File Types and Permissions

Info
- Change Mode (chmod)
  - Change permissions
  - Must have permission to act on a file or directory
  - Can’t do things with a file you have permissions on if it is in a folder that you don’t
  - Same applies to others you want to allow access to files and directories
  - This can create a security risk!

Demo
- chmod [ugoa] [rwx]-
  - u=owner, g=group, o=other, a=all, r=read, w=write, x=execute
  - $: chmod u+x Test_1
  - $: ls -l #see the change?
  - $: chmod u-x Test_1 #more on executes next time

Local vs Remote

“The Network”
Bandwidth is important

There are “bottlenecks” on the network

Archival Storage

Login Node

Outgoing HPS “NexApp”

300TB

New HPS “Panasas”

Building

BOYD

GACRC

Building

Login Node

300TB

New HPS “Panasas”

Local vs Remote

“The Network”
Bandwidth is important

There are “bottlenecks” on the network

Archival Storage

Login Node

Outgoing HPS “NexApp”

300TB

New HPS “Panasas”

Building

BOYD

GACRC

Building
Reading Files (and getting used to terminal output)

- **Reading Files**
  - `cat` can read and combine files
  - `>` specifies that the output of the command be written to a file
    - When `cat` was used to make a file and to combine a file
  - Can’t act on things that are not in the working directory. Unless you are in the same place, or give the file path

Getting Things Moving

- **Info**
  - Oops! Text files should have the `.txt` extension
  - Rename with move (mv)
    - Specify what and where
      - Careful with file extensions!
  - Use mv to move files in terminal
    - Specify what and where
  - `*` is a wildcard character
    - Matches any files or folders with names that contain what comes before or after it
      - `name*` or `*name`

- **Demo**
  - `$: mv Test_1 Test_1.txt`
  - Move Test_1.txt into Test_3 in Finder
  - `$: mv* Test_3 #What happened?`
  - `$: mv Test_3 TestDir_3 #What happened?`
  - `$: mv TestDir_3/* . #Should see 3 text files and TestDir_3`
  - `$: rmdir TestDir_3 #Gone forever`
  - `$: rm Test_1.txt`
  - `$: rm -i Test_2 #safer, asks if you are sure`

Making Copies

- **Copy (cp)**
  - Works like move
  - Copy to different directory
    - `$: cp file location`
  - Copy to same directory
    - Need new name for copy
    - `$: cp file_1 file_2`
  - Can copy whole directories
    - `$: cp -R directory location`
  - Copy from somewhere else to current directory
    - `$: cp file/path`

Recording your work

- **Copy (cp)**
  - Most of my research notes are in text files
  - Copy and paste from the terminal
    - Use `#`s to mark comments
  - This is how you will hand in most of your homework for this section, as text files with copied terminal output
  - Example in first homework assignment
Remote Login from Windows

- It is possible to simulate a UNIX environment on a Windows machine
  - Use a Mac or UNIX machine if possible (eliminates variables)
  - Use Cygwin to log in
  - Cygwin will give you root access to your Windows machine, use carefully

- [http://sitesoft.uga.edu/](http://sitesoft.uga.edu/) (Look around the page)
  - SSH (Secure Shell Utility for Windows) - Software that allows you to connect securely to computers on the internet. Supports secure file transfers.

When Things Go Bad...

- Errors will happen
- Most common errors are typos and file path mistakes (which are often typos)
- Less often, permission errors
- If you are lucky, you get an error message. Pay attention to these!
  - Can be simple or cryptic
  - Try to interpret them

X Windows (X11)

- Applications/Utilities/X11
- Some UNIX programs are written to use a GUI
- X11 is a special kind of terminal for simulating a Windows or Mac-like environment
  - Simple explanation
  - I don’t use this much, but sometimes it is necessary
- Terminal behaves (mostly) like the shells we have been using
  - Many users prefer use X11

Demo

\$: xclock --analog #run xclock with analog arg
\$: xclock --digital & #run in background
\$: man xclock #man pages as you are used to
\$: kill process_id #kill process in background

X Windows via ssh

- Programs can be X Windows only, or for both X11 and Terminal
- Main features
  - UI (Keyboard, Mouse Screen) is on the machine you are physically at (aka Server)
  - X Windows Client is the machine actually running the program
- CygwinX or Xming allow you to ‘tunnel’ an ssh connection from Windows
  - [http://computing.bio.cam.ac.uk/local/x-win.html](http://computing.bio.cam.ac.uk/local/x-win.html) for more information
  - For more info

X Windows (X11)

- Genome assembly programs (consed) on rcluster use X11
  \$: ssh -X user@rcluster.rcc.uga.edu #Tells rcluster you are using X11

A Word On Text Editors for your PC

- Any good text editor should be able to
  - Show invisible characters
  - Show line numbers
  - Search and replace in one or more files with the option of using regular expressions (more later)
  - Softwrap text
  - Compare documents
  - Operate on text files (Text Menu)
- TextWrangler is free version of BBEdit
Text Editors and Line Breaks

- UNIX is picky about line breaks
- Hit Return and the effect is the same in any editor or word processor, you move down a line
- Actually creates a newline carriage return or line feed character depending on program
  - \n or \r (Turn on invisibles and look in BBedit)
- Save in a UNIX state
  - Microsoft word does not
- Most text editors give the option between UNIX, DOS and (on a Mac) Mac save states

Shell Scripts

- UNIX = programming
  - Who needs a ‘fancy’ language
- A ‘Script’ is a text file (usually) that contains a program that can be ‘executed’ by the shell
- A ‘Shell Script’ is just a series of UNIX commands in a text file
- Can get quite a lot done this way, as you will see in the coming lectures

Unix Text Editors

Info
- Can create files within terminal
  - Already seen cut command
  - There are UNIX text editors
- Ideal for soft work, don’t have to move files, just make them
- Features are aimed at programming
- Takes a little practice, some based on more archaic command structures
- vi, (x)emacs, pico, nano
- Cheat sheets and info online

Demo
- #from Desktop
  - $nano NANO_TEST.txt
- #Ctrl x to close
  - $ls -l on working directory
  - $cat NANO_TEST.txt
  - $chmod u+x NANO_TEST.txt
  - $ls -l
  - $cat

Your First Script!

Info
- 1st line only, UNIX reads through ‘#’
  - ‘!’ = this file is a program and the following path is where to find executable program for this shell
- Save as Hello.sh
- ‘.’ means current directory

Demo
- #from Desktop
- $ cd Desktop
- $ nano
- $! /bin/bash
- #My first UNIX Shell Script
- # echo “Hello World!”
- #!save and close
- $ cat Hello.sh
- $ chmod u+x Hello.sh
- $ ./Hello.sh #executes

One More Shell Script

#! /bin/bash
#This script will list the files in the current directory. This is a comment within the script
#echo “hello, $USER. I wish to list some files of yours”
#echo “listing files in the current directory, $PWD”
ls # list files
Environmental Variables: Behind the Scenes

- Environmental Variables:
  - $USER, $PWD (TEST2.sh)
  - Like arguments for commands, but for the system
  - Can be changed

- $PATH
  - Colon delimited list of locations where the shell looks for commands
  - Can make a script work from anywhere if you add its location to $PATH

$PATH Finder

- The shell checks multiple locations for $PATH on startup
- Mac (read at startup and executed like a shell script)
  - Checked in this order
  - ~/.profile
  - ~/.bash_profile (hidden, ls -a)
  - ~/.bash_login (hidden, ls -a)
  - ~/.profile
    (hidden, ls -a)
- Add line 'export PATH=~/Full/Path/you/want/to/add' to one of these and it is added to $PATH on startup
  - This is the same command we saw in the terminal on the last slide

$PATH Finder

- Not all accounts have these
  - RCC uses ~/.profile
  - Class machines don’t use ~/.profile
  - Can make a ~/.file for your home directory (more in HW)
- Unless you make it permanent by adding to one of the above files, additions to $PATH reset when you quit the terminal program
  - If you make a change to a terminal setting and it doesn’t take effect, restart the terminal

Working with sequence files

- Fasta files
- Match making with grep
- Linking commands: Pipe
- Standard I/O
- More grep
  - Regular Expressions
  - Counting
- Search and Replace: tr and sed
- GFF files
  - cut, sort, uniq
- Getting Serious

Fasta Format

- Standard format for sequence data
- Each sequence has 1 header line that starts with ‘>’ followed by other info
- Sequence may be on 1 or many lines
- File may have 1 or many sequences
  - DNA or amino acid
- Very common
- For DNA, file represents both strands
Match Making with **grep**

**Info**
- Often you will have to examine contents of LARGE files
  - # of sequence in fasta file
  - # of lines
  - Gene names, motifs
  - Extract information
  - etc.
- Can do this in BBEdit:
  - Count seqs in PF.fasta
- grep matches patterns
  - Extremely useful command!
  - Can match any pattern in any file

**Demo**
- From ~/Desktop/Data/
  - $ grep "ATG" PF.fasta
  - Prints all non-header lines
  - $ grep -v "ATG" PF.fasta
  - All by matching "<>"
  - Can make lots of output and take a while to complete if file is large

**Linking Commands with |**

**Info**
- Pipe "|
  - Find on keyboard, it hides
  - Tells shell to send output of a command as input to another command
  - Won’t work if second command doesn’t accept input
  - Connects commands and programs

**Demo**
- From ~/Desktop/Data/
  - $ grep "ATG" PF.fasta | grep -v "ATG" PF.fasta
  - Connects commands and programs

**Standard I/O**

**Info**
- Commands (and Scripts) read from stdin and write to stdout.
- By default, stdin comes from terminal and stdout goes to terminal:
  - < reads contents of a file as input
  - > writes output to a file
  - > > appends output to a file
  - | pipes output to a command

**Demo**
- Mostly use "<" and ">
  - From ~/Desktop/Data/
    - $ grep "ATG" PF.fasta | grep -v "ATG" PF.fasta
    - Connects commands and programs

**grep: global regular expression print**

**Info**
- Search for patterns instead of literal text like “ATG”
  - Very powerful
  - Can be complex and quite specific
  - BBEdit (TextWrangler) -> Help -> Grep Reference
  - Great resource!
- Much more on regular expressions in PERL
  - grep skill in UNIX = regular expression skill in PERL (vice versa)
- YOU CANNOT BE ‘TOO GOOD’ AT THIS!

**Counting with grep**

**Info**
- Already counted number of sequences in PF.fasta with BBEdit
- Grep can do more than just show you output, it can calculate

**Demo**
- From ~/Desktop/Data/
  - $ grep "ATG" chr.fasta
  - Only returns counts for files with the pattern
  - "ATG" is not the same in UNIX and grep
- Match anything (UNIX) vs match one or more of the preceding character (grep)
Regular Expressions and `less`

**Info**
- You can search while viewing a file with `less` (completely different?)
- Pattern into something

**Demo**
- From `~/Desktop/Data/`:
  - `grep "ATG" PF.fasta | less`
  - Finds all lines that contain ATG
  - Once it is open in `less`, type `/ ATG` and hit enter, searches forward
  - Can search with regular expressions within less `/ATG`
  - `/?` like `/` but searches backwards

Search and Replace: `tr`

**Info**
- `tr` (transliterate) can change one range of characters into another set
- Eg: some programs may only work on upper or lower case sequences
- You can overwrite original file if you want to, not a good idea
  - Only delete original after correct changes are confirmed
  - Better yet, keep all originals!

**Demo**
- From `~/Desktop/Data/`:
  - `head –n 2 chr.fasta` #portion of chr 1 from A. thaliana
  - `head –n 2 chr.fasta | tr 'A-Z' 'a-z'`
  - Change only shows in standard out, file is unchanged
  - `head –n 2 chr.fasta | tr 'A-Z' 'a-z' > TEST.fasta`
  - `ls`
  - `cat TEST.fasta` #new file has changes
  - `rm TEST.fasta` #clean up

Search and Replace: `sed`

**Info**
- What if you want to change a pattern into something completely different?
  - BBEdit has a search and replace
  - UNIX has the Stream Editor (`sed`)
  - Extremely powerful!
- In the Demo, we will change a fasta header

**Demo**
- From `~/Desktop/Data/`:
  - `head –n 1 chr.fasta`
  - `head –n 1 chr.fasta | sed ‘/Chr1/ Chromosome_1/’`
  - `/s` = substitute mode
  - `/s/pattern_to_find/ pattern_to_replace/`
  - `like with tr, if you want to change the file, have to redirect std out`

GFF Files

**Info**
- Describes the location of genomic features on a chromosome (or any DNA molecule)
- Many generic GFF formats
  - Second most common file type
- Gene finders often output in GFF
- Features can be exons, genes, binding sites, etc
- 9 tab-delimited fields for each feature (1 per line)
  - NAME\SOURCE\TYPE\START\END\SCORE\STRAND\FRAME\GROUP
  - `\` regular expression match for tab

**Demo**
- From `~/Desktop/Data/`:
  - `$: more genes.gff` # ~ half of the features on A. thaliana chr 1
  - Something in every field, even if it is just ‘-‘
  - Last field is a catch all
  - `wc –l genes.gff` # of lines in the file = number of features
  - `wc` can also count characters in a file
    - This can tell you the number of base pairs in a fasta file
    - How many different features are in the GFF file?
    - Could load into Excel…
      - ~177k lines? No thanks.
    - From `~/Desktop/Data/`:
      - `$: cut –f 3 genes.gff`
    - Luckily, cut ‘cuts’ at \t so this worked well
      - Can specify other delimiters with argument to cut command
      - Not there yet…
**cut, sort**

- From `~/Desktop/Data/

$:$ cut -f 3 genes.gff | sort

- Does just what it sounds like, sorts alphanumerically
  — Still too much output

**cut, sort, uniq**

- Pipes are your friend

- From `~/Desktop/Data/

$:$ cut -f 3 genes.gff | sort | uniq

- Removes all but one of each item in a list
  
  - **Have to sort first!**
  - That’s more like it!
  - Could pass std out to a file with ‘>’

**Getting Serious**

- From `~/Desktop/Data/

$:$ cut -f 3,4 genes.gff | sort -n -k 2 | grep "gene" > TEST.gff

- What is this doing?

**Getting Serious**

- From `~/Desktop/Data/

$:$ cut -f 3,4 genes.gff | sort -n -k 2 | grep "gene" > TEST.gff

- Pipe to sort, -n says sort numerically and not alphanumerically based on, -k 2, the second column

- Pipe to grep to locate lines that contain "gene"

- Matching lines are redirected from std out to TEST.gff

$:$ less TEST.gff

**OhMyLookWhatICanDo.sh**

```
#!/bin/bash

# This shell script will cut the 3rd and 4th columns
# (presumably the feature type and start coordinate)
# from a gff file named 'genes.gff', sort them by start coordinate, extract all feature types that contain
# the pattern "gene" and return the results to the file 'TEST.gff'. You can alter the script to search for other
# fields, search a different file or return a different file name

echo Beginning Search...
cut -f 3,4 genes.gff | sort -n -k 2 | grep "gene" > TEST.gff
echo Search complete. Results should be in the working directory.
```

**Where to find material for exercises**

- Copies of all slides and handouts are posted at: [http://qbcg.uga.edu](http://qbcg.uga.edu) in the BBB tab

- Text files and shell scripts discussed in slides and demonstrated in class can be found on the GACRC zcluster at: /

  NOTE: this directory is read only. You can only look and copy material out of this directory.