Intro to Unix & Bash

Getting started with the CLI

galvanize

Objectives

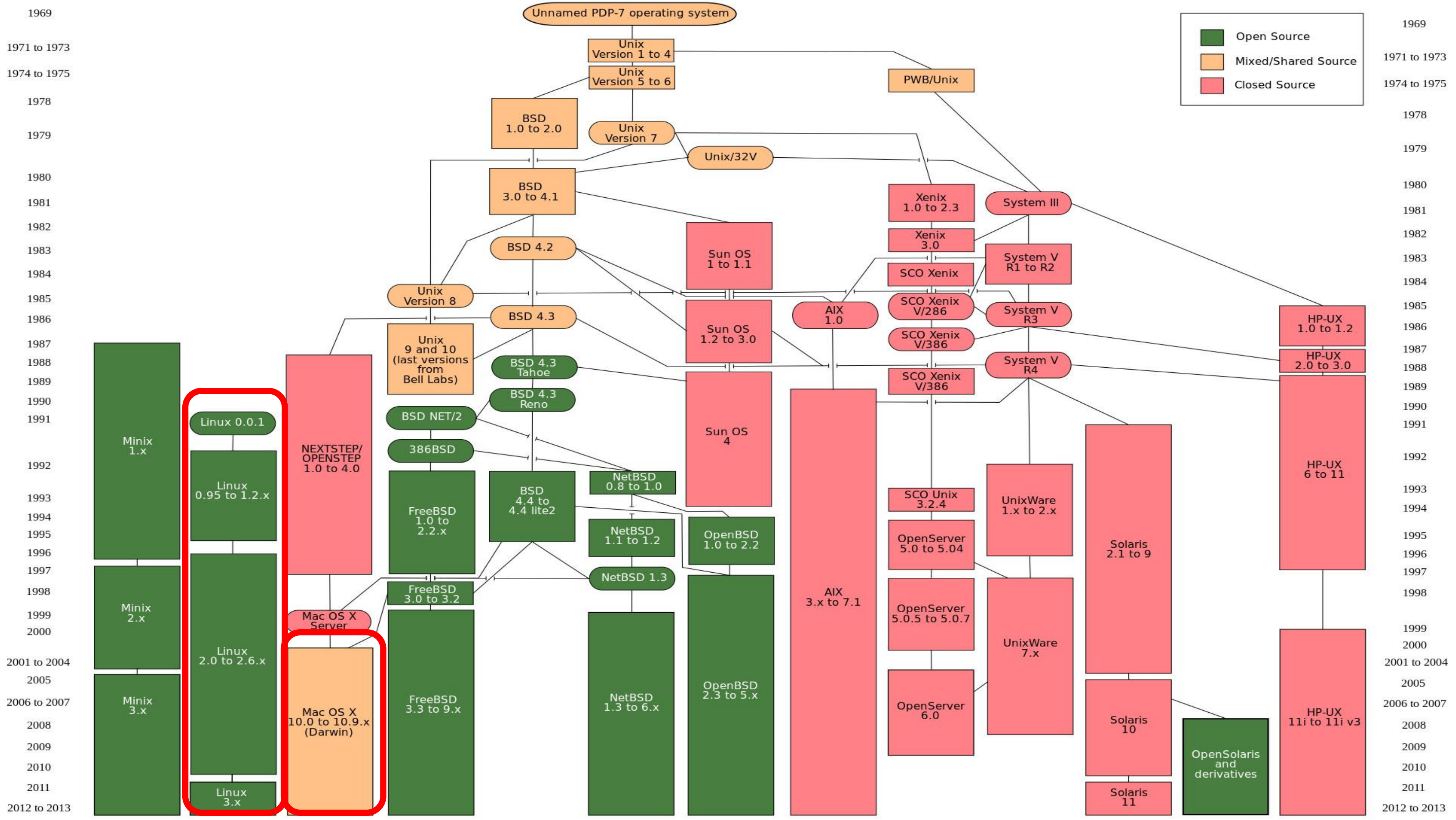
- Explain what UNIX is
- Explain the difference between Bash and the terminal
- Use bash commands:
  - man
  - pwd, ls, cd
  - mv, rm, cp, mkdir
  - chmod
- Create, inspect or modify a file on the command line
  - emacs/vim/nano
  - more/less/head/tail
  - grep, wc
- Use command piping

Unix

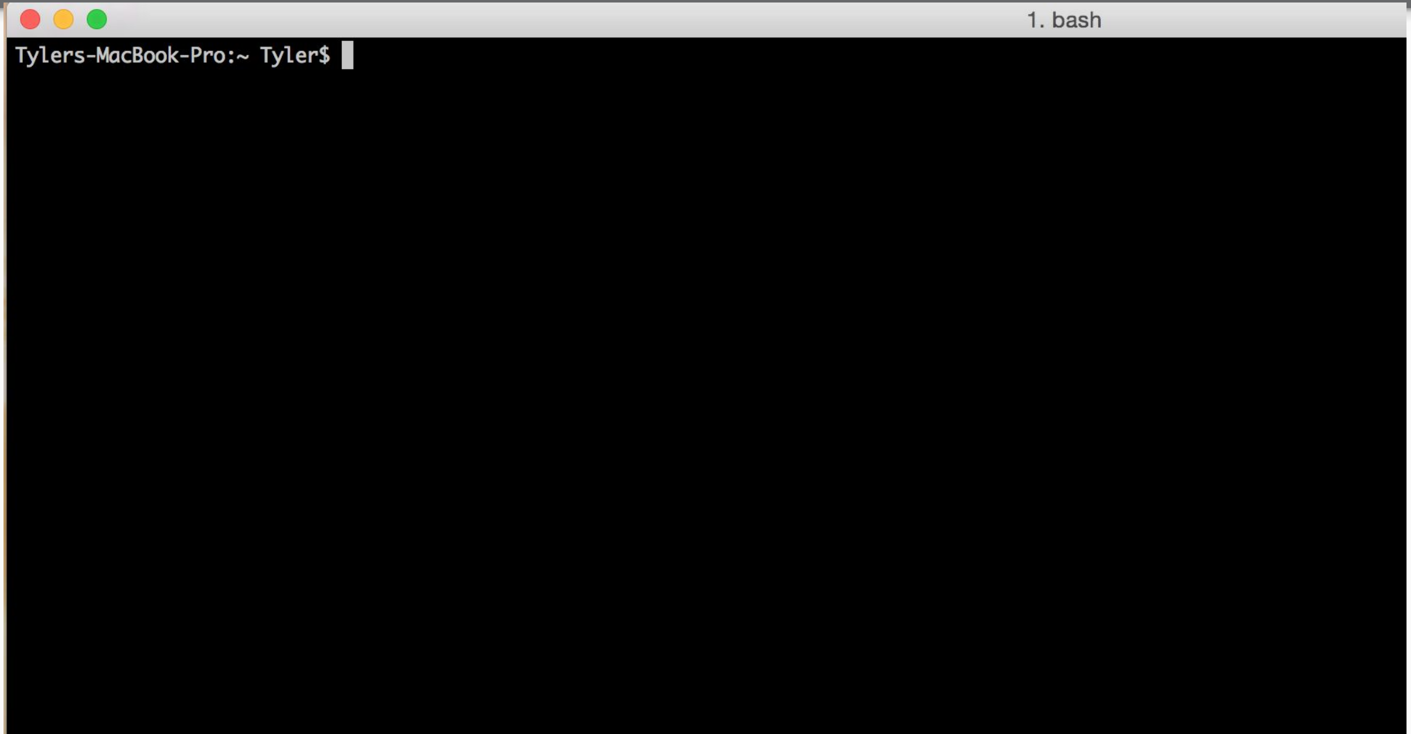
Unix is an operating system

Unix per se doesn’t exist anymore. Instead when we say Unix we often mean Unix-like or Unix-based, referring to Linux or MacOS and similar operating systems

Bash is a type of shell (arguably the most popular one)


Open Source
Mixed/Shared Source
Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
System V R3
AIX 1.0
BSD 4.3
Sun OS 1.2 to 3.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
Unix 9 and 10 (last versions from Bell Labs)
Minix 1.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
BSD 4.3 Tahoe
HP-UX 2.0 to 3.0
System V R4
SCO Xenix V/386
BSD 4.3 Reno
Sun OS 4
AIX 3.x to 7.1
BSD NET/2
Linux 0.0.1
386BSD
HP-UX 6 to 11
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
SCO Unix 3.2.4
UnixWare 1.x to 2.x
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
Minix 2.x
FreeBSD 3.0 to 3.2
Mac OS X Server
Linux 2.0 to 2.6.x
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
HP-UX 11i to 11i v3
Mac OS X 10.0 to 10.9.x (Darwin)
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
OpenBSD 2.3 to 5.x
OpenServer 6.0
Minix 3.x
Solaris 10
OpenSolaris and derivatives
Linux 3.x
Solaris 11

The command line interface (CLI)



```
Tylers-MacBook-Pro:~ Tyler$
```

The command line interface (CLI)

The CLI is one of two interfaces between user and computer, the other one being the graphical user interface (GUI), which your parents are used to

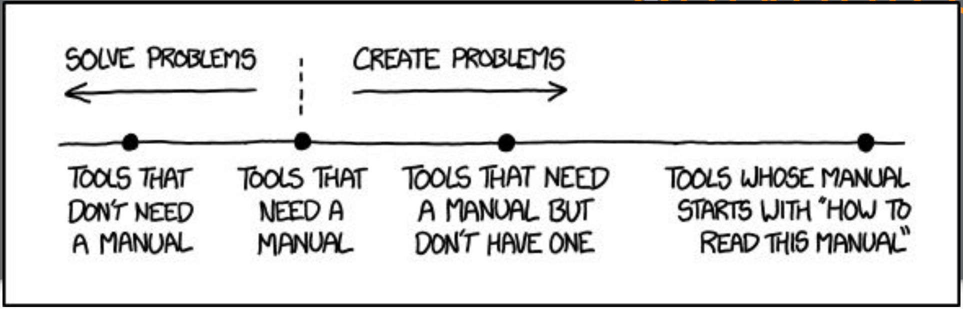
Many words are often used synonymously:

terminal, console, command line, command prompt, shell, bash, tty

Technically:

- shell = command line interpreter (interprets commands typed into console)
- bash = a (very popular) type of shell
- virtual terminal/virtual console/terminal emulator = text input/output environment (wrapper program running the shell)

The manual


SOLVE PROBLEMS
CREATE PROBLEMS
TOOLS THAT DON'T NEED A MANUAL
TOOLS THAT NEED A MANUAL
TOOLS THAT NEED A MANUAL BUT DON'T HAVE ONE
TOOLS WHOSE MANUAL STARTS WITH "HOW TO READ THIS MANUAL"

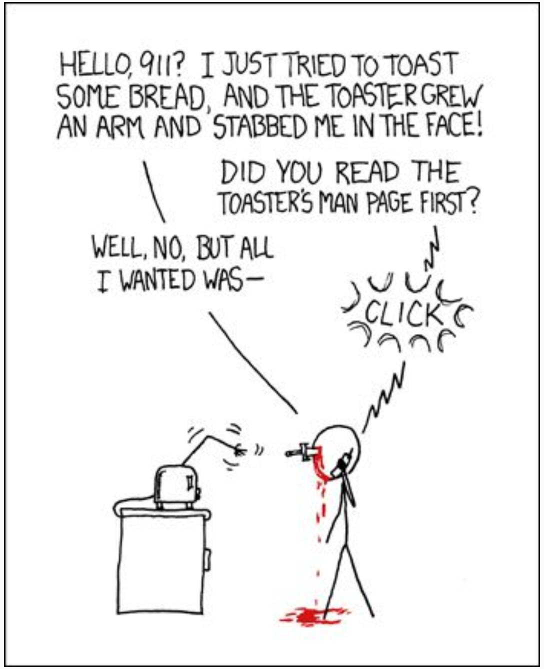
Type the following into your terminal:

```
$ man man
```

The “man pages” are very helpful.

www.explainshell.com helps too!

Sometimes it’s easier to find examples on the net. Even old material is good for these commands, because they rarely change


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

Basic commands in the terminal

Where am I?

`$ pwd`

Who am I?

`$ whoami`

Open current location in Finder:

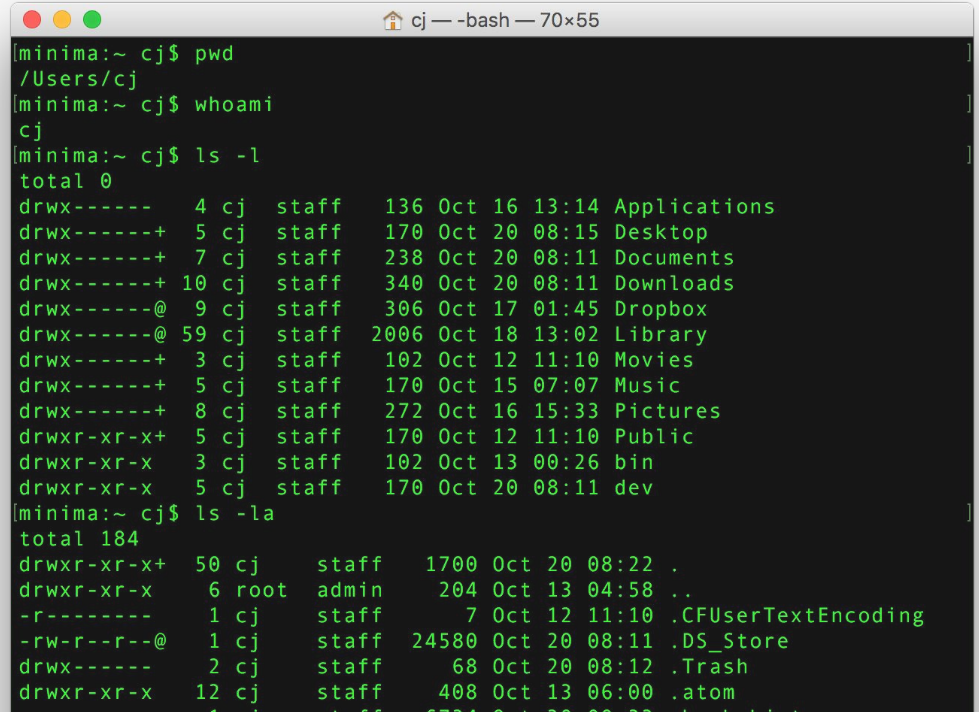
`$ open .` *(dot means current folder, can also specify folder or file name)*

Open current location in Atom:

`$ atom .` *(or path or filename)*

Open current location in Emacs:

`$ emacs .` *(or path or filename)*



```
minima:~ cj$ pwd
/Users/cj
minima:~ cj$ whoami
cj
minima:~ cj$ ls -l
total 0
drwx------    4 cj  staff   136 Oct 16 13:14 Applications
drwx------+   5 cj  staff   170 Oct 20 08:15 Desktop
drwx------+   7 cj  staff   238 Oct 20 08:11 Documents
drwx------+  10 cj  staff   340 Oct 20 08:11 Downloads
drwx------@   9 cj  staff   306 Oct 17 01:45 Dropbox
drwx------@  59 cj  staff  2006 Oct 18 13:02 Library
drwx------+   3 cj  staff   102 Oct 12 11:10 Movies
drwx------+   5 cj  staff   170 Oct 15 07:07 Music
drwx------+   8 cj  staff   272 Oct 16 15:33 Pictures
drwxr-xr-x+   5 cj  staff   170 Oct 12 11:10 Public
drwxr-xr-x    3 cj  staff   102 Oct 13 00:26 bin
drwxr-xr-x    5 cj  staff   170 Oct 20 08:11 dev
minima:~ cj$ ls -la
total 184
drwxr-xr-x+  50 cj    staff   1700 Oct 20 08:22 .
drwxr-xr-x    6 root  admin    204 Oct 13 04:58 ..
-r--------    1 cj    staff      7 Oct 12 11:10 .CFUserTextEncoding
-rw-r--r--@   1 cj    staff  24580 Oct 20 08:11 .DS_Store
drwx------    2 cj    staff     68 Oct 20 08:12 .Trash
drwxr-xr-x   12 cj    staff    408 Oct 13 06:00 .atom
```

Basic navigation in the terminal

List directory contents:

`$ ls`, `$ ls -l`, `$ ls -la` *(things with a - are called “options” or “flags”)*

Change Directory:

`cd [relative or absolute path]` (go to specified location)

`cd ~` (tilde is an alias for your home directory)

`cd` (without argument, same as “cd ~”)

`cd ..` (“..” means parent directory)

`cd /` (“/” means root directory)

`cd -` (“-” means last directory)

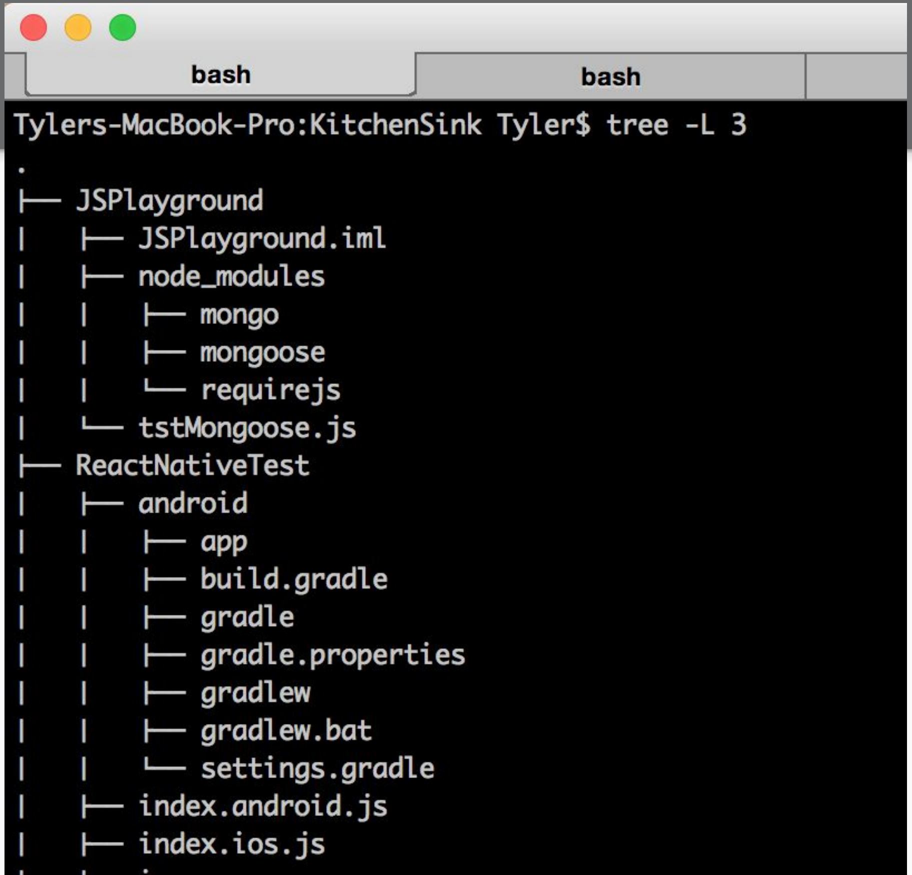
Whenever changing directories, orient yourself with `pwd` and `ls`

tree

Your Filesystem is a “tree”

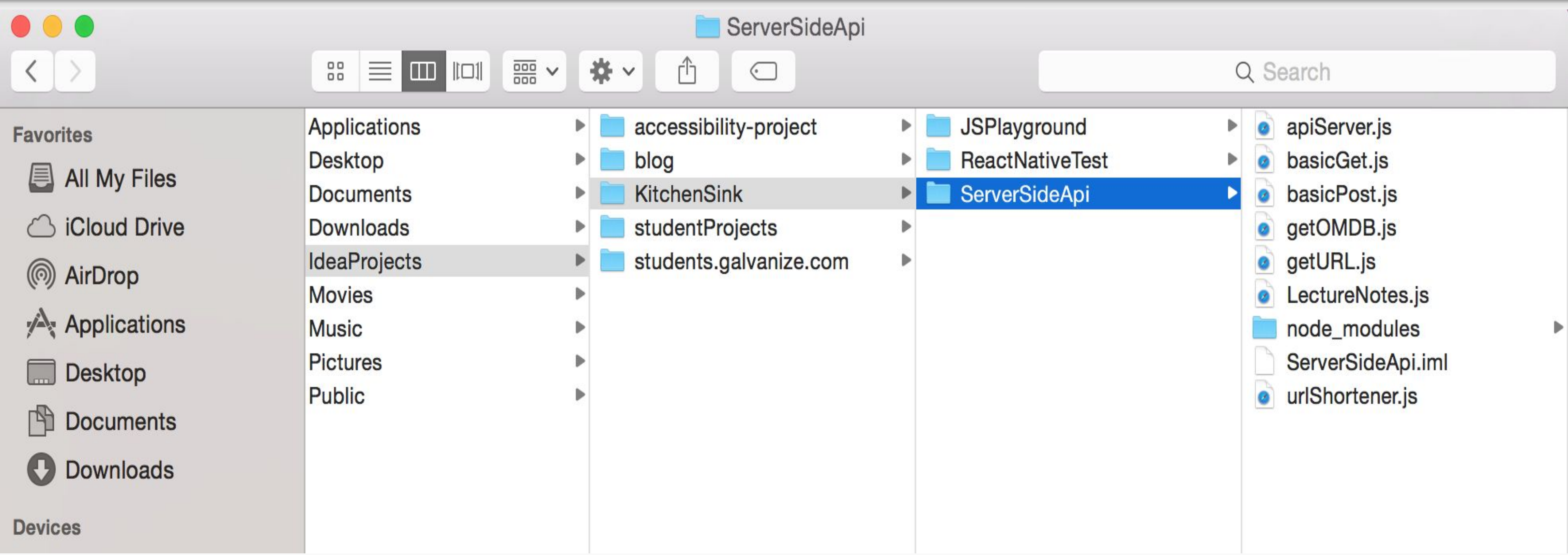
```
$ brew install tree
$ tree
$ tree -L 3
```

-L 3 means “only 3 *Levels deep”*



```
Tylers-MacBook-Pro:KitchenSink Tyler$ tree -L 3
.
├── JSPlayground
│   ├── JSPlayground.iml
│   ├── node_modules
│   │   ├── mongo
│   │   ├── mongoose
│   │   └── requirejs
│   └── tstMongoose.js
├── ReactNativeTest
│   ├── android
│   │   ├── app
│   │   ├── build.gradle
│   │   ├── gradle
│   │   ├── gradle.properties
│   │   ├── gradlew
│   │   ├── gradlew.bat
│   │   └── settings.gradle
│   ├── index.android.js
│   ├── index.ios.js
```

Finder can display your file system too



Applications
Desktop
Documents
Downloads
IdeaProjects
Movies
Music
Pictures
Public

accessibility-project
blog
KitchenSink
studentProjects
students.galvanize.com

JSPlayground
ReactNativeTest
ServerSideApi

apiServer.js
basicGet.js
basicPost.js
getOMDB.js
getURL.js
LectureNotes.js
node_modules
ServerSideApi.iml
urlShortener.js

Absolute and relative paths

- Absolute Paths
  - Any path that starts with /
  - Start at the root directory and work your way down the tree.
  - Your home directory has an absolute path of /Users/[yourAccountName]
- Relative paths
  - Any path that doesn’t start with a / is relative to your **current working directory**
- If you are in currently in home dir these two commands are equivalent:

```
$ cd folder1
$ cd /Users/AccountName/folder1
```

Current location (home)

Play around with rel/abs paths and aliases

- `$ cd ~`
- `$ ls ~/Documents # My Documents`
- `$ ls Documents`
- `$ ls /`
- `$ ls ../..`
- `$ ls ~/Documents/..`
- Try this tricky one:

```
$ ls /Users/`whoami`
```

Tab completion

- Your best friend
- Start typing a path or filename and press tab
- If there is only 1 option, tab completes the line with that option
- If there are multiple options double tapping tab displays all the competing options
- In your home directory try:
    - `$ ls D[TAB]`
    - `$ ls Do[TAB]`
    - `$ ls Doc[TAB]`

Practice time

Grab a buddy and practice:

Exercise: 3 minutes in pairs

Using Finder: Pick a directory somewhere under the /Users directory on your partner's computer

Your Task: Navigate to that directory in a single command from your home directory using a relative or absolute path

Help your partner if they are having trouble and use Tab Completion

echo

echo: prints whatever you tell it to print

```
$ echo “hello world”
$ echo whoami
$ echo `whoami`
$ echo $PATH
```

SURPRISE!

“$” invokes a variable. PATH is an environment variable. More on this in the sprint.

Output redirection

“>” redirects output to a file, overwriting what was there

```
$ echo “hello world” > newFile.txt
$ tail newFile.txt
$ echo "Look at how cool this text is" > newFile.txt
$ tail newFile.txt
```

“>>” redirects to a file, appending to whatever was there.

```
$ echo “I am loving this redirect thing” >> newFile.txt
$ tail newFile.txt
$ echo "I mean it is seriously legit" >> newFile.txt
$ tail newFile.txt
```

Pipe and grep

Pipes allow us to use the output from one command as the input for another

grep allows us to search through text.

```
$ ls -t | grep D

$ ls -t | grep D | sort

$ ls -t | grep D | sort > sortedItems.txt
```

File manipulation

- “mv” moves or renames a file
  - `$ mv file.txt Documents/` (move file.txt into the Documents folder)
  - `$ mv file.txt file2.txt` (rename file.txt to file2.txt)
- “cp” copies the file, like “mv” but leaves original untouched
  - `$ cp file.txt Documents/` (create file.txt copy in the Documents folder)
  - `$ cp file.txt file2.txt` (create file.txt copy called file2.txt in current folder)
  - `$ cp file.txt Documents/file2.txt`
    (create file.txt copy called file2.txt in Documents folder)

File manipulation

- “rm” removes a file (CAREFUL!!!! There is no undoing this!)
  - `$ rm file.txt`
  - `$ rm -r folder1/` (recursively remove folder 1 and all its contents **SUPERDUPER CAREFUL!!!!!!!!!!**)
- “mkdir” creates a new empty directory
  - `$ mkdir test_directory/`
- “rmdir” removes an empty directory (not the same as rm -r, which removes all contents of a non empty directory)
  - `$ rmdir test_directory/`

File inspection and editing

- “touch” creates an empty file
  - `$ touch testfile.txt`
  - `$ ls`
- Open and edit a file with Emacs, Vim or Nano on the commandline
  - `$ emacs/vi/nano filename.txt`
  - CAREFUL: If you don’t know the key bindings for the editor it can be hard to get out :)
- Similarly you can start a GUI editor on the commandline:
  - `$ emacs/atom/gedit filename.txt` (GUI emacs needs to be set up)
  - `$ open filename.txt` (Mac only)
- “head” and “tail” will show you the first or last few lines or a file
  - `$ head testfile.txt` (-n for number of lines, -c for number of chars)
  - `$ tail testfile.txt`

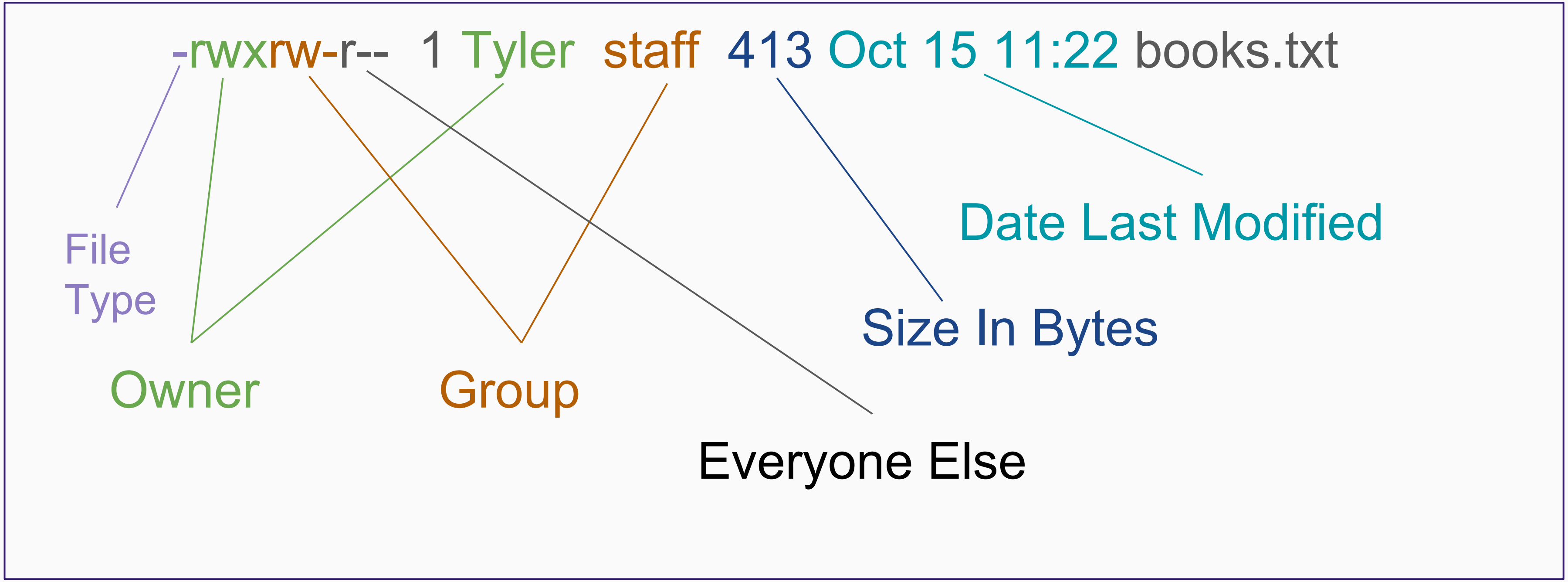
File inspection and editing

- “Less” and “more” will let you stream and search the file on the commandline
  - `$ less testfile.txt` (this is the **best option**!)
  - `$ more testfile.txt`
  - Use enter to scroll down
  - Use “/” to search for text
  - Use “q” to exit
  - Hint: “man” uses “less” to show the manual pages
- “cat” shows the ENTIRE file (careful!) (hint: more/less will work too)
  - Can be used together with piping:
  - `$ cat filename.txt | grep “href” | wc -l`
    (counts number of “href” - links - in a file)
  - `$ grep “href” filename.txt| wc -l` (will do the same without cat)

File execution

- Execute a python script:
  - `$ python script.py`
- Execute a shell script (use chmod to make executable first):
  - `$ chmod a+x script.sh`
  - `$ ./script.sh`

Permissions (type “ls -l”)



```
-rwxrw-r--  1 Tyler  staff  413 Oct 15 11:22 books.txt
```

File Type

Owner

Group

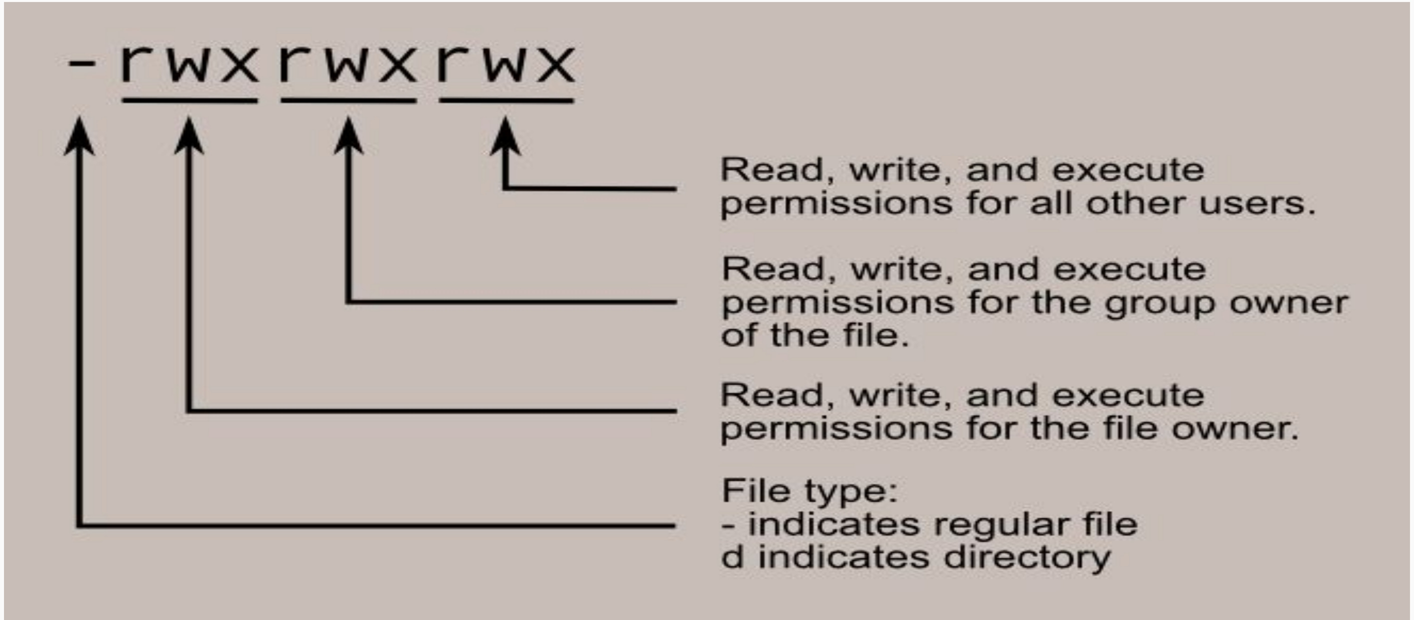
Everyone Else

Size In Bytes

Date Last Modified

Permissions (type “ls -l”)

```
-rwxrw-r--  1 Tyler  staff  413 Oct 15 11:22 books.txt
```



-rwxrwxrwx

Read, write, and execute permissions for all other users.

Read, write, and execute permissions for the group owner of the file.

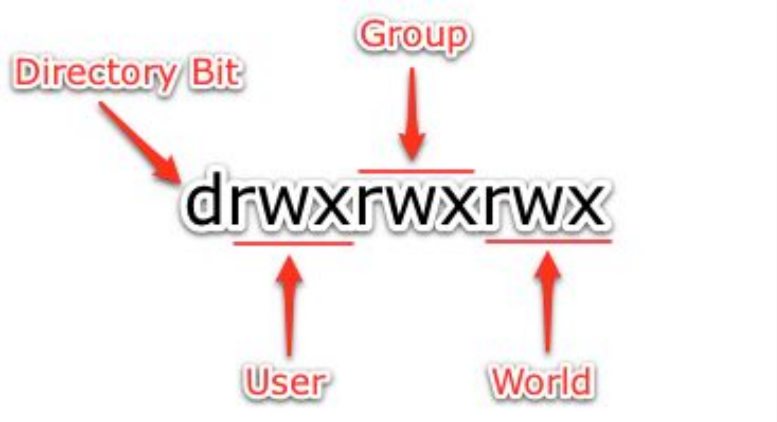
Read, write, and execute permissions for the file owner.

File type:
- indicates regular file
d indicates directory

Changing permissions (chmod)

```
$ chmod u/g/o/a +/- r/w/x [file]
```

- User (u)/ group(g)/ others(o)/ all three(a) will be given (+) or removed (-) reading (r) / writing (w) or execution (x) rights
- `$ chmod g+rw-x file.txt`
  gives group reading and writing right, removes execution right
- `chmod a+x file.txt`
  makes file an executable (for everyone)

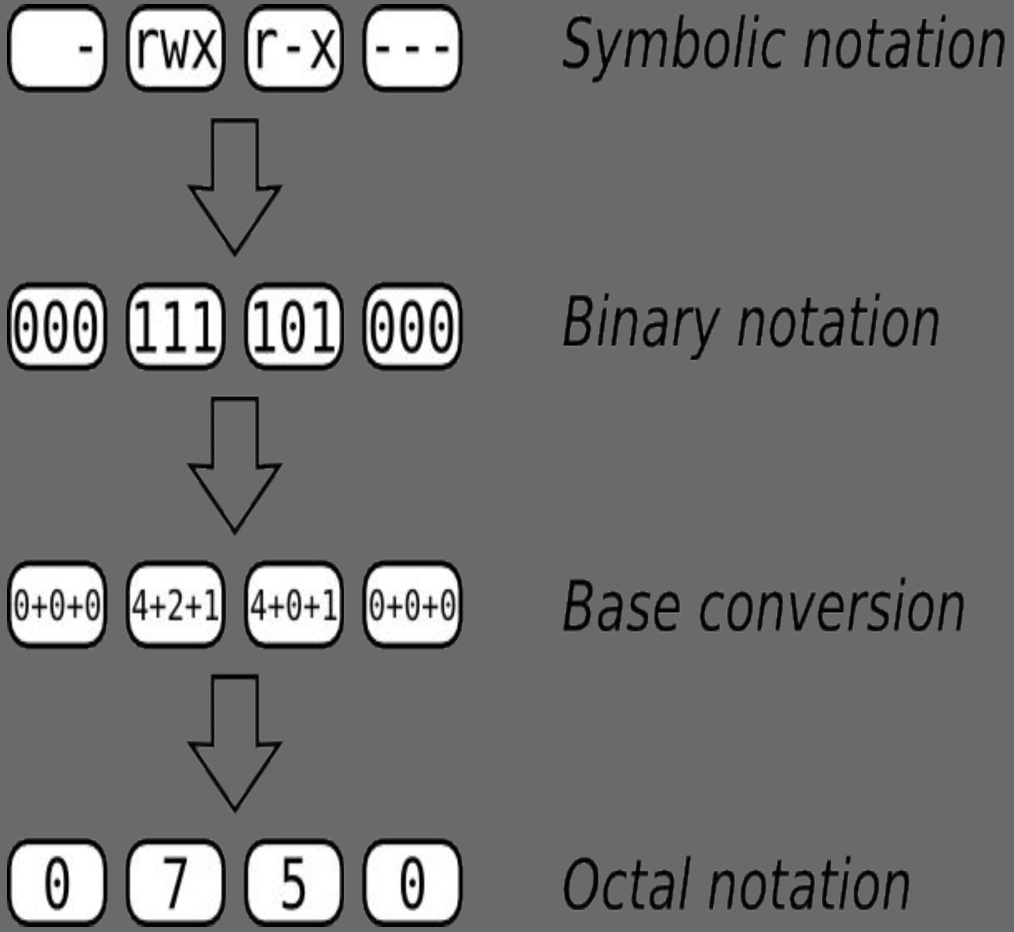


Changing permissions

- Chmod octal notation

    -
      ```
      $ chmod 750 file.txt
      ```
    - Sets permissions to -rwx r-x ---
- chown
    - change the ownership

    -
      ```
      $ chown user:group file.txt
      ```
- sudo [command]
    - Executes as root user/admin
    - root has ALL POWERS
    - sudo responsibly
- You will probably look this up a thousand times



- rwx r-x --- Symbolic notation

000 111 101 000 Binary notation

0+0+0 4+2+1 4+0+1 0+0+0 Base conversion

0 7 5 0 Octal notation

Command history

```
$ history
```

Using the “up” arrow will go back in history

```
$ history | grep ls
```

Gets all previous commands with “ls” in them

My favorite is reverse command search:

CTRL+r and then start typing parts of a previous command, keep hitting CTRL+r to go back through the results

Example of CLI power

Power and humor, a CLI example:

Website: bad.horse

```
$ traceroute bad.horse
```

(traceroute shows the path of a packet across the internet)

```
$ whois bad.horse
$ whois bad.horse | grep Billing
```

James Renken, Sandwich.net

→ https://www.linkedin.com/in/jrenken (let's hire this guy)

Resources

- Terminal Cheatsheet for Mac
- Explain Shell
- Command Line Crash Course Online Book
- Command Line Murder Mystery
- Google!

Objectives

- Explain what UNIX is
- Explain the difference between Bash and the terminal
- Use bash commands:
  - man
  - pwd, ls, cd
  - mv, rm, cp, mkdir
  - chmod
- Create, inspect or modify a file on the command line
  - emacs/vim/nano
  - more/less/head/tail
  - grep/wc
- Use command piping

Pair Programming Core Principles

- Get to know your partner: “What did you think about the lecture?”
- Take turns: trade driver and navigator roles every 30 minutes.
- Listen: if your partner asks “Why are we doing that?” then take the time to answer. Don’t interrupt.
- Be patient: if your partner types something that looks wrong, try to understand it before correcting it.
- Be clear: explaining technical concepts is hard. Practice.
- Be humble
- Disagree productively
- Switch partners daily