



Unix Workshop

STRI 28 July 2015
Paul B. Frandsen
OCIO/ORIS

Outline

1. Unix- what and why
2. Understanding the file systems
3. Basic commands for working with files
4. grep
5. Pipes and redirection
6. Compressing files



If you know the basics of this content, pull up the man page and dig deeper.

Unix

Unix

- A family of operating systems
- Created in the 1970s
- Multi-user, multi-task
- Very flexible: scriptable, extensible
- **Linux+MacOS+Solaris+... = most of the computing world**

Why learn Unix

- Long lived- tools learned here will be applicable for your career
- Just about every major technology is built around Unix (especially Linux)

Linux

- Developed in early 90s
- All open source- before there were licenses and fees



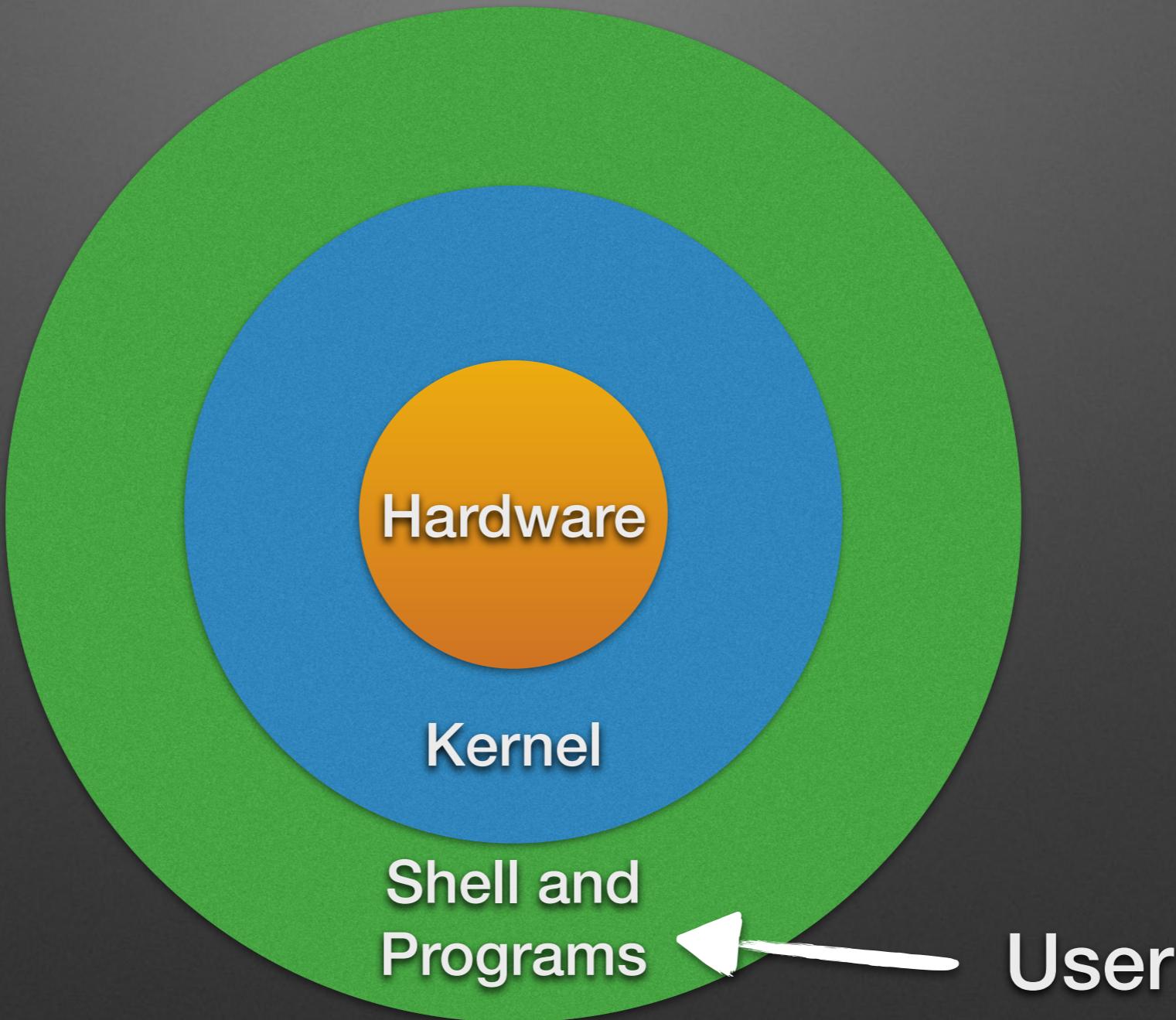
Kernel (communicates with hardware)
developed by Linus Torvalds

+

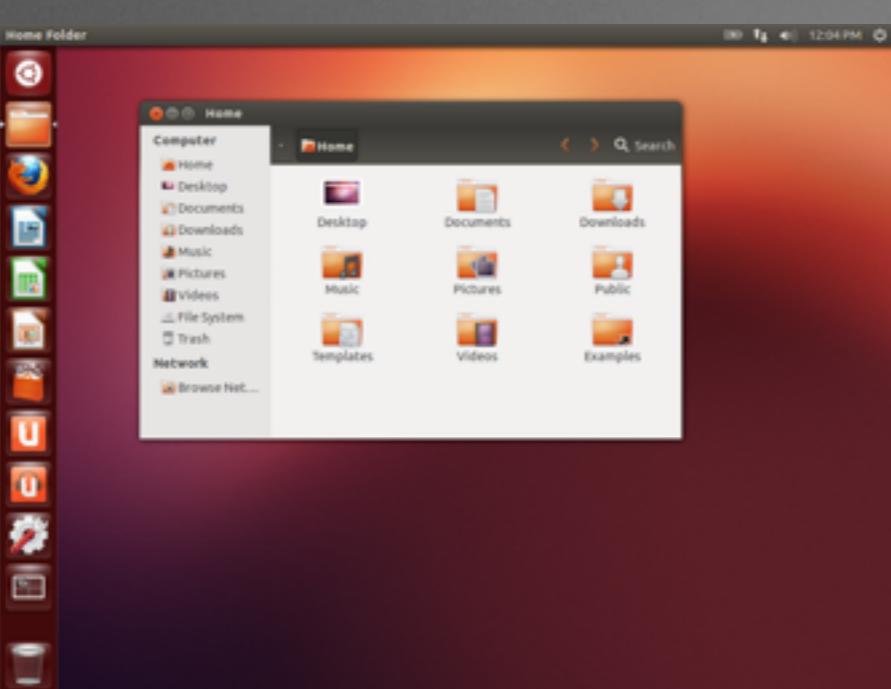
Shell and core programs
developed by Richard Stallman
(GNU = GNU's Not Unix)

- Hundreds of versions and branches (distributions)
 - Ubuntu, Red Hat, **CentOS (on Hydra)**, SUSE, Debian

Kernel + shell/programs



There can be a GUI



Or no GUI...





```
dave@nostromo:~  
drwx----- 6 dave dave 4096 Aug 31 22:56 .purple  
drwxrwxr-x 3 dave dave 4096 May 1 20:48 .pyrenamer  
drwx----- 4 dave dave 4096 Apr 9 22:40 .recoll  
drwx----- 2 dave dave 4096 Aug 19 09:40 .remmina  
-rw-rw-r-- 1 dave dave 66 Apr 10 21:51 .selected_editor  
drwxrwxr-x 4 dave dave 4096 Apr 9 21:11 .shotwell  
drwxrwxr-x 3 dave dave 4096 Aug 17 07:36 .shutter  
drwxrwxr-x 3 dave dave 4096 Jun 23 14:22 .subversion  
drwxrwxr-x 6 dave dave 4096 Jun 23 15:23 SVN  
drwxrwxr-x 3 dave dave 4096 Apr 29 15:13 .teamviewer  
drwxr-xr-x 2 dave dave 4096 Apr 9 16:12 Templates  
drwxr-xr-x 4 dave dave 4096 Aug 29 22:04 .themes  
drwx----- 4 dave dave 4096 Apr 9 16:28 .thumbnails  
drwx----- 4 dave dave 4096 Oct 22 2011 .thunderbird  
drwxrwxr-x 2 dave dave 4096 Jun 1 22:24 Ubuntu One  
drwx----- 2 dave dave 4096 Sep 1 22:57 VBShared  
drwxr-xr-x 149 dave dave 4096 Aug 24 20:57 Videos  
-rw----- 1 dave dave 707 Jul 20 22:26 .viminfo  
drwxrwxr-x 2 dave dave 4096 Sep 1 18:08 .VirtualBox  
drwx----- 5 dave dave 4096 Jul 1 17:41 VirtualBox VMs  
-rw----- 1 dave dave 53 Sep 2 09:24 .Xauthority  
-rw----- 1 dave dave 166801 Sep 2 17:11 .xsession-errors  
-rw----- 1 dave dave 1163071 Sep 1 23:49 .xsession-errors.old  
dave@nostromo:~$
```

```
top - 17:13:03 up 8:04, 4 users, load average: 0.11, 0.14, 0.22  
Tasks: 172 total, 3 running, 169 sleeping, 0 stopped, 0 zombie  
Cpu(s): 2.0%us, 1.8%sy, 0.0%ni, 95.8%id, 0.3%wa, 0.0%hi, 0.1%si, 0.0%st  
Mem: 4122428k total, 1879668k used, 2242760k free, 253144k buffers  
Swap: 4190204k total, 0k used, 4190204k free, 1062176k cached  


| PID   | USER | PR | NI | VIRT  | RES  | SHR  | S | %CPU | %MEM | TIME+    | COMMAND         |
|-------|------|----|----|-------|------|------|---|------|------|----------|-----------------|
| 1843  | dave | 20 | 0  | 291m  | 133m | 38m  | S | 7    | 3.3  | 14:53.01 | compiz          |
| 1246  | root | 20 | 0  | 71232 | 52m  | 12m  | R | 5    | 1.3  | 14:54.29 | Xorg            |
| 16709 | dave | 20 | 0  | 187m  | 47m  | 19m  | S | 3    | 1.2  | 0:01.86  | shutter         |
| 15911 | dave | 20 | 0  | 125m  | 19m  | 12m  | S | 1    | 0.5  | 0:03.00  | gnome-terminal  |
| 2073  | dave | 20 | 0  | 94568 | 5932 | 4540 | S | 1    | 0.1  | 3:16.70  | conky           |
| 1994  | dave | 20 | 0  | 41816 | 10m  | 8168 | S | 0    | 0.3  | 0:08.98  | gtk-window-deco |
| 16395 | dave | 20 | 0  | 2836  | 1160 | 860  | R | 0    | 0.0  | 0:00.46  | top             |
| 1     | root | 20 | 0  | 3516  | 1908 | 1248 | S | 0    | 0.0  | 0:00.80  | init            |
| 2     | root | 20 | 0  | 0     | 0    | 0    | S | 0    | 0.0  | 0:00.01  | kthreadd        |
| 3     | root | 20 | 0  | 0     | 0    | 0    | S | 0    | 0.0  | 0:00.94  | ksoftirqd/0     |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.57  | kworker/u:0     |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.00  | migration/0     |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.22  | watchdog/0      |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.00  | migration/1     |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.84  | ksoftirqd/1     |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.21  | watchdog/1      |
|       |      |    |    |       |      |      |   | 0    | 0.0  | 0:00.00  | migration/2     |


```

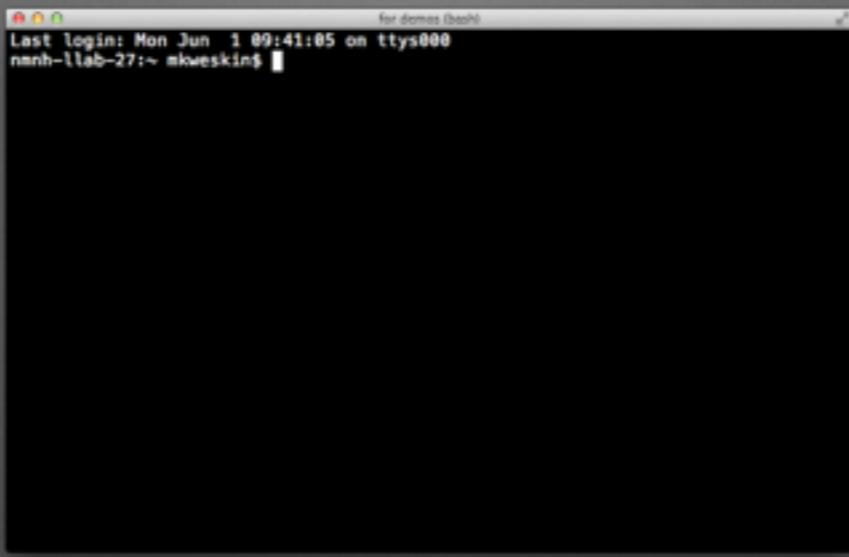
```
alias l='ls -CF'  
# Add an "alert" alias for long running commands.  
# sleep 10; alert  
alias alert='notify-send --urgency=low -i "$(echo $0 | sed -e "s/^/./; s/\//\./; s/://")" "$history|tail -n1|sed -e '\''$s/^/./; s/\//\./; s/://'\''"  
  
# Alias definitions.  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.  
  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then  
    . /etc/bash_completion  
fi  
dave@nostromo:~$
```



Why command line?

- Many bioinformatics programs can only be run from the command line
- Access
 - To computers via remote login
 - Multiple users can access at one time
- Advantages
 - Tight control on what is being done
 - Batch processing require scripts (commands)
- Shared resource
 - Not sole user of computer

Shell

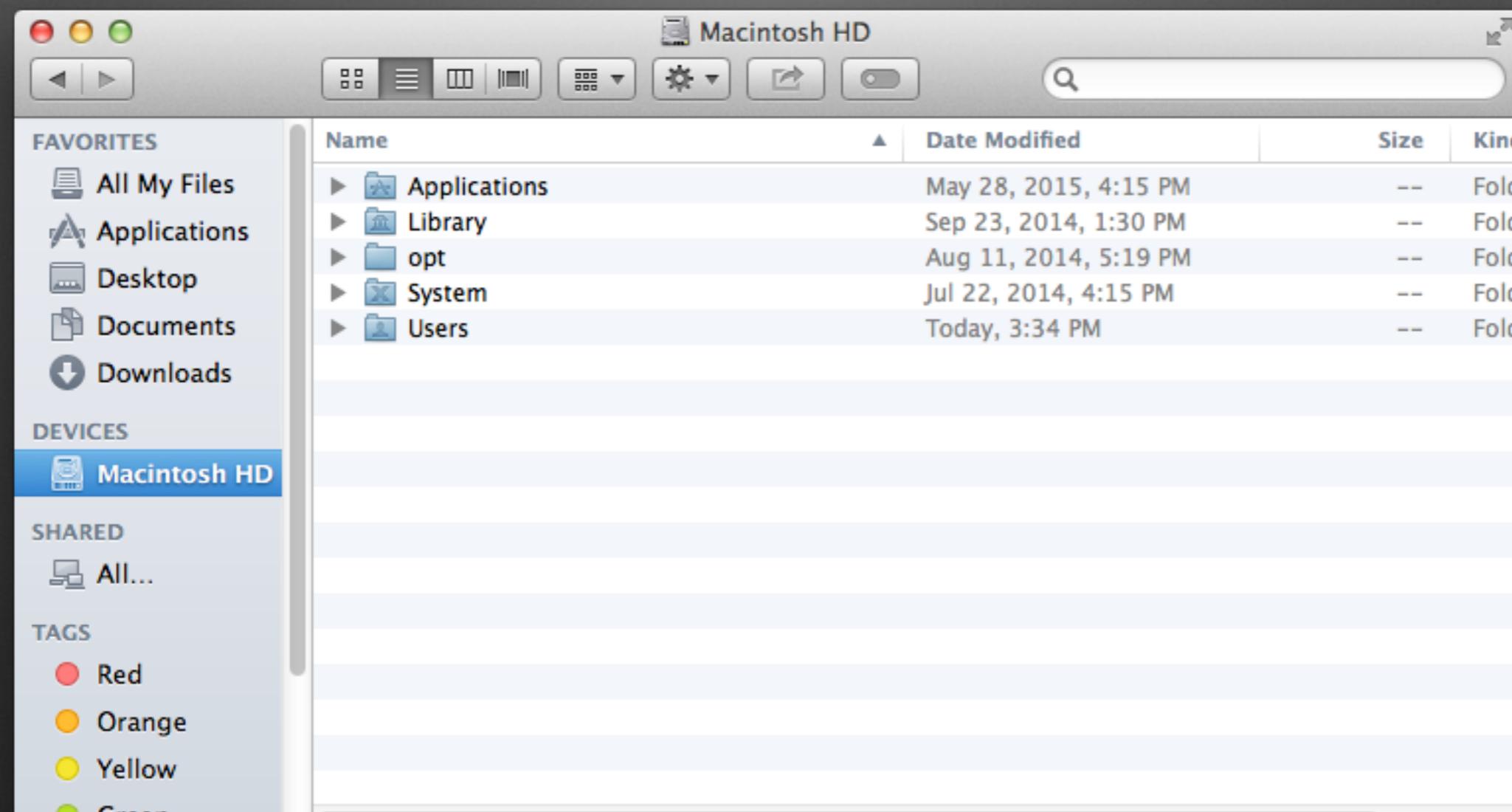


- Lingo for the program that is started when you login
- The command line interpreter- takes and executes your instructions
- Two major shells: bash (or sh) and csh (or tcsh)
 - bash being taught here and we're using as default for genomics users on Hydra
 - Some formatting differs between shells. You can try another shell

File structure

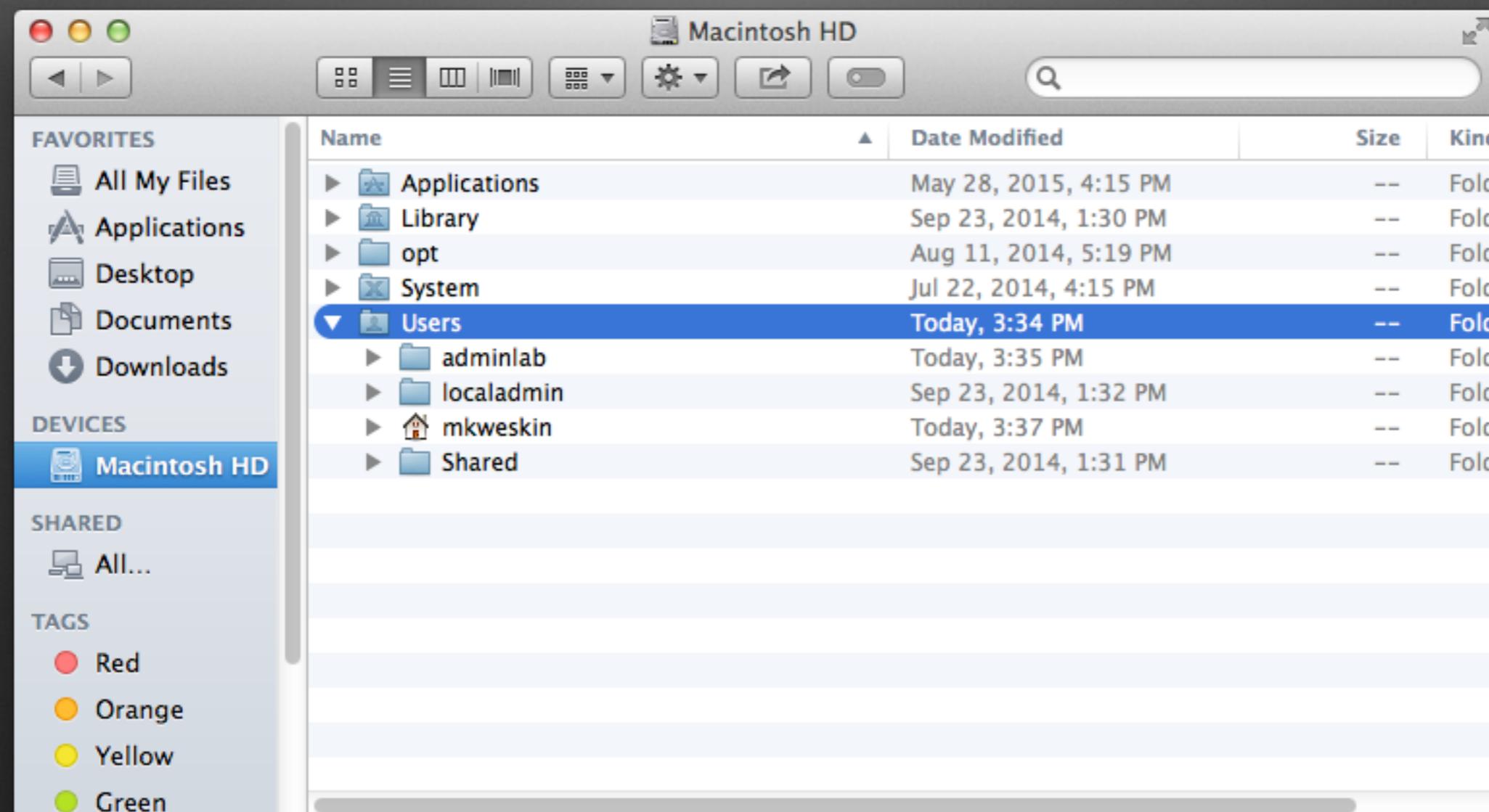
How things are organized on a disk

Identifying where things are Paths



/ “Root”: top most directory on a system, everything is inside it

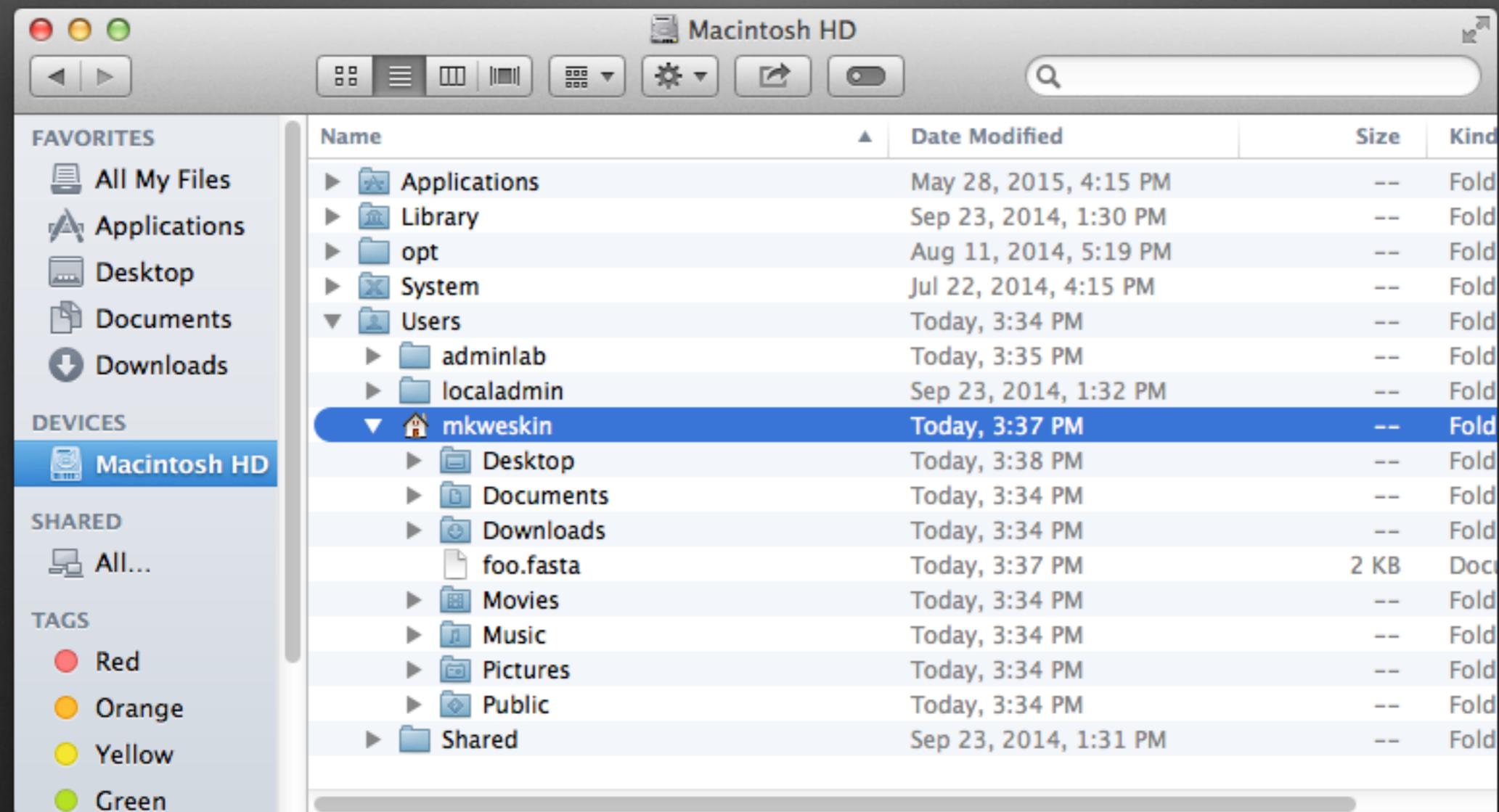
Identifying where things are Paths



/Users/

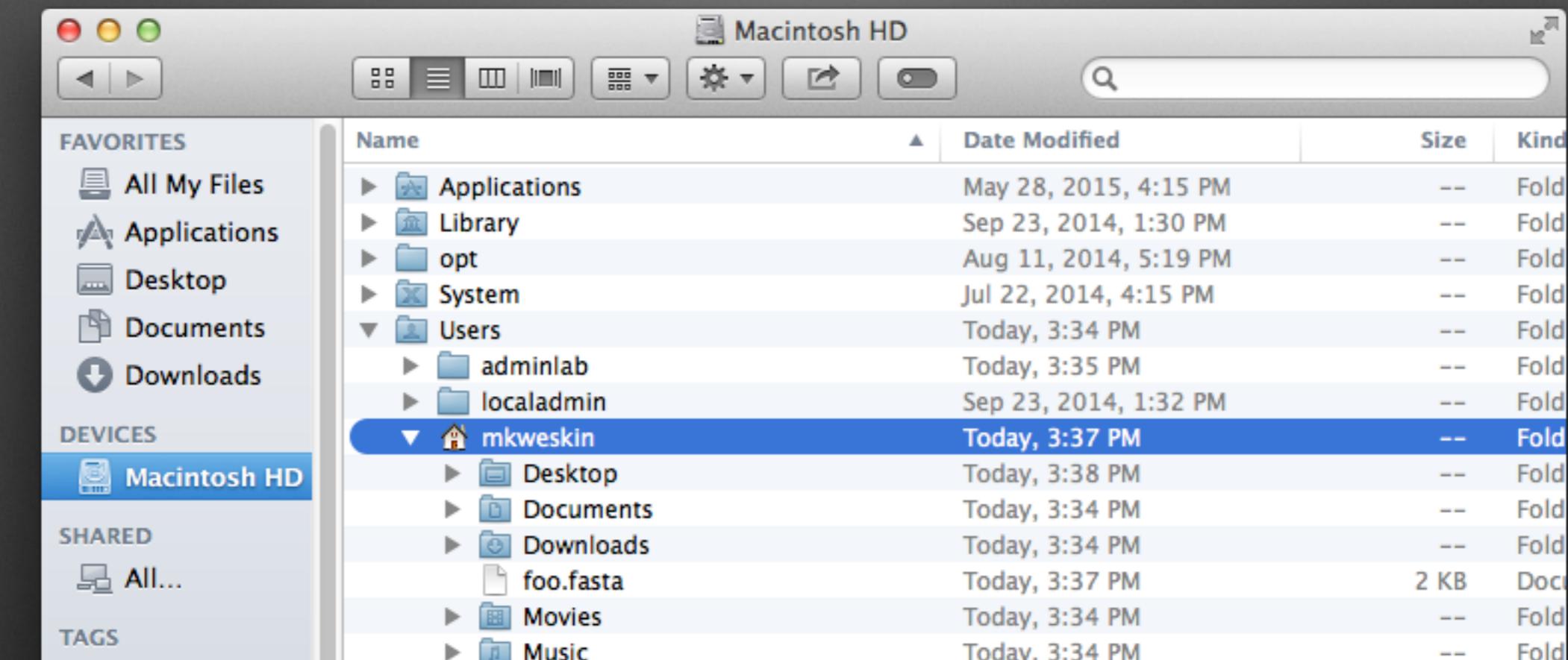
Ending / means a directory

Identifying where things are Paths



/Users/mkweskin/

Identifying where things are Paths

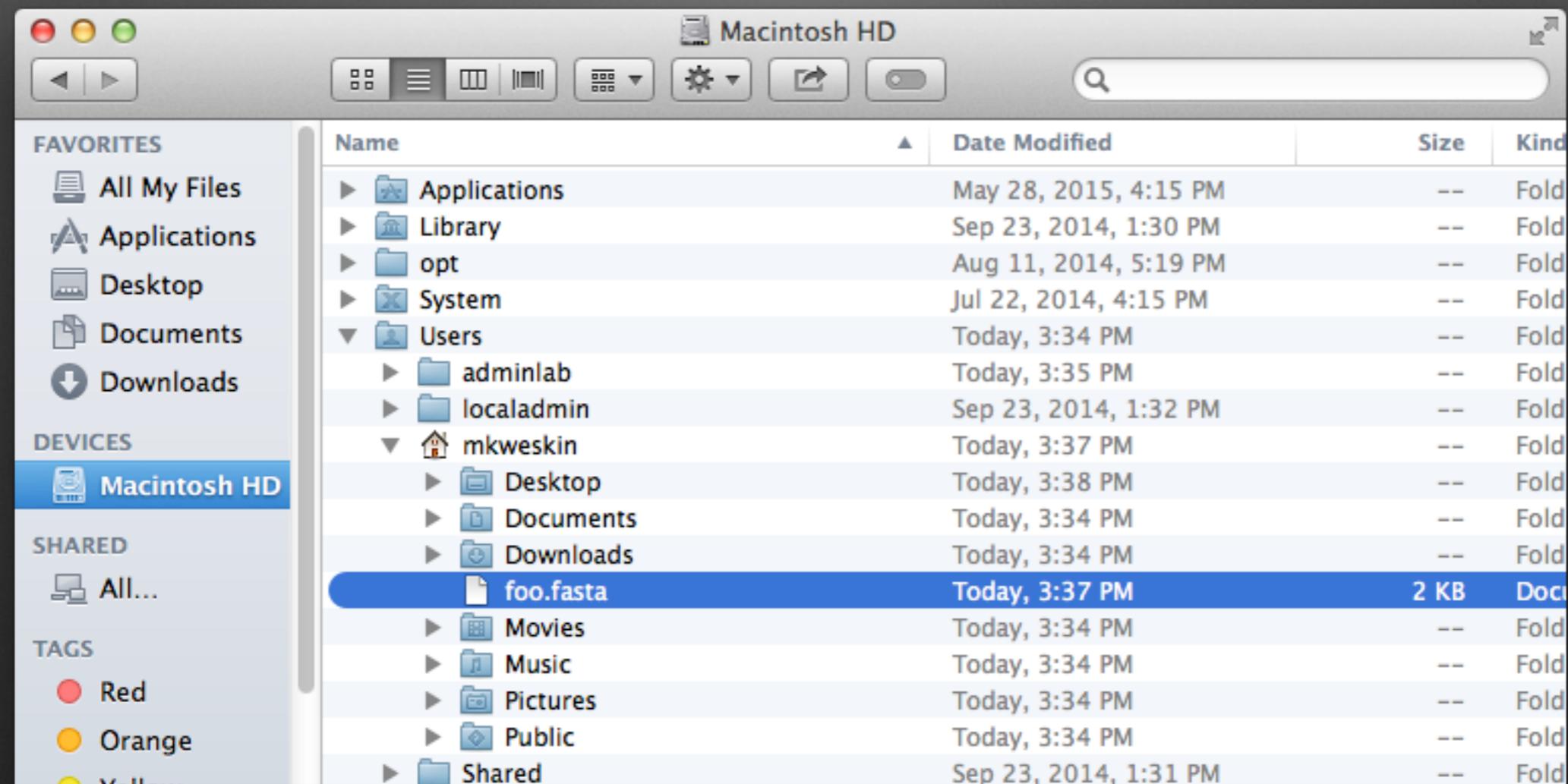


/Users/mkweskin

Or ~

(Mac home directory, linux is /home/user)

Identifying where things are Paths



/Users/mkweskin/foo.fasta
or ~/foo.fasta

Absolute vs relative paths

- Absolute file path: starts with a /
 - Complete path to the file. Works regardless of current directory.

```
/Users/mkveskin/foo.fasta
```

- Relative file path: starts with another directory or no directory
 - Depends on current directory

```
foo.fasta  
mkveskin/foo.fasta  
../localadmin/bar.fasta
```

Absolute vs relative paths

- Some analysis programs require full paths to input files. If the program documentation says full path, give it.

Special files/directories

- **. file** Files/directories starting with a **.** are hidden when you `ls`. Often used for config files, especially in your home directory.
- **.** Refers to the current directory
- **..** Refers to the parent directory

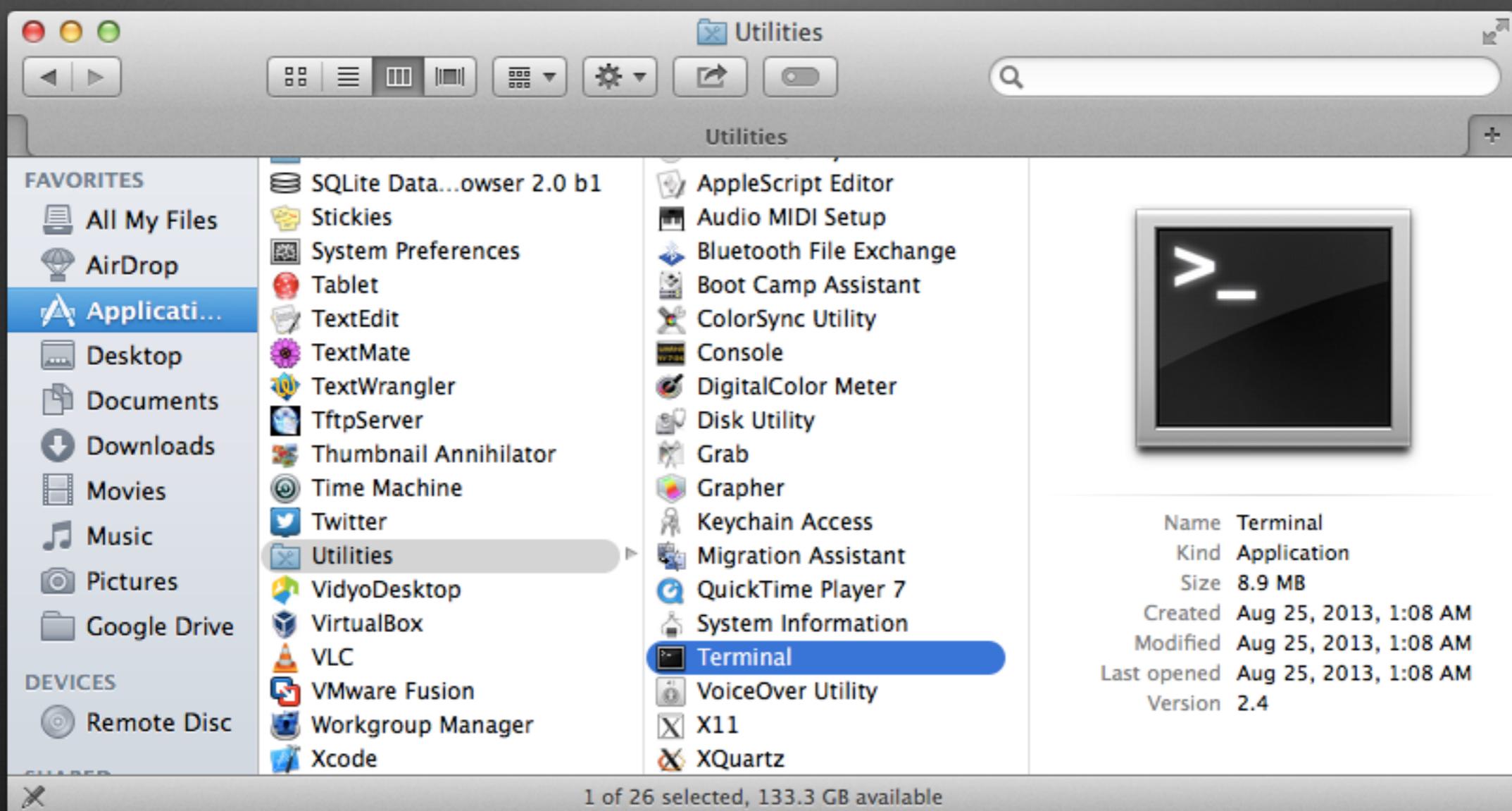
File/Directory Naming

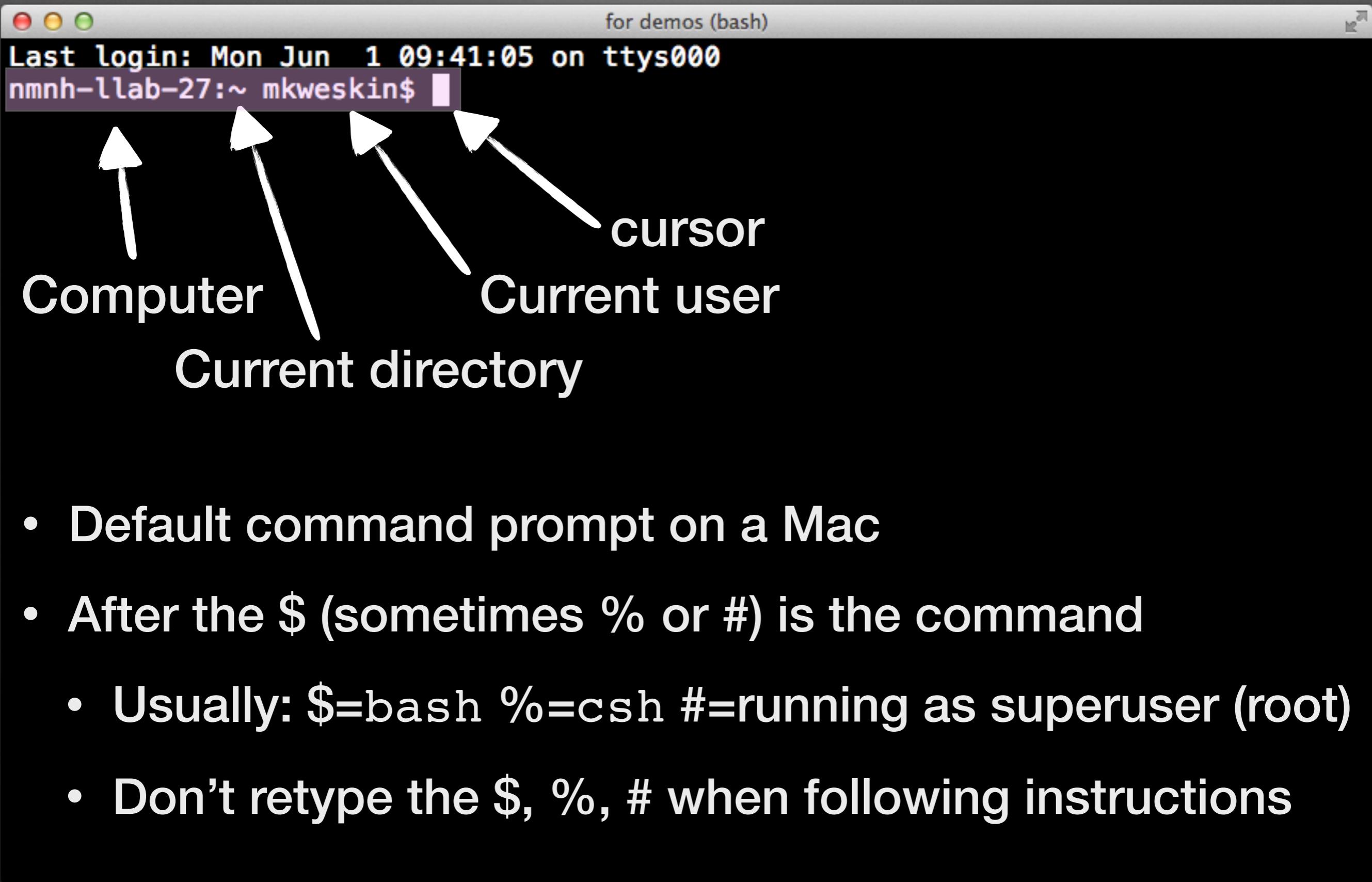
- Names are case sensitive
- There are no rules, just conventions
- Technically you can use just about weird character you want, but DON'T
- USE:
 - Letters numbers _ - +
- AVOID:
 - space < > & | ! () ? * ' " \ \$
 - non-printable chars, emoji
 - names starting with - or ~



The command prompt...
Let's run commands!

Starting the Terminal (Mac)





Current directory

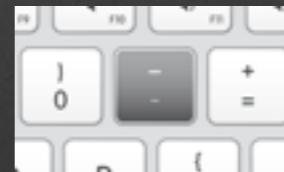
- You're always in a directory at the command prompt
 - Equivalent to window being show in the Mac Finder or Windows Explorer
 - Commands you run will affect files in this directory unless you specify another

Commands

```
command [ options ] arguments
```

```
$ ls -l ~/Downloads
```

- command: name of the command
- options (or flags or switches): additional program options
 - -x or --name (often: one hyphen for single letter, two for longer)
 - [] means optional
 - When sure it's a “regular” hyphen not — or —
- arguments (or parameter): file or data for program to use



Getting help

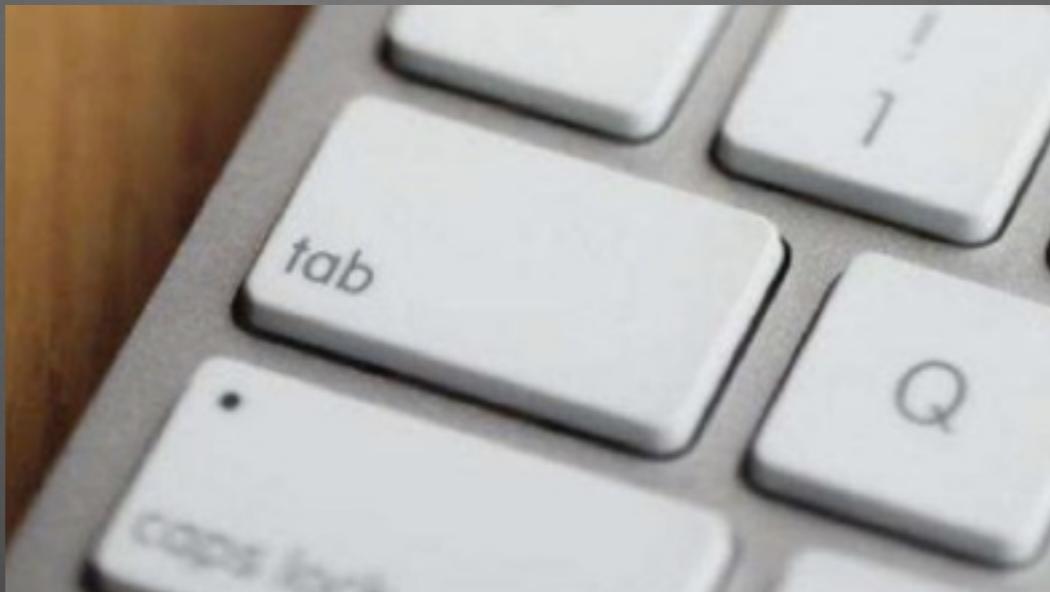
- **man** Manual page, reference instructions
 \$ man command

- Brief usage screen (varies by command)

```
$ command -h  
$ command --help  
$ command --help  
$ command
```

- Google! Try: linux *command*

Tab completion (will change your life)



- Try it by starting to type a command or file name and then press **tab**
- Complete a file or command name until there is an ambiguity
- Tab twice to show all options available (in bash shell)

For Demos (bash)

```
MattAir:tabdemo mkweskin$ ls
contigs.fa      read02.fa
read01.fa      readme
MattAir:tabdemo mkweskin$ █
```

For Demos (bash)

```
MattAir:tabdemo mkweskin$ ls
contigs.fa      read02.fa
read01.fa      readme
MattAir:tabdemo mkweskin$ less r█ tab
```



For Demos (bash)



```
MattAir:tabdemo mkweskin$ ls  
contigs.fa      read02.fa  
read01.fa      readme
```

```
MattAir:tabdemo mkweskin$ less read
```

tab

tab



For Demos (bash)



```
MattAir:tabdemo mkweskin$ ls  
contigs.fa      read02.fa  
read01.fa      readme  
MattAir:tabdemo mkweskin$ less read  
read01.fa  read02.fa  readme  
MattAir:tabdemo mkweskin$ less read
```



For Demos (bash)



```
MattAir:tabdemo mkweskin$ ls  
contigs.fa      read02.fa  
read01.fa      readme  
MattAir:tabdemo mkweskin$ less read  
read01.fa  read02.fa  readme  
MattAir:tabdemo mkweskin$ less readm█ tab
```



For Demos (bash)



```
MattAir:tabdemo mkweskin$ ls  
contigs.fa      read02.fa  
read01.fa      readme  
MattAir:tabdemo mkweskin$ less read  
read01.fa  read02.fa  readme  
MattAir:tabdemo mkweskin$ less readme █
```



space added automatically
Indicates tab completion is done

Wildcards (globbing in Unix speak)

- * any string (length ≥ 0) of characters

```
$ ls -l *.txt
```

- ? any one character

```
$ ls -l read0?.fasta
```

- [abc] any one character within the brackets

```
$ ls -l read[ 01 ].fasta
```

Working with files

ls = list directory contents

```
$ ls
Applications    Library
Desktop          Movies
Documents        Music
Downloads        Pictures
Geneious 7.1    Public
```

Common options:

- **-l** long listing: gives file type size and other info
- **-a** list all files including hidden files starting with “.”
- **-t** sorted by time last modified, newest at top
- **-tr** sorted by time last modified, reverse sorted newest at bottom
- **Argument: Directory(s) or file(s) to list**

cd = change directory

```
$ cd Downloads  
$ cd ~/Downloads  
$ cd ..
```

- Argument
- Directory to move to. If you give a file instead you will get an error like this:

```
-bash: cd: foo.fasta: Not a directory
```

mkdir = make directory

```
$ mkdir mydir  
$ mkdir ../mydir
```

- Arguments
 - Directory(s) to create

rmdir = remove directory

```
$ rmdir mydir  
$ rmdir ../mydir
```

- Can only delete empty directory (use `rm -r` if full)

```
rmdir: mydir: Directory not empty
```

- Arguments
 - Directory(s) to create

cp = copy file

```
$ cp foo.fasta bar.fasta  
$ cp -r mydir ~/Downloads  
$ cp ../foo.fasta .
```

- Common options:
 - **-r recursive copy, include directories.** Otherwise error like:
`cp: omitting directory `mydir'`
 - **cp will overwrite current files!**
 - **-i (interactive, prompts before overwrite)** 
- Arguments
 - files/directories to copy, Last argument is destination

mv = move file

```
$ mv foo.fasta bar.fasta  
$ mv mydir ~/Downloads
```

- **mv will overwrite current files!**
- **-i (interactive)** 
- Arguments
 - file/directory to move
 - Last argument is destination

rm = remove file

```
$ rm foo.fasta  
$ rm -r mydir
```

- Common options:

- -r recursive, include removing directories. Otherwise error:
`rm: cannot remove `mydir': Is a directory`

- There's no undo, so check twice before using rm



- Be careful using wildcards. I never use rm *



- Arguments

- files/directories to remove (can be multiple)

pwd = print working directory

- Shows the full path of your current directory

```
$ pwd  
/pool/cluster0/workshop
```

Stuck in a command

- <control+c> Will usually kill current program, sometimes <control+d>
- If that doesn't work (emacs, vi, less) Google: quit vi or <control+z> to suspend program (but then you have to kill it)

Viewing text files

Viewing full files

cat and less

- Show FULL contents (or concatenate several files)

```
$ cat file
```

- Show file one screen at a time (one command, two names)

```
$ less file
```

```
$ more file
```



*****The Tragedie of Hamlet*****

The Tragedie of Hamlet

Actus Primus. Scoena Prima.

Enter Barnardo and Francisco two Centinels.

Barnardo. Who's there?

Fran. Nay answer me: Stand & vnfold
your selfe

Bar. Long liue the King

Fran. Barnardo?

Bar. He

Fran. You come most carefully vpon your houre

Bar. 'Tis now strook twelue, get thee to bed Francisco

Fran. For this releefe much thankes: 'Tis bitter cold,
And I am sicke at heart

:|

less = interactive file display to screen

```
$ less foo.fasta
```

- Single key commands when running:
 - q quit
 - <space> forward one screen
 - <arrow up> <arrow down> move up/down one line
 - /word search for “word” in document (case sensitive)
 - -i to make case insensitive
 - G bottom of document
 - gg top of document

Viewing files- head and tail

- First 10 lines of file `$ head file`
 - `-<number>` to specify number of lines to show
 - e.g. `$head -40 myfile`
- Last 10 lines of file `$ tail file`
 - `-<number>` to specify number of lines to show
 - e.g. `$tail -10 myfile` or `$tail -10 job*.out`

Editing text files

Editing/creating text files

- You can edit and create files directly from the command line (on the remote system)
- nano easiest to get started with
- vi “pro” text editor, also see emacs
- emacs “pro” text editor, also see vi

nano = play on pico

```
$ nano foo.fasta  
$ nano newfile.fasta
```

- Can edit a file or create a new file
- Arrow keys to navigate
- Use <control+letter> options at bottom (^=control-key)
 - ^x = <control+x> exits
 - Will prompt to save changes “Save modified buffer”, type y to save and then return to use the same name



bio_user@hydra-3:~/Day1/data (ssh)



GNU nano 2.0.9

File: ecoli_ref-5m_small.fasta

```
>EAS20_8_6_1_3_1486/1
GGGGTAACGGCTCACCTAGGCGACGATCCCTAGCTGGTCTGAGAGGGATGACCAGCCACACTGGAACTGAGACACGG$
>EAS20_8_6_1_3_1671/1
GCGGTGTTGGCGAAATAAGCGAAAACGAGGAGATAAACAAATGAGTCAAACCATAACCCAGAGCCGTTACGCATTG$
>EAS20_8_6_1_3_300/1
GTTGATTGCCAATAACCTTAAGCATTGATTACGGAATCGTAAAATGAAGCAGTTCTTGATTTTTACCGCTGG$
>EAS20_8_6_1_3_1543/1
TATCTTAAAAAGCTGGCGTTGAGTACAAGATTGTTGAAGAGAATACTTACGGTATCGTGAAAGAGAAGATTCCAG$
>EAS20_8_6_1_3_224/1
TTATGGGTCGCGATTGCCGACGTCAGCTACTATGTGCGTCCGTCAACGCCGCTGGACAGAGAAGCGCGTAACCGTG$
>EAS20_8_6_1_3_238/1
TGACTATAATAATAGACGTTCTTGTTCCTTGCTTGTAAATACGCCACAAACTCTGTAAATTAAATGTATGAATA$
>EAS20_8_6_1_3_95/1
GCGTGTTCCTGGTTGCACGCCGGCGGTGATGTCATCTTCCGAATCACCAGCAAATACCGGGACTTCGCCTGTA$
>EAS20_8_6_1_3_1184/1
CCATCAGCTTGTCTTCGCTGCCGTTCGCACAAACCAAGTCGCCAAAGATTGAAATAGCTCGGCAAACGCC$
```

```
>EAS20_8_6_1_3_900/1
```

```
CGCGCGGTATTGATGTCGGGGCGAAGTTGAAATTATCAGTTAATTGCCGAACTGGCGAAGAAAGGCAAGGGGAT$
```

```
>EAS20_8_6_1_3_677/1
```

[Read 200000 lines]

^G Get Help **^O WriteOut** **^R Read File** **^Y Prev Page** **^K Cut Text** **^C Cur Pos**
^X Exit **^J Justify** **^W Where Is** **^V Next Page** **^U UnCut Tex** **^T To Spell**

echo = display text to output

```
$ echo "Hello world"  
Hello world
```

- Display text to screen
- Why?
 - useful in scripting
 - Display variables (\$PATH)

which = which directory is this command found?

```
$ which mkdir  
/bin/mkdir  
  
$ which java  
/usr/java/latest/bin/java
```

- Shows location of the command being used to make sure it's the right version

Hands-on #1-4

grep: global regular expression print



🐢 grep searches input files for a search string and prints the lines that match it

grep reads through your file from the beginning,
copies a line into a buffer,
compares it against the search string,
and if the comparison passes,
prints the line to the screen

grep example

Redirecting and Pipes





Redirecting

- > (redirect operator) creates a file
- >> (append operator) appends to the file
- >& redirects stderr and stdout to a file

great for creating log files

```
/apps/allpaths-lg-old/bin/RunAllPathsLG PRE=/pool/genomics  
DATA_SUBDIR=M_zebra REFERENCE_NAME=fish RUN=run1 THREADS=24  
OVERWRITE=True > fish_5_19.log
```



Using pipes



- 🐢 A pipe is a form of redirection that sends the output of one program to another program for further processing
- 🐢 Redirection is the transferring of standard output to some other destination, such as another program, a file or a printer, instead of stdout
- 🐢 Pipes create temporary direct connections between two or more simple programs

```
command_1 | command_2 [ | command_3 . . . ]
```



Using pipes

simple example: pipe the standard output of one program to the standard input of another program

```
ls -l | grep 'rwx'
```

you will see all of the files in the working directory whose permissions (or name) contain the letters rwx in order

- 🐢 ‘ls’ lists files to its standard output
- 🐢 ‘grep’ takes its input and sends any lines that match a particular pattern to its standard output
- 🐢 the pipe operator (|), which tells the shell to connect the standard output of ls to the standard input of grep



tar and gzip



archive a file system:

- [-c] create new archive
- [-x] extract to disk
- [-z] zip/unzip with gzip
- [-v] verbose
- [-f] file

```
$ tar -cvf hello.tar hello.txt hello.md hello.py  
result: hello.tar
```



compress and replace with a .gz file

```
$ gzip hello.tar  
result: hello.tar.gz
```



Do it all in one command

```
$ tar -cvzf hello.tar.gz hello.txt hello.md hello.py
```



tar and gzip

🐢 gunzip **uncompress a .gz file**

```
$ gunzip hello.tar.gz  
result: hello.tar
```

🐢 tar **unpack a tarball**

```
$ tar -xvf hello.tar  
result: hello.txt, hello.md, hello.py
```

🐢 **Do it all in one command**

```
$ tar -xvzf hello.tar.gz
```