18-600 Recitation #1
Boot Camp

August 30th, 2016

Welcome to 18-600!

- Purpose of recitation
  - Useful tools, information pertaining to the labs
  - Hands-on activities
  - Problem solving and exam prep
  - Last ~30 mins of recitation will be used as OH time, for 1-on-1 questions
- Dos and Don’ts
  - DO start the labs early
  - DO ask questions (Piazza, recitation, & OH) well before deadlines
  - DO come to recitation
  - DON’T cheat. Seriously. CMU’s advanced cheating software/smart TAs are excellent at identifying this.
  - DON’T use office hours in replacement of recitation

Today - Boot Camp!

Getting started with your code development environment (Linux, AFS, Autolab, Vim, etc.)

Connecting

SSH

Windows users: PuTTY
(http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

Mac/Linux users: Use ‘ssh’ command at terminal

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

Files

Windows: Tectia file transfer

Mac/Linux users: Use ‘scp’ command at terminal:

scp –r andrewid@shark.ics.cs.cmu.edu:~/private/myfolder /some/local/folder

scp myfile.c andrewid@shark.ics.cs.cmu.edu:~/private/myfolder

FOR THE LOVE OF ALL THAT IS HOLY AND SACRED, **USE THE SHARK MACHINES FOR WORKING ON ALL OF YOUR ASSIGNMENTS**

Welcome!

```
$ ls
$ cd private
$ mkdir 18-600
$ cd 18-600
$ mv ~/Downloads/datalab-handout.tar .
$ tar xvf datalab-handout.tar
$ cd datalab-handout
```

Some Nice Terminal Shortcuts

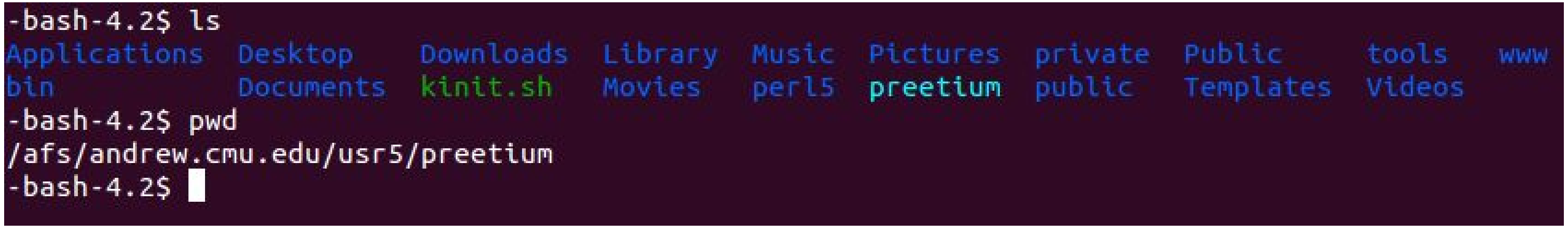
- Pressing *`tab`* will **autocomplete** file and folder names!
- `Control+C` will **stop** execution of your current program!
- `Control+R` will let you **search** your command history!
- `Control+L` will **clear** your screen!
- `cmd arg1 … argN > file1.txt` will put the output of `cmd` into `file1.txt`!
- `cmd arg1 … argN < file2.txt` will pull the input of `cmd` from `file2.txt`!
- Use the **up** and **down** arrow keys to **scroll through your command history**!

Linux file pathing

- ~ is your **HOME DIRECTORY**
  - This is where you start from after you SSH in
  - On bash, you can also use $HOME
- . is an alias for your **PRESENT WORKING DIRECTORY**!
- .. is the file path for the **PARENT DIRECTORY** of your present working directory!
- / is the file path for the **TOP-LEVEL DIRECTORY**
  - You probably won’t use this too much in this class

`ls <dir>` - LiSt

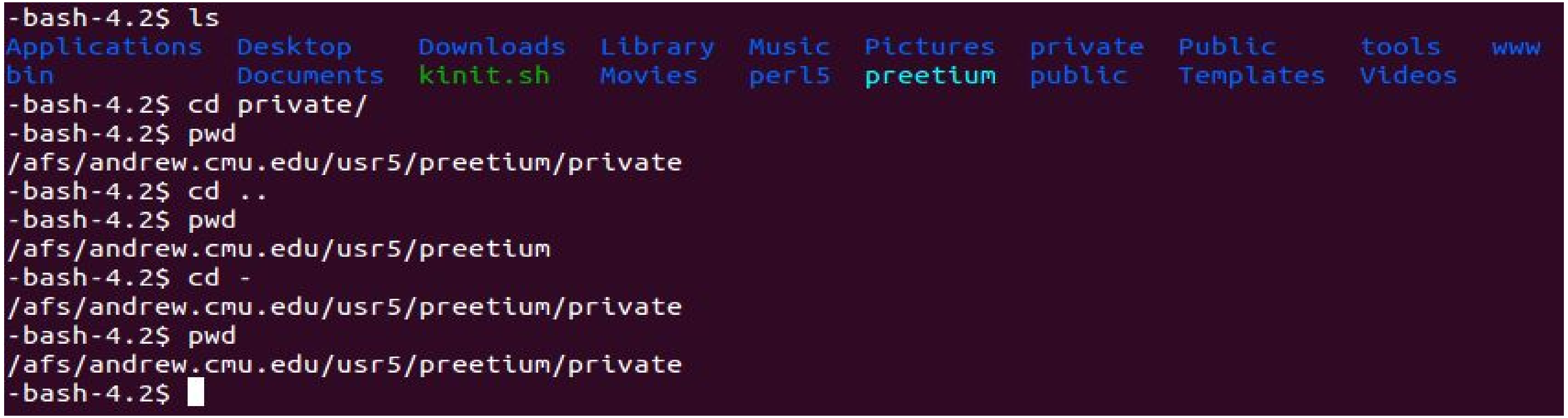
- Lists the files in the present working directory, or, if specified, `dir`.
  - -l lists ownership and permissions.
  - -a shows hidden files (“dotfiles”).
- `pwd` tells you your Present Working Directory.



```
-bash-4.2$ ls
Applications   Desktop     Downloads  Library  Music   Pictures  private  Public     tools   www
bin            Documents   kinit.sh   Movies   perl5   preetium  public   Templates  Videos
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium
-bash-4.2$ 
```

cd <directory> - Change Directory

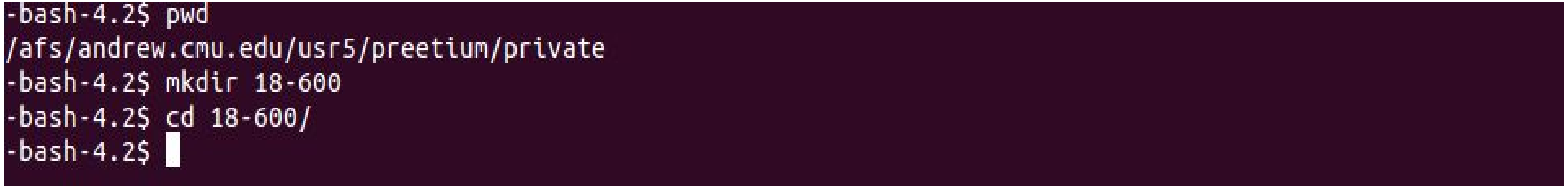
- Changes your present working directory to `directory`
  - Try running `cd -` to return to the previous directory.
  - Try running `cd ..` to return to the parent directory.
- Your main tool for navigating a unix file system



```
-bash-4.2$ ls
Applications  Desktop    Downloads  Library  Music  Pictures  private  Public     tools   www
bin           Documents  kinit.sh   Movies   perl5  preetium  public   Templates  Videos
-bash-4.2$ cd private/
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private
-bash-4.2$ cd ..
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium
-bash-4.2$ cd -
/afs/andrew.cmu.edu/usr5/preetium/private
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private
-bash-4.2$
```

mkdir <dirname> - MaKe DIRectory

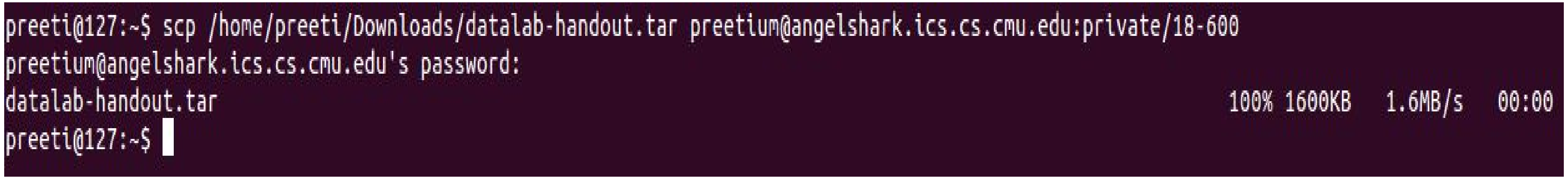
- Makes a directory `dirname` in your present working directory.
- Directories and folders are the **same thing!**



```
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private
-bash-4.2$ mkdir 18-600
-bash-4.2$ cd 18-600/
-bash-4.2$ 
```

`scp <src> <dest>`

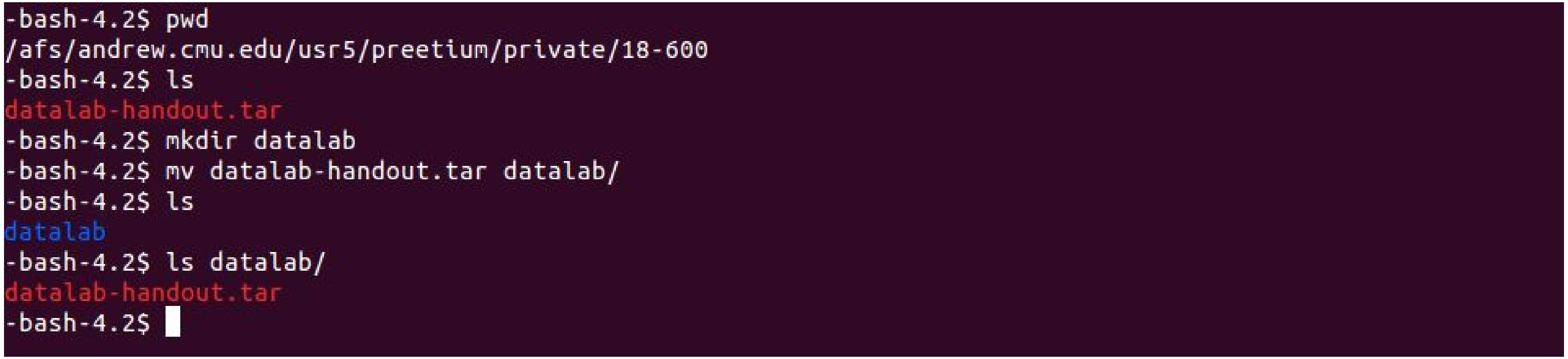
- Allows files to be copied to/from or between different hosts.
  - The full path to the remote host needs to be specified
  - Use the `-r` option to copy folders



```
preeti@127:~$ scp /home/preeti/Downloads/datalab-handout.tar preetium@angelshark.ics.cs.cmu.edu:private/18-600
preetium@angelshark.ics.cs.cmu.edu's password:
datalab-handout.tar                                                                        100% 1600KB   1.6MB/s   00:00
preeti@127:~$
```

mv <src> <dest> - MoVe

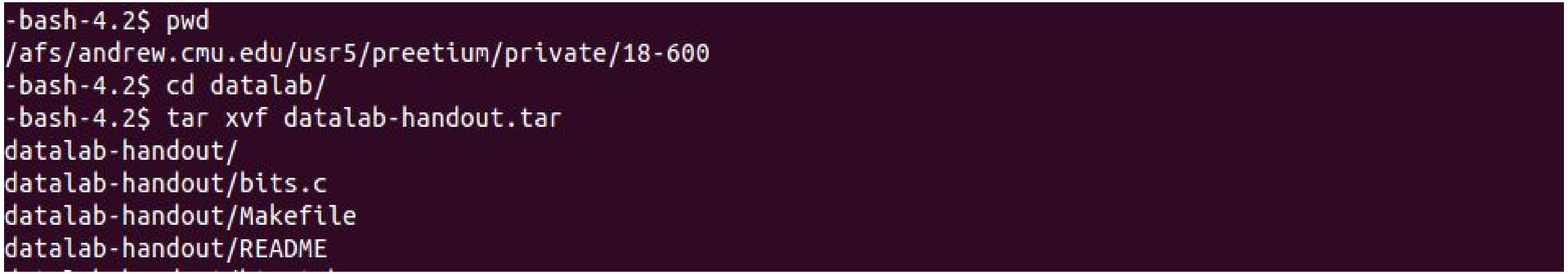
- `cp` works in exactly the same way, but copies instead
  - for copying folders, use `cp -r`
- `dest` can be into an existing folder (preserves name), or a file/folder of a different name
- Also used to re-name files without moving them
- `src` can be either a file or a folder



```
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private/18-600
-bash-4.2$ ls
datalab-handout.tar
-bash-4.2$ mkdir datalab
-bash-4.2$ mv datalab-handout.tar datalab/
-bash-4.2$ ls
datalab
-bash-4.2$ ls datalab/
datalab-handout.tar
-bash-4.2$
```

`tar <options> <filename>` - Tape ARchive

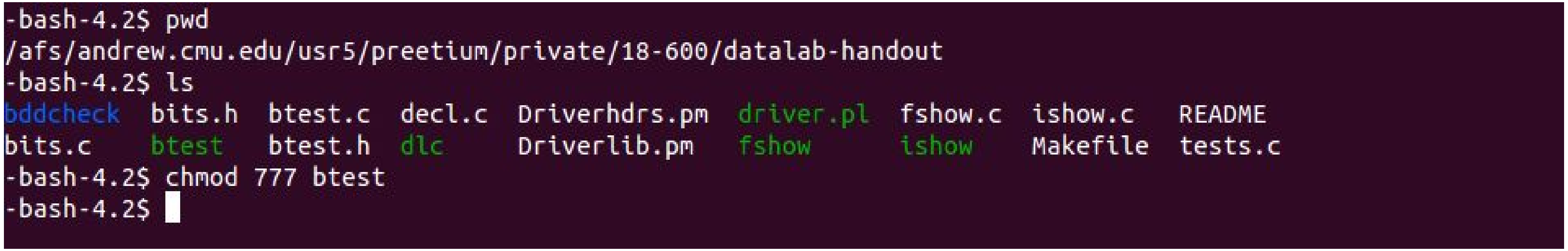
- Compression utility, similar to zip files on Windows
- For full list of options, see `man tar`
- As name suggests, was used on tapes!
- `x` - extract, `v` - verbose, `f` - file input
- All of our handouts will be in `tar` format.



```
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private/18-600
-bash-4.2$ cd datalab/
-bash-4.2$ tar xvf datalab-handout.tar
datalab-handout/
datalab-handout/bits.c
datalab-handout/Makefile
datalab-handout/README
```

`chmod <permissions> <src>`

- `chmod` is used to change the permissions of a file or directory.
  - `777` will give all permissions
  - `src` can be either a file or a folder



```
-bash-4.2$ pwd
/afs/andrew.cmu.edu/usr5/preetium/private/18-600/datalab-handout
-bash-4.2$ ls
bddcheck  bits.h  btest.c  decl.c  Driverhdrs.pm  driver.pl  fshow.c  ishow.c   README
bits.c    btest   btest.h  dlc     Driverlib.pm   fshow      ishow    Makefile  tests.c
-bash-4.2$ chmod 777 btest
-bash-4.2$ 
```

`rm <file1> <file2> … <filen>` - ReMove

- Essentially the delete utility
- 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.**
    - **If someone asks you to use** `rm -rf /` **ignore them**

pipes and redirects

- A *pipe* redirects output from one program as input to another program.
  - Ex: `objdump -d a.out | grep “mov”`
  - Ex: `ls *.c | grep malloc`
  - Ex: `ls -l | grep jbiggs | wc -l`
- Can *redirect* output to a file.
  - Ex: `cmd arg1 … argn > file.txt` will write the output of `cmd` **over** `file.txt`.
  - Ex: `cmd arg1 … argn >> file.txt` will append the output of `cmd` to `file.txt`.

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

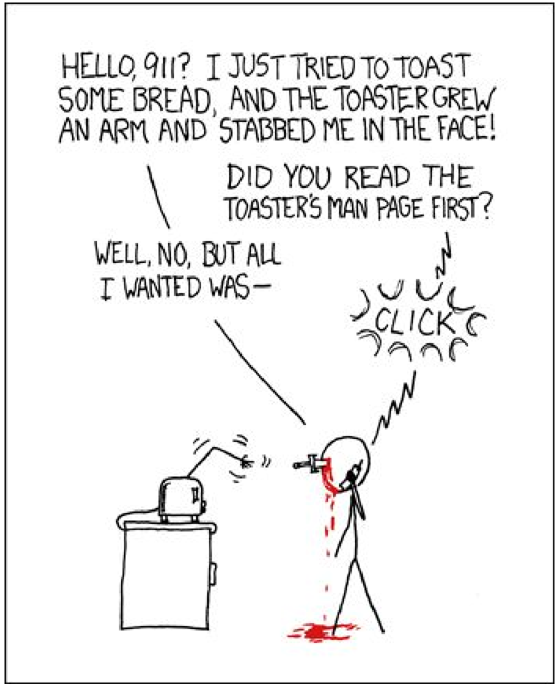
- `cat <file1> <file2> … <filen>` lets you display the contents of a file in the terminal window.
  - Use `cat -n` to add line numbers!
- You can *combine* multiple files into one!
  - `cat <file1> … <filen> > file.txt`
- Good for seeing what's in small files.
- Try `cat -n bits.c`. Too big, right?

What’s in a file? (using `less`)

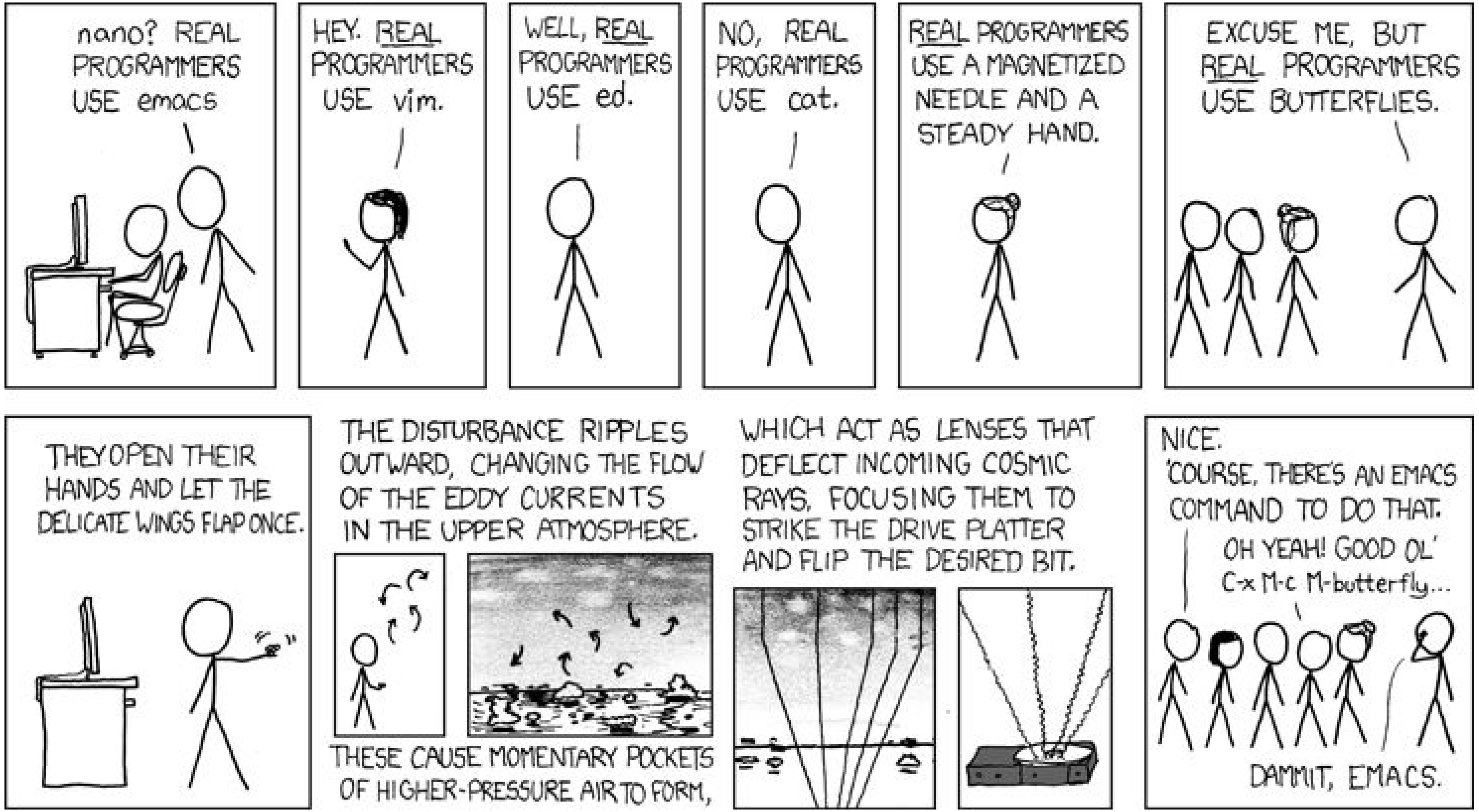
- `less <file>` will give you a scrollable interface for viewing large files **without** editing them.
  - To find something, use `/`
    - To view the next occurrence, press `n`
    - To view previous occurrence, press `N`
  - To quit, use `q`
- Try it: Type “`/isPower2`”

`man <thing>`

- What is that command? What is this C standard library function? What does this library do? Check to see if it has a `man` page!
- Pages viewed with `less`
- Try it!
  - `man grep`
  - `man tar`
  - `man printf`
  - `man strlen`


HELLO, 911? I JUST TRIED TO TOAST SOME BREAD, AND THE TOASTER GREW AN ARM AND STABBED ME IN THE FACE!
DID YOU READ THE TOASTER'S MAN PAGE FIRST?
WELL, NO, BUT ALL I WANTED WAS—
CLICK

Editors (a touchy subject)


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

Vim (vi – improved) Basics

- Some different modes:
  - Normal mode:
    - The first mode you enter. Hit the **escape key** to return to this mode at any time
    - Everything entered here is interpreted as a *command*
  - Command-line mode:
    - Used for entering *editor commands* (necessary to save file & quit the editor)
    - Enter “**:**” in Normal mode to get to this mode
  - Insert mode:
    - To edit text
    - Enter “**i**” in Normal mode to get to this mode

Vim Basics

- Useful commands:
  - Copying/pasting/deleting lines:
    - yy (yank) or 5 yy (yank next 5 lines)
    - dd (delete) or 5 dd (delete next 5 lines)
    - p (paste)
  - Search (/search_string or ?search_string)
- Useful editor commands:
  - Write (w)
  - Quit (q) quit no-save (q!)

Vimrc File

- Stores vim configuration info
- Can make your editing experience even better!
- Notably:
    - Smart indentation
    - Line numbers
    - Changing tabs to default to 2 or 4 spaces
    - Colors

- To edit, type: vim ~/.vimrc

More resources on Vim

- A good intro tutorial: http://www.engadget.com/2012/07/10/vim-how-to/
- An interactive tutorial: http://www.openvim.com/
- man vim
- Google

Commands related to 18-600

- `gdb`, the **G**NU **Deb**ugger, will be used for attack lab
- `objdump -d` displays the symbols in an executable.
- `gcc` is the **G**NU **C** **C**ompiler.
- `make` reads a configuration file to run a series of commands. Often used for compiling your programs.
- We will provide other tools in the handouts as well

Labs

- All labs released/turned in through autolab:
  https://autolab.andrew.cmu.edu/courses/18600-f16
  More info on which files to upload will be in each handout.
  - Check out the **scoreboard** to see where you stand next to the rest of the class (uses nickname provided)
- First lab, **Data Lab**, out this Thursday at 11:59PM PDT (note the time zone), due exactly two weeks later
  - Familiarizing yourself with C, bit manipulation and floating point

Labs - More Info

- All labs will be due (submitted to Autolab) at **11:59PM PDT/PST on the posted due date**
- Can submit to autolab infinite times, but last submission is always the one officially used
- Do not use autolab to check your submissions while working on them, we provide drivers for you to check on your own for this. Autolab should be used when ready to submit.
- Autolab hates tabs. Make sure your editor uses spaces for indentation.

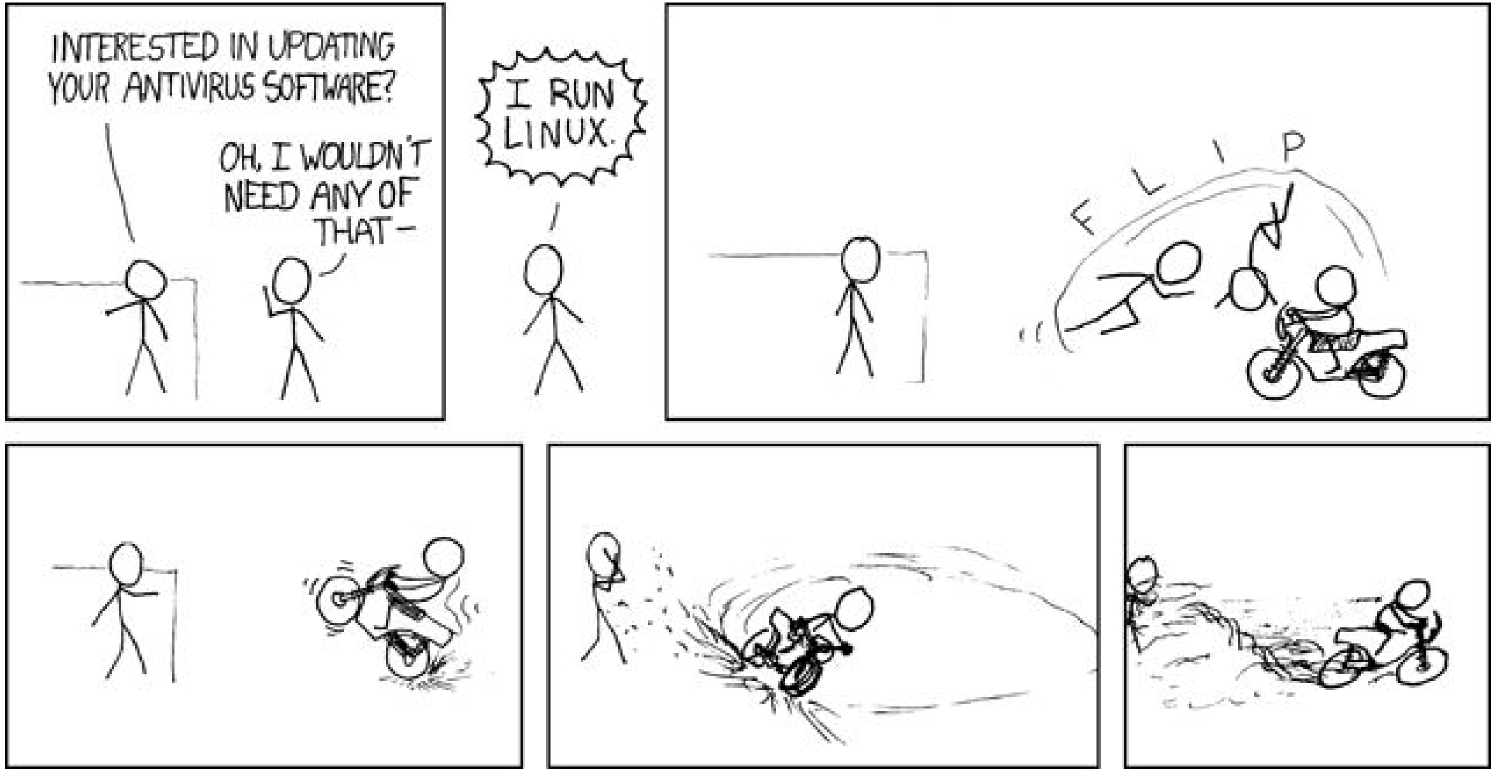
Labs - Final Thoughts

- Do not submit close to the deadline (Autolab may get overloaded)
- Use your late days wisely (5 total, up to 2 on any assignment). **Advice: Save them for Malloc.**
- Code style is an important part of this class, **points will be assigned for this** in the appropriate labs. Please refer to the style guide on the course website for what we expect.
- Maintain a backup of your lab solutions on your local machine
- **Please read the lab handout thoroughly before asking questions!**

Any questions?

Summary:

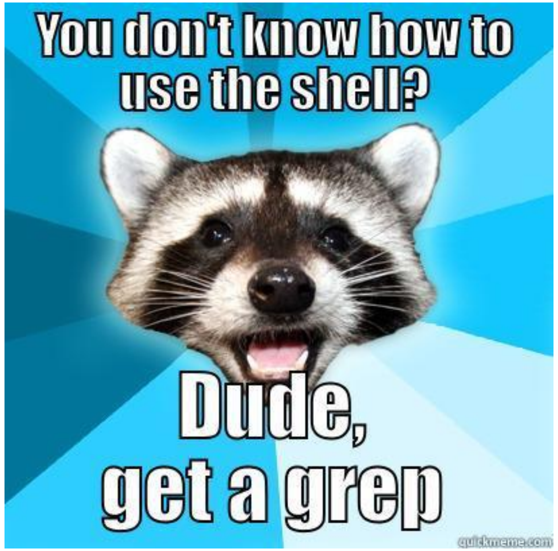
- Connecting to AFS
- Linux commands
- Vim Editor
- Labs


INTERESTED IN UPDATING YOUR ANTIVIRUS SOFTWARE?
OH, I WOULDN'T NEED ANY OF THAT–
I RUN LINUX.
FLIP

Appendix (for future reference)

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 -R` will search *recursively*
- Try it: `grep 'isPower2' bits.c`
  - `grep -v '*' bits.c`
  - `grep -R 'unsigned' .`



screen

- Window manager that allows you to run multiple programs simultaneously in different tabs
- Programs continue to run even if you get disconnected from the servers
- `screen` creates a new screen session
- `screen -ls` lists the currently active screen sessions and their ids
- `screen -r` resumes a detached screen session
- Each screen command consists of `<Ctrl-A>` followed by one other character

Alternative: **tmux**

More screen commands

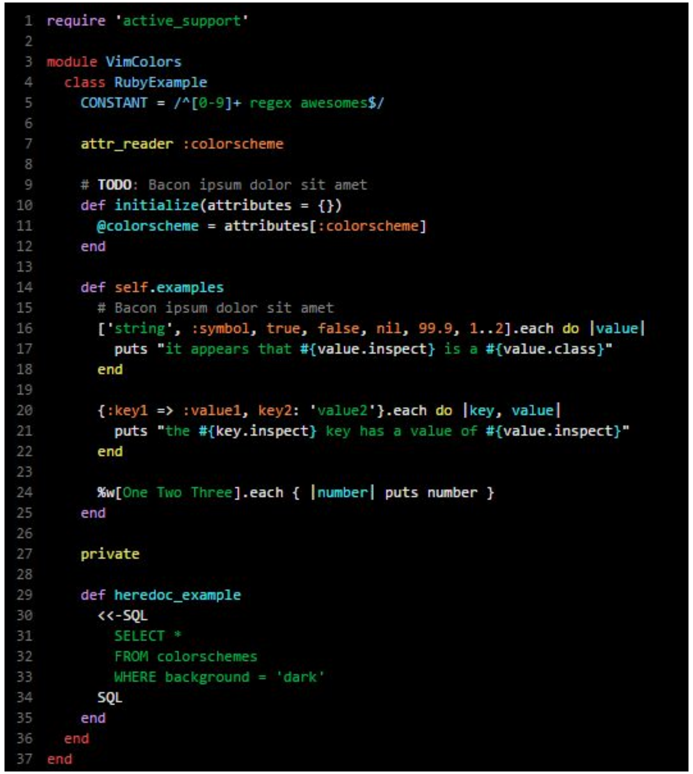
- <Control-a>, then press c: create new tab
- <Control-a>, then press k: kill current tab
  - Consider exiting bash rather than killing window (bad)
- <Control-a>, then press n: go to next tab
- <Control-a>, then press p: go to previous tab
- <Control-a>, then press <number>: go to tab <number>
- <Control-a>, then press a: send “Control-a” to window
- <Control-a>, then press ?: help

Git

- Tracks and maintains a log of the history of changes made to files.
- Changes can be grouped together and are assigned unique ids.
- You can use git to keep track of the progressive changes you make to your solution files.
- Git can also be used to keep track of different implementations of your solution. Especially useful for malloc lab.
- You can maintain a backup of your lab solutions by cloning the git repository on your local machine from the remote machine.
- Basic commands: git init, git add, git commit, git revert, git rebase, git clone, git pull, git status, git branch, git checkout, git stash.
- Definitely use git if you are already familiar with it by initializing your working directory as a git repository. Don't stress otherwise. It is not essential for labs.
- Resources: https://git-scm.com/doc

Vim colors

- Download a .vim color scheme file from the web (or make your own)
- Copy to ~/.vim/colors folder (make this folder if it doesn't exist)
- Some useful places to download color schemes:
  - http://vimcolors.com/
  - http://cocopon.me/app/vim-color-gallery/
- Makes your editor pretty!



```ruby
require 'active_support'

module VimColors
  class RubyExample
    CONSTANT = /^[0-9]+ regex awesomes$/

    attr_reader :colorscheme

    # TODO: Bacon ipsum dolor sit amet
    def initialize(attributes = {})
      @colorscheme = attributes[:colorscheme]
    end

    def self.examples
      # Bacon ipsum dolor sit amet
      ['string', :symbol, true, false, nil, 99.9, 1..2].each do |value|
        puts "it appears that #{value.inspect} is a #{value.class}"
      end

      {:key1 => :value1, key2: 'value2'}.each do |key, value|
        puts "the #{key.inspect} key has a value of #{value.inspect}"
      end

      %w[One Two Three].each { |number| puts number }
    end

    private

    def heredoc_example
      <<-SQL
        SELECT *
        FROM colorschemes
        WHERE background = 'dark'
      SQL
    end
  end
end
```

Jenna’s Basic Vimrc File

```
set tabstop=4
set shiftwidth=4
set expandtab
set viminfo='100,h
colorscheme desertedocean
set number
syntax on
filetype on
filetype indent on
filetype plugin on
set smartindent
match Error /\%81v.\+/
```