.
- Control+R lets you search your command history.
- Control+A jumps to the beginning of the line.
- Control+E jumps to the end of the line.
- Control+U clears everything to the left of the cursor.
- Control+C kills your current program.
- Control+D (on a blank line) exits the terminal.
- Control+L clears your screen.

`ls <dir>`

- Lists files in the present working directory, or, if specified, `dir`.
  - `-l` lists ownership and permissions.
  - `-a` shows hidden files (“dotfiles”).
- `pwd` tells you your present working directory.

`cd <directory>`

- Try running `cd -` to return to the previous directory.
- Try running `cd ..` to return to the parent directory.
- Changes your present working directory.

`mkdir <dirname>`

- Makes a directory `dirname` in your present working directory.
- Directories and folders are the **same thing!**

`mv <src> <dest>`

- `cp` works in exactly the same way, but copies instead
  - for copying folders, use `cp -r`
- `dest` can be into an existing folder (preserves name), or a file/folder of a different name
- `src` can be either a file or a folder

`tar <options> <filename>`

- For full list of options, see `man tar`
- `tar` stands for **t**ape **ar**chive. Was used on tapes!
- `x` - extract, `v` - verbose, `f` - file input, `p` - keep perms
- All of our handouts will be in `tar` format.

`rm <file1> <file2> … <filen>`

- To remove an (empty) directory, use `rmdir`
  - To remove a folder and its contents, use `rm -rf`
    - **Please be careful, don't delete your project.**
    - **There is no "Trash" here. It's gone.**
    - **Contact ugradlabs@cs.cmu.edu to restore.**
    - **Latest restore is up to a day old!**

- **Restore most recent version yourself if you use git!**

pipes and redirects

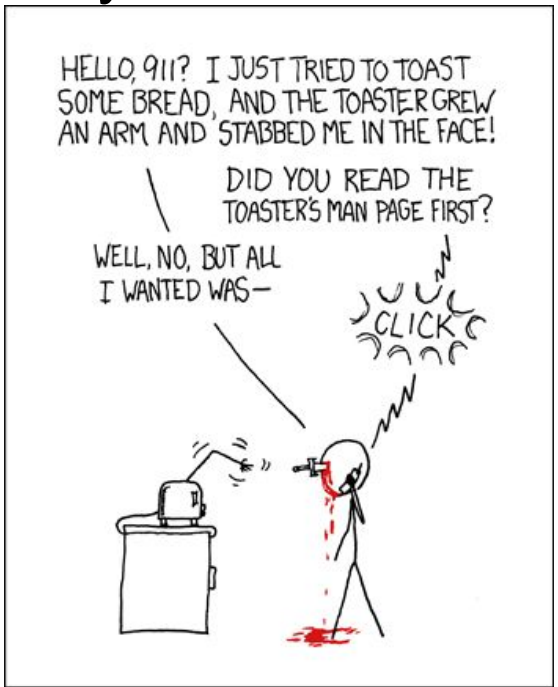
- A *pipe* redirects output from one program as input to another program.
  - Ex: `cat filename | outputfile`
  - Ex: `cat filename | grep 15213`
  -
- Can *redirect* output to a file.
  - Ex: `echo hello > file.txt` writes "hello" **over** `file.txt`.
  - Ex: `echo hello >> file.txt` *appends* "hello" to `file.txt`.

What's in a file? (using `cat`)

- `cat <file1> <file2> … <filen>` lets you display the contents of a file in the terminal window.
  - Use `cat -n` to add line numbers!
- You can *combine* multiple files into one!
  - `cat <file1> … <filen> >> file.txt`
- Good for seeing what's in small files.
- Try `cat -n bomb.c`. Too big, right?

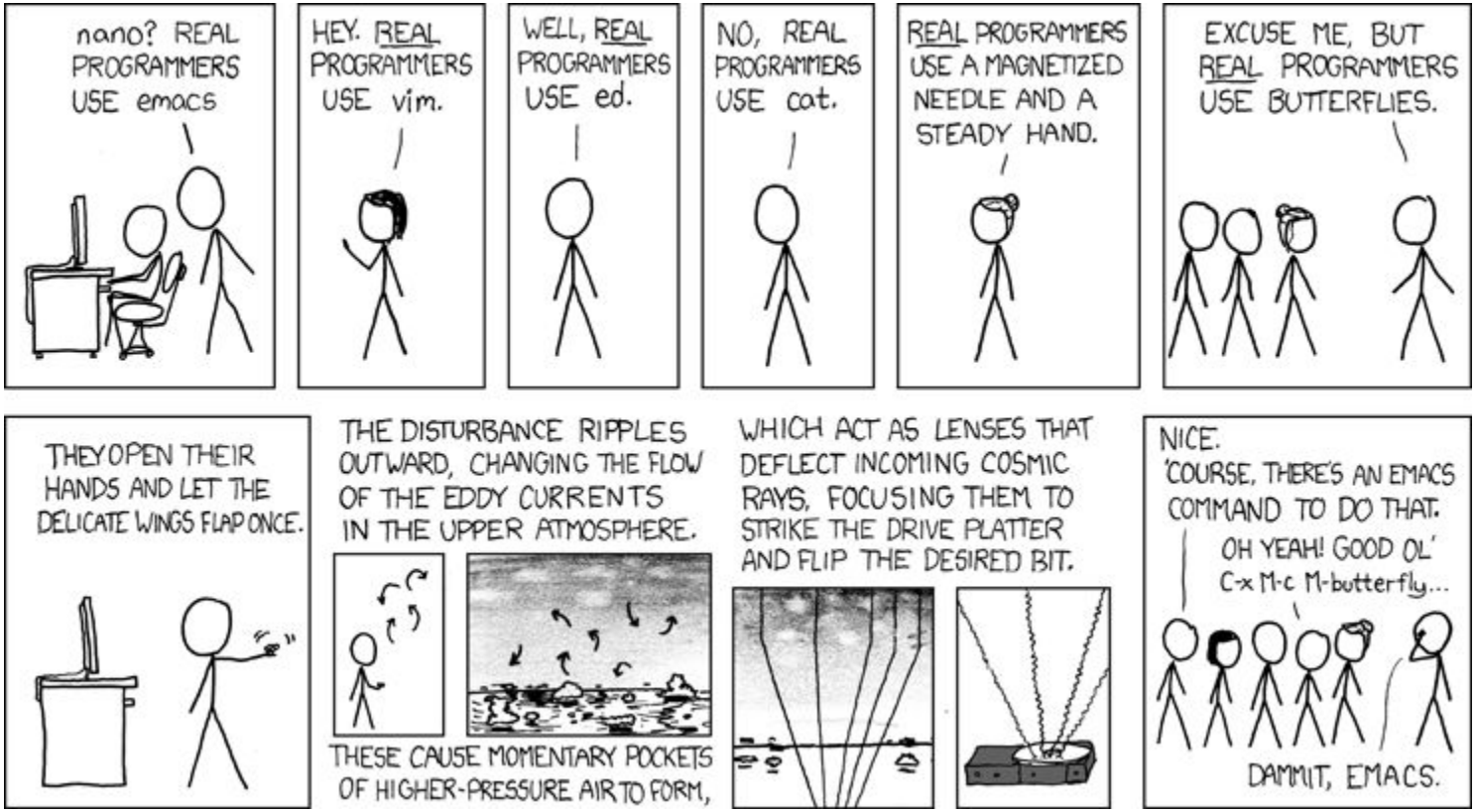
man <thing>

- What is that command? What is this C standard library function? What does this library do?
- Pages viewed with `less`
- Try it!
  - `man grep`
  - `man tar`
  - `man strlen`
  - `man 3 printf`
  - `man stdio.h`
  - `man man`



Appendix

Editors (a touchy subject)



Editors (a touchy subject)

- `vim` is nice, made for very powerful text editing
  - Try running `vimtutor` to get started learning
- `emacs` is nice, made to be more versatile
  - Emacs tutorial in emacs: “Ctrl-h t”
- `gedit` has a GUI
  - Requires X Forwarding: See Appendix
- I **strongly** recommend editing on the terminal.
- **Gist**: Use an editor with auto-indent and line numbers

Configuring bash

The file `~/.bashrc` is run every time you log in.

Put the following code:

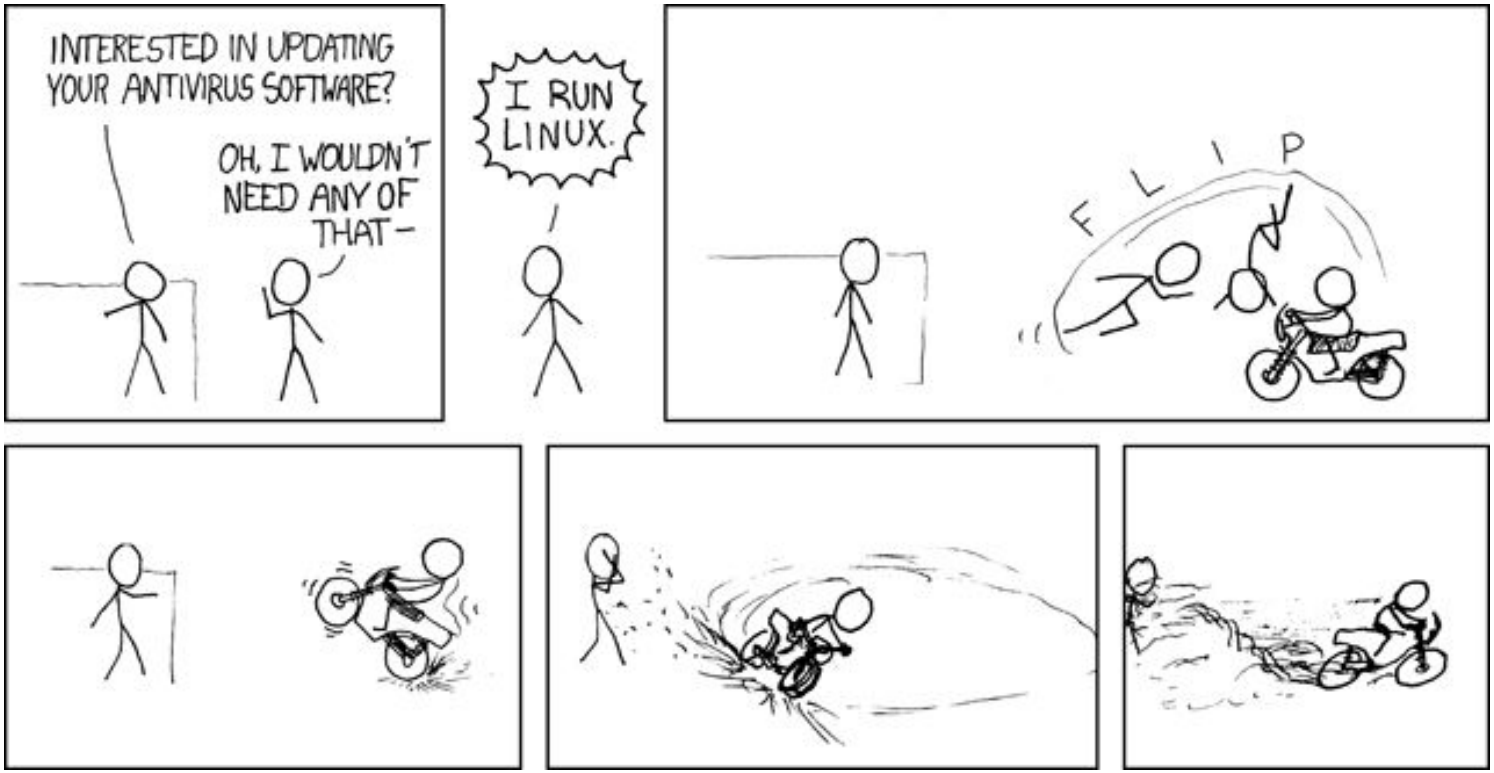
```
PS1="[\u@\h:\w] \$ "
alias ls='ls --color=auto'
```

to change your prompt to look like:

```
[szz@makoshark:~/private/15213] $ ls
attacklab  bomblab  lab-answers
```

Commands related to 15-213

- `gdb`, the **G**NU **Deb**ugger, will be used for bomb lab.
- `objdump` displays the symbols in an executable.
- `gcc` is the **G**NU **C** **C**ompiler.
- `make` is a configurable build system often used for compiling programs.
- We will provide other tools in the handouts as well


INTERESTED IN UPDATING YOUR ANTIVIRUS SOFTWARE?
OH, I WOULDN'T NEED ANY OF THAT-
I RUN LINUX.
F L I P

Vimtutor Walkthrough

- Chapters 1-3
- Cheatsheet: http://bit.ly/2cl0lJ0

Resources

Ask the Course Staff!

http://cs.cmu.edu/~213/help/

Resources

- Quick references: cs.cmu.edu/~213/resources.html
- CMU Computer Club
  - www.contrib.andrew.cmu.edu/~sbaugh/emacs.html
  - club.cc.cmu.edu/talks/fall15/power-vim.html
  - club.cc.cmu.edu/talks/fall15/power-git.html
- Great Practical Ideas
  - www.cs.cmu.edu/~15131/f15/topics/bash/
  - www.cs.cmu.edu/~15131/f15/topics/git/
- Official manuals
  - `info bash`
  - `info emacs`
  - `:help` in Vim

tmux

```
$ tmux
```

Ctrl+b, then c: create a new tab

Ctrl+b, then n: move to next tab

Ctrl+b, then p: move to previous tab

Ctrl+b, then x: kill the current tab

Ctrl+b, then ?: help

Ctrl+b, then ": split horizontal

Ctrl+b, then %: split vertical

Ctrl+b, then arrow keys: move between panes

Fancy Terminal Shortcuts

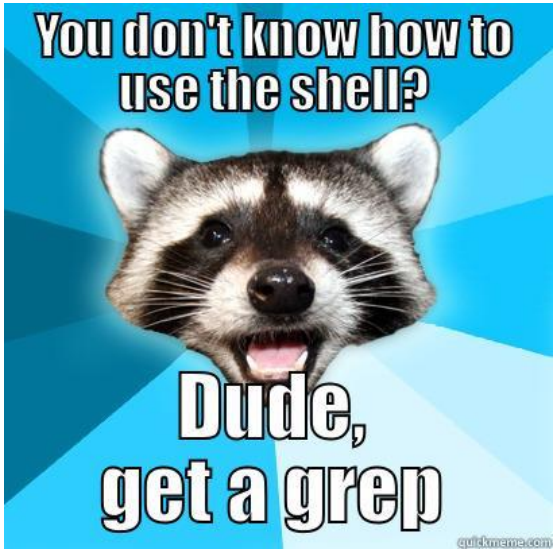
- Bash automatically splits things up in brackets!
  - Ex: `cp foo{1,2}.txt = cp foo1.txt foo2.txt`
  - Ex: `cp foo.txt{,.bak} = cp foo.txt foo.txt.bak`
  - For when typing the same filename gets annoying
- Bash has `for` loops!
  - Ex: Append “15-213” to every file ending in .c

```
for file in *.c; do echo “15-213” >> $file; done
```

- Have fun, but don’t break things or lose track of time

What's in a file? (using `grep`)

- `grep <pattern> <file>` will output any lines of `file` that have `pattern` as a substring
  - `grep -v` will output lines *without* pattern as substring
  - `grep -n` prints line numbers
  - `grep -R` will search *recursively*
- Try it: `grep 'phase' bomb.c`
  - `grep -n 'printf' src.c`
  - `grep -R 'unsigned' .`



Looking for something? `grep -A -B`

- `grep -B <x>`: include x lines **B**efore match.
- `grep -A <y>`: include y lines **A**fter match.
- Ex: `objdump -d | grep -A 25 explode_bomb`
- Ex: `grep -B 20 return *.c`