Linux Basics

Learn to love the terminal

- 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 and launch a new terminal
  - 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`.
- `pwd` tells you your Present Working Directory.

`cd <directory>` - Change Directory

- Changes your present working directory to `directory`
- Your main tool for navigating a unix file system

```
jbiggs@blueshark ~ $ ls
cover_letter.pdf  factorial.py   Movies    resume.pdf    test.wav
demo.py           foo2.py        Music     school        timer.py
Desktop           foo.txt        Pictures  solutions.py  www
display.py        Fravic.pdf     private   src
Documents         Library        public    Templates
Downloads         Minecraft.jar  Public    test.py
jbiggs@blueshark ~ $ cd private/
jbiggs@blueshark ~/private $
```

`mkdir <dirname>` - MaKe DIRectory

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

```
jbiggs@blueshark ~ $ ls
cover_letter.pdf  factorial.py   Movies     resume.pdf    test.wav
demo.py           foo2.py        Music      school        timer.py
Desktop           foo.txt        Pictures   solutions.py  www
display.py        Fravic.pdf     private    src
Documents         Library        public     Templates
Downloads         Minecraft.jar  Public     test.py
jbiggs@blueshark ~ $ cd private/
jbiggs@blueshark ~/private $ mkdir 15-213
jbiggs@blueshark ~/private $ cd 15-213
jbiggs@blueshark ~/private/15-213 $
```

`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

```
jbiggs@blueshark ~ $ cd private/
jbiggs@blueshark ~/private $ mkdir 15-213
jbiggs@blueshark ~/private $ cd 15-213
jbiggs@blueshark ~/private/15-213 $ mv ~/Downloads/datalab-handout.
tar .
```

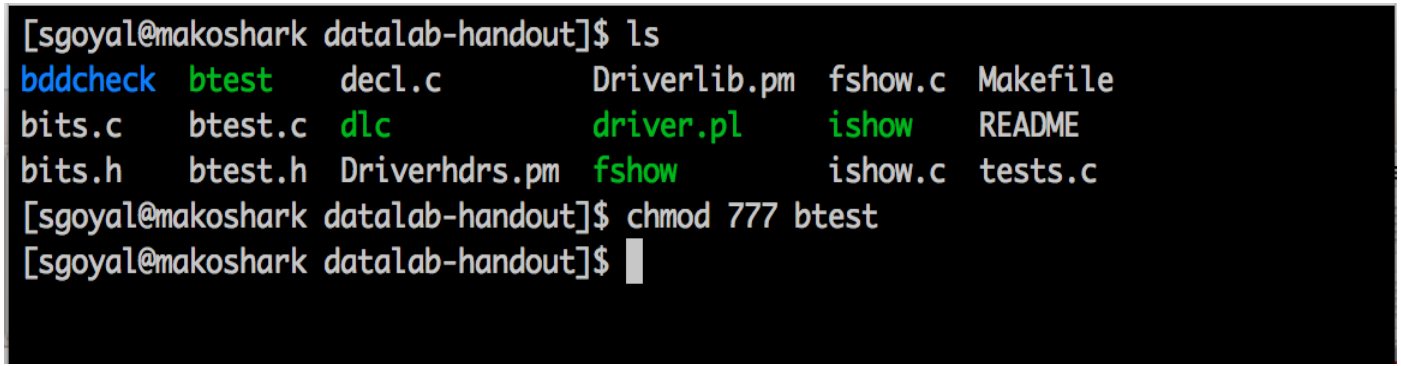
`tar <options> <filename>` - Tape ARchive

- 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.

```
jbiggs@blueshark ~/private/15-213 $ tar xvf datalab-handout.tar
datalab-handout/
datalab-handout/bits.c
datalab-handout/Makefile
datalab-handout/README
datalab-handout/btest.h
datalab-handout/btest.c
datalab-handout/bits.h
datalab-handout/decl.c
datalab-handout/tests.c
datalab-handout/fshow.c
```

`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



`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

```
[sgoyal@makoshark datalab-handout]$
[sgoyal@makoshark datalab-handout]$
[sgoyal@makoshark datalab-handout]$
[sgoyal@makoshark datalab-handout]$ scp -r bovik@shark.ics.cs.cmu.edu:/afs/andrew
.cmu.edu/usr/bovik/private/15213/datalab-handout some/local/folder
```

`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**

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

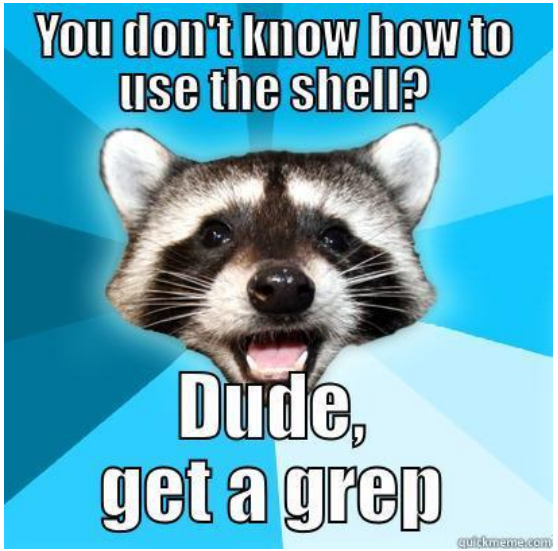
- `cat <file1> <file2> … <filen>` lets you display the contents of a file in the terminal window.
  - Use `cat -n` to add line numbers!
- You can *combine* multiple files into one!
  - `cat <file1> … <filen> > file.txt`
- Good for seeing what's in small files.
- Try `cat -n bits.c`. Too big, right?

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`"

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

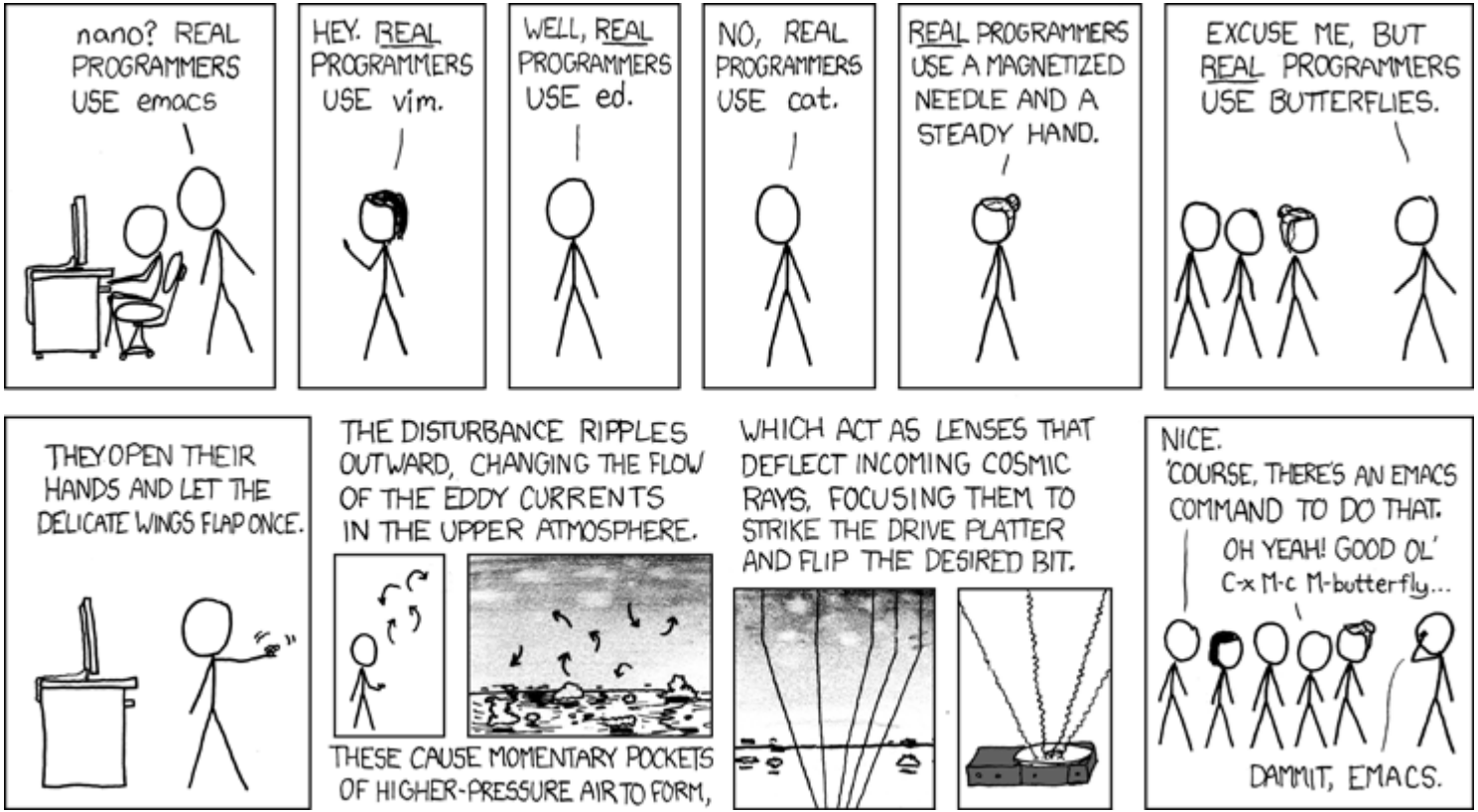
- `grep <pattern> <file>` will output any lines of `file` that have `pattern` as a substring
  - `grep -v` will output lines *without* pattern as substring
  - `grep -R` will search *recursively*
- Try it: `grep 'isPower2' bits.c`
  - `grep -v '*' bits.c`
  - `grep -R 'unsigned' .`



`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`

Editors (a touchy subject)



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

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
```

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

Binaries that we will use in the course

- `gdb`, the **G**NU **D**e**b**ugger, will be used for bomb 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