Brief Unix tutorial

What's UNIX/Linux?

- An operating system run on many servers/ workstations
- Invented by AT&T Bell Labs in late 60's
- Command line system
- Modular

- Currently there are different versions and variants of UNIX such as SunOS, Linux, Solaris, BSD, Mac OS X (sorta), Android (sorta) …

Introduction: Why Unix/Linux?

- Linux is free
- It's fully customizable
- It's stable (i.e. it almost never crashes)

- These characteristics make it an ideal OS for programmers and scientists

Basic Linux

- Most functionality is accessed through the prompt.
  - Text based, like MS-DOS

- Mastery of Linux comes from being familiar with different commands
  - We will cover some of the basic commands in this class
  - You will gain mastery through TIME and PRACTICE

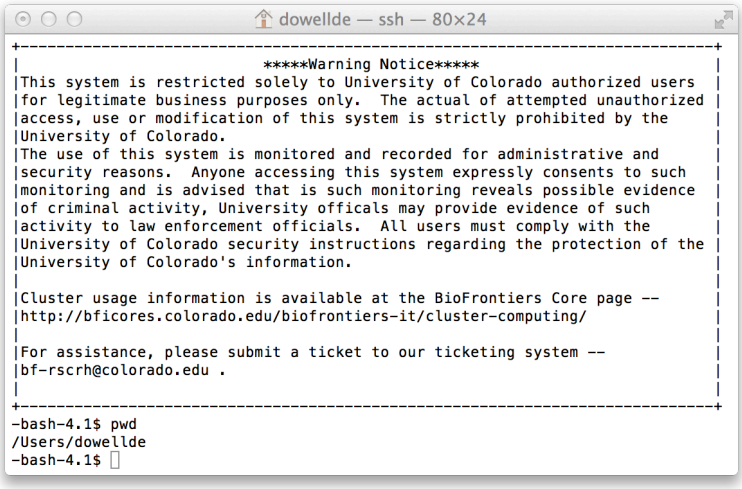
Recommended: If you are unfamiliar with Linux, work through (one of may) an online tutorial:
http://www.ee.surrey.ac.uk/Teaching/Unix/

Today we'll quickly go over the basics.

- **Getting help while on the system**
- **The shell**
- **Working with files & directories**
- **Wild card characters**
- **Security**
- **I/O redirection**
- **pipes**
- **process and job control commands (next week)**

Connecting to a Unix/Linux system

Open up a terminal:



Logging in

- To login
  ssh <identikey>@fiji.colorado.edu
- To logout
  logout or **exit**

Note: all Unix commands are case sensitive

What exactly is a "shell"?

- After logging in, Linux/Unix starts another program called the **shell**
- The shell interprets commands the user types and manages their execution
  - The shell communicates with the internal part of the operating system called the **kernel**
  - The most popular shells are: tcsh, csh, korn, and bash
  - The differences are most times subtle
  - For this tutorial, we are using bash
- Shell commands are **CASE SENSITIVE!**

General command format

Command -options arguments

- options/flags generally identify some optional capabilities
- some parts of a command are optional. These are indicated in the man pages with **[ ]**
- case sensitive

Getting Help from the System

- All Unix commands are described online in a collection of files called "man pages"
  man command

Note that the man pages uses an ancient method of viewing content:

- <spacebar> to go down a page
- Arrow keys *should* work to review already seen segments.
- <q> to quit the man pages

- For help on some topic
  man -k *keyword*
- For more information on using the man pages
  man man

Files and Directories

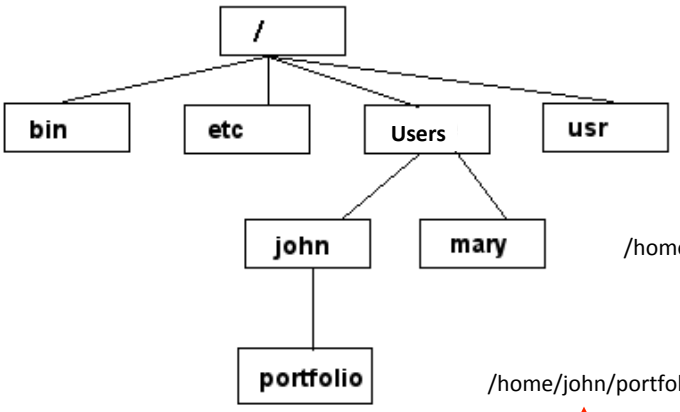
- Home directory:
  - The actual path of your home directory may be something like:

    /Users/identikey

  - **Note the forward slashes**

Unix/Linux File System

NOTE: Unix file names are **CASE SENSITIVE!**



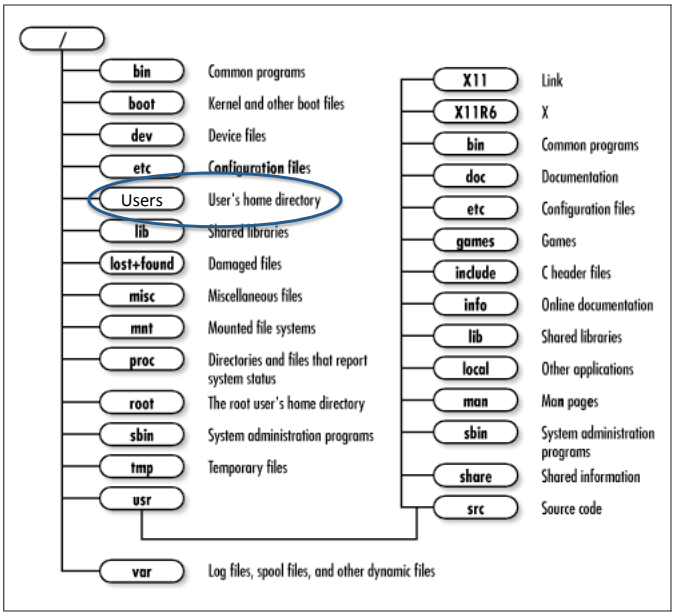
/home/mary/

/home/john/portfolio/

The Path

Linux Directory Structure

- Files are grouped in the directory structure. The file-system is arranged like hierarchical tree structure.
- The top of the tree is called "root" which usually contains several sub-directories.
  - "/"(forward slash) is used to present the "root".



Pathnames

- **pwd (print working directory)**

```
-bash-4.1$ pwd
/Users/dowellde
```

Listing contents of a directory

Ls (list files and directories)

ls

– The ls command lists the contents of your current working directory.

Listing contents of a directory

- To generate a detailed listing

```
ls  -l
```

- To display all files (including hidden)

```
ls -a
```

- To make information "human readable"

```
ls  -h
```

Command: ls

- ls has many options
  - -l long list (displays lots of info)
  - -t sort by modification time
  - -S sort by size
  - -h list file sizes in human readable format
  - -r reverse the order
- "man ls" for more options
- Options can be combined: "ls -alhr"

Listing contents of a directory

- To generate listing of a specific directory

```
ls  -alh  pathname
```

where **pathname** is the path of intended directory.

pathname may be full path or relative path.

"." : points to the current directory
".." : points to the parent directory of the current working directory
"~" : points to your home directory

Pathnames

- Absolute Pathnames
  $ /Users/dowellde/4521/FASTQ/file1

- Relative pathnames
  - If you are already in the /Users/dowellde directory, the relative pathname for file1 is:
    4521/FASTQ/file1

Wildcards

- **~:** a tilde at the beginning of a word expands to the name of your home directory.

e.g: `ls ~`

`cat ~/proj1.cc`

- if you append **~** to a user name, it refers to that user's home directory.

e.g: `ls ~smith`

lists all files in home directory of user smith

Wildcards

- The characters `*` will match against one or more characters in a file or directory name.

  `ls proj*`

- The character `?` Will match against any single character
- `[ ]` **:** the brackets enclose a set of characters any one of which may match a single character in that position.

  e.g `cat proj[125]`

  `cat proj[1-7]`

Some tips

- "tab" is used for auto-complete.
  - If a file/directory name was partly typed in, tab will auto-complete it.
  - If there are multiple options, tab will auto-complete up to the point where the options branch and show you a list of possible options
- "*" is used as a wild card.
  - "rm *blah**" removes all files which start with *blah*, so blah1, blah2, and blahblah would all be removed
  - Using "cp public/* private/" copies all files in your public directory into your private directory, and keeps all file names intact.

Making Directories

mkdir (make directory)

```
mkdir    name
```

- creates a subdirectory in current working directory

```
mkdir    somepath/name
```

- creates a subdirectory in directory **somepath**

Changing to a different directory

cd (change directory)

```
cd   pathname
```

- change current working directory to **`pathname.`**

- **`cd`** by itself will make your home directory the current working directory.
- **`cd  ..`** : cd to parent of current directory
- **`cd  ~`** : cd to home directory

Copying files

cp (copy)

```
cp file1  file2
```

- makes a copy of file1 and calls it file2. File1 and file2 are both in current working directory.

```
cp pathname1/file1 pathname2
```

- copies file1 to pathname2

e.g **cp ~/tutorial/science.txt .**

Moving files

mv (move)

```
mv  file1  file2
```

- moves (or renames) file1 to file2
- use the -i option to prevent an existing file from being destroyed

```
mv  -i file1  file2
```

- if **file2** already exist, **mv** will ask if you really want to overwrite it.

Removing files and directories

rm (remove)

```
rm  file1 [file2]…
```

- Use the -i option for interactive remove:

```
rm -i proj*.*
```

Be very careful, deletions are permanent in Unix/Linux

Removing files and directories

rmdir (remove directory)

```
rmdir path
```

- will not remove your current working directory
- will not remove a directory that is not empty
- To remove a directory and any files and subdirectories it contains use **-r** (recursively)

```
rmdir -r path
rmdir -ir  path
```

Displaying the contents of a file on the screen

cat (concatenate)

```
cat myfile
```

displays the contents of myfile on monitor

```
cat file1 file2 file3
```

more (less)

displays a file on the screen one page at a time. Use space bar to display to next page.

Head -- displays first 10 lines

tail -- displays last 10 lines

A brief segue:
Sequence Files
(fasta and fastq)

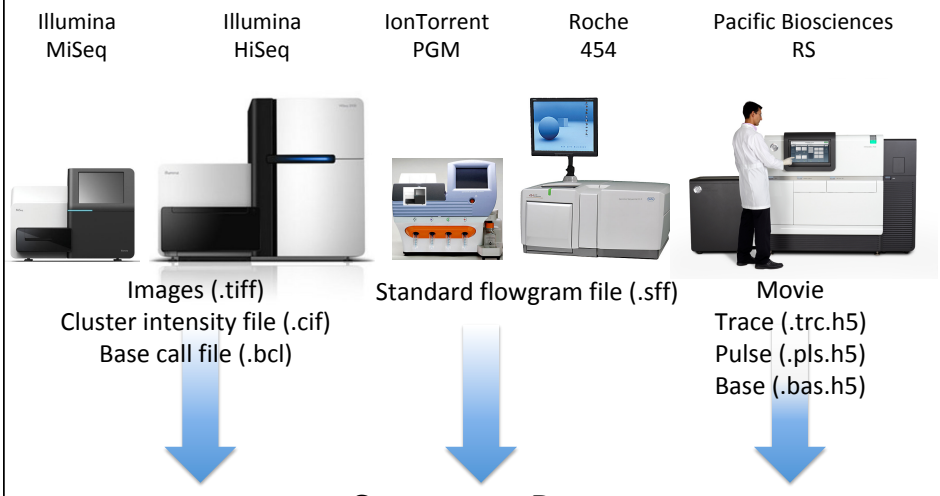
Simple fasta format

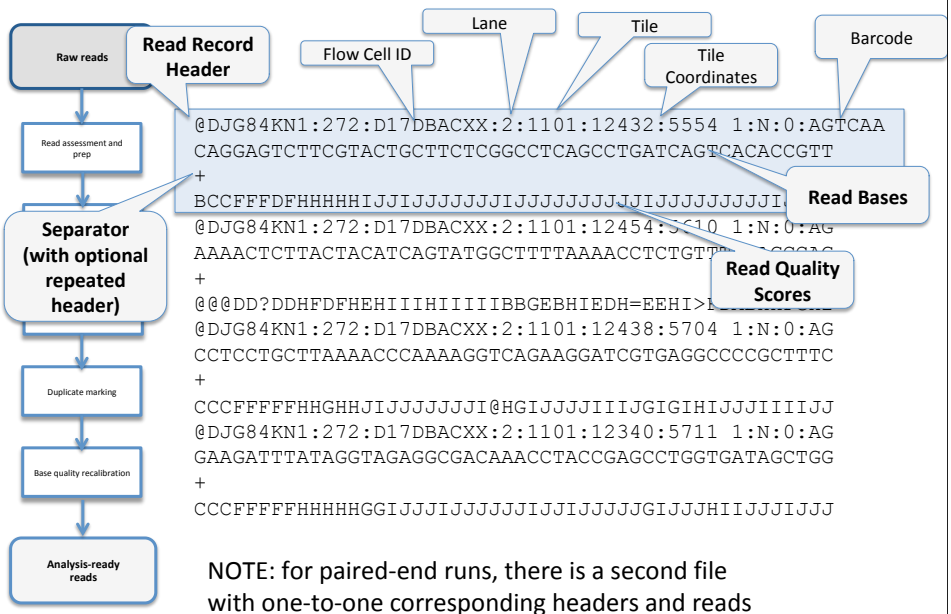
- **FASTA format:** A sequence record in a FASTA format consists of a single-line description (sequence name), followed by line(s) of sequence data. The first character of the description line is a greater-than (">") symbol.

https://en.wikipedia.org/wiki/FASTA_format

Example FASTA file:

>seq0
FQTWEEFSRAAEKLYLADPMKVRVVLKYRHVDGNLCIKVTDDLVCLVYRTDQAQDVKKIEKF
>seq1
KYRTWEEFTRAAEKLYQADPMKVRVVLKYRHCDGNLCIKVTDDVVCLLYRTDQAQDVKKIEKFHSQLM
RLME LKVTDNKECLKFKTDQAQEAKKMEKLNNIFFTLM
>seq2
EEYQTWEEFARAAEKLYLTDPMKVRVVLKYRHCDGNLCMKVTDDAVCLQYKTDQAQDVKKVEKLHG
K
>seq3
MYQVWEEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVCLQYKTDQAQDVK
>seq4
EEFSRAVEKLYLTDPMKVRVVLKYRHCDGNLCIKVTDNSVVSYEMRLFGVQKDNFALEHSLL
>seq5
SWEEFAKAAEVLYLEDPMKCRMCTKYRHVDHKLVVKLTDNHTVLKYVTDMAQDVKKIEKLTTLLMR
>seq6
FTNWEEFAKAAERLHSANPEKCRFVTKYNHTKGELVLKLTDDVVCLQYSTNQLQDVKKLEKLSSTLLRSI
>seq7
SWEEFVERSVQLFRGDPNATRYVMKYRHCEGKLVLKVTDDRECLKFKTDQAQDAKKMEKLNNIFF
PDTTRYVVKYRHCEGKLVLKVTDNHECLKFKTDQAQDAKKMEK





NOTE: for paired-end runs, there is a second file with one-to-one corresponding headers and reads

Phred* quality score *Q* with base-calling error probability P

$Q = -10 \log_{10} P$

* Name of first program to assign accurate base quality scores. From the Human Genome Project.

Q score	Probability of base error	Base confidence	Sanger-encoded (Q Score + 33) ASCII character
10	0.1	90%	"+"
20	0.01	99%	"5"
30	0.001	99.9%	"?"
40	0.0001	99.99%	"I"

```
 SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
 ..............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
 LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL.....................................................
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 |                         |    |        |                              |                     |
33                        59   64       73                            104                   126

S - Sanger        Phred+33  range: 0 to 40
I - Illumina 1.3+ Phred+64  range: 0 to 40
L - Illumina 1.8+ Phred+33  range: 0 to 41
```

Searching the contents of a file

- **Searching using more**

For example, to search **myfile** for the word science, type

```
more myfile
```

then type

```
/ science
```

Type **n** to search for the next occurrence of the word

GREP (a **powerful** approach)

- Searching using **grep**

```
> grep music myfile
```

- To ignore upper/lower case distinctions, use the -i option

```
> grep -i music myfile
```

- To search for a phrase or pattern, you must enclose it in single quotes. For example to search for the phrase operating systems, type

```
> grep -i 'operating systems'
  myfile
> grep -i 'operating systems' *
```

Searching the contents of a file

Some of the other options of grep are:

```
-v  display those lines that do
  NOT match
-n  precede each matching line
  with the line number
-c  print only the total count of
  matched lines
```

Whole books are written on using GREP.

Other Useful Commands

wc (word count)

- To do a word count on myfile, type

  `wc -w myfile`

- To find out how many lines the file has, type

  `wc -l myfile`

- To do both

  `wc myfile`

Redirecting Input and Output

- In general, Unix commands use the standard input (keyboard) and output (screen).
- **<** : redirect input
- **> and >>** : redirect output

Redirecting Input and Output

- An example: search for the word mysort in all the c source files in the current directory and write the output to file1.

- **`grep mysort *.c > file1`**

Using redirection to concatenate files

- Examples:

```
cat  file1 > file2
```

copies file1 into file2

- To concatenate files:

```
cat  file1  file2 > file3
```

- or

```
cat  file2 >> file1
```

Pipes

- A pipe is a way to use the output from one command as the input to another command without having to create intermediary files.
- Example: want to see who is logged in, and you want the result displayed alphabetically:

```
who > namelist
sort namelist
```

- Using a pipe:

```
who  |  sort
```

Protecting files and directories

- The **`ls -l`** command display detailed listing of a file, including its protection mode:

```
drwxrwxrwx  owner  size directoryname …..
-rwxrwxrwx  owner size filename …
```

- the first character (**`d`** or **`-`**) indicates whether it is a file or directory name.
- The following 9 character indicate the protection mode.

Protecting files and directories

`rwx rwx rwx`

- Each group of three characters describes the access permission: read , write and execute
- the first three settings pertain to the access permission of the owner of the file or directory, the middle three pertain to the group to which the owner belongs, and the last three pertain to everyone else.

Access rights on files.

- **r** (or **-**), indicates read permission, that is, the presence or absence of permission to read and copy the file
- **w** (or **-**), indicates write permission; that is, the permission to change a file
- **x** (or **-**), indicates execution permission; that is, the permission to execute a file, where appropriate

example: -rwxrw-r--

Access rights on directories.

- **r**: allows users to list files in the directory;
- **w**: means that users may delete files from the directory or move files into it.
  - Never give write permission to others to your home directory or any of its subdirectories.
- **x**: means the right to access files in the directory. This implies that you may read files in the directory if you have read permission on the individual files.

example: drwxrw-r--

Changing file access permission

chmod (changing protection mode)

- Consider each group of three to be a 3-bit number

example: you want to set permission to

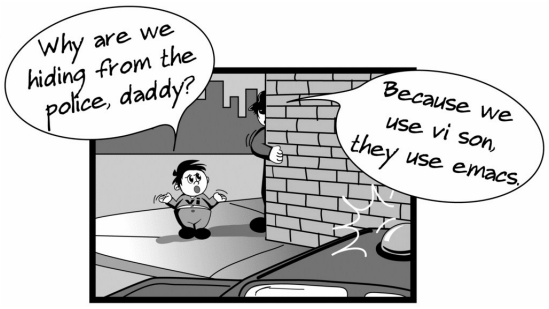
```
rwx r-- ---
111 100 000   (in binary)
7   4   0
```

chmod 740 filename

chmod [**u**ser/**g**roup/**o**thers/**a**ll]+[permission] [file(s)]

Creating files in Unix/Linux

- Can "touch" to create an empty file, can copy an existing file, etc.
- But novel content requires using an editor
- Various Editor
  1) vi / vim
  2) emacs



Recommended Reading:

Bill Noble's **A Quick Guide to Organizing Computational Biology Projects**

PLoS Comput Biol 5(7): e1000424. doi:10.1371/journal.pcbi.1000424

Assignment this week:

- Practice interacting with Fiji: login, move files around, create and edit files
- Create a README file in your home directory that you will use throughout the class. Include appropriate documentation of any files/ directories created today.

Prepare for next week:

- We will be going over transferring files, process management, and submitting jobs to the cluster − we will run our first sequence quality analysis.

- Recommended Videos:
  - Day 3: Cluster usage
  - Day 4: Data Quality (Sequencing QC)