Unix Scripting

Michael Schatz

Sept 3, 2013

QB Bootcamp Lecture 4

Outline

Part 1: Overview & Fundamentals

Part 2: Sequence Analysis Theory

Part 3: Genomic Resources

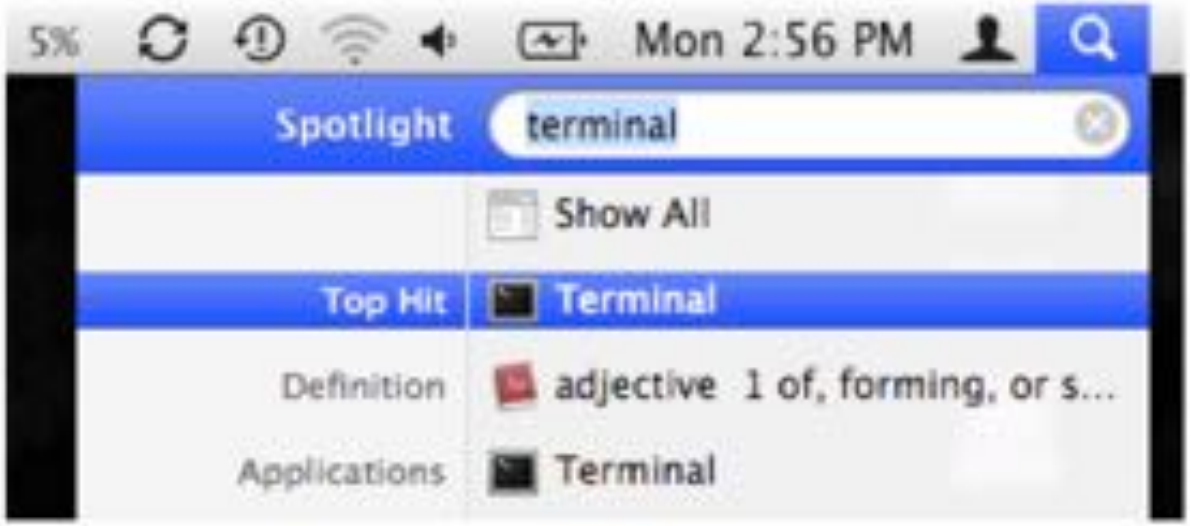
Part 4: Unix Scripting

Part 5: Example Analysis

How does scientific software operate?

- The software we need to run is very specialized, there is no 'analyze genome' button in Excel
  - Data files are huge, so probably wouldn't want one anyways
- It takes a lot of work (and time/money) to create a graphical interface to software, so most scientific software uses a 'command line' interface
  - Important to become comfortable using command line tools
- Scientific analyses tend to use workflows consisting of several applications where the output of one phase becomes the input to the next
  - Develop a workflow for dataset X, apply again to dataset Y

Where is the command line?



- Your Mac has a very powerful command line interface hidden just beneath the graphical environment
  - This command line interface is (basically) the same as that used by our scientific cluster BlackNBlue
  - Big data files are stored on our central storage system BlueArc

- This environment has a universe of programs you can use to manipulate files and data in novel ways
  - Learning to use this environment is a lot like learning a new language
  - http://korflab.ucdavis.edu/Unix_and_Perl/index.html

File Hierarchy

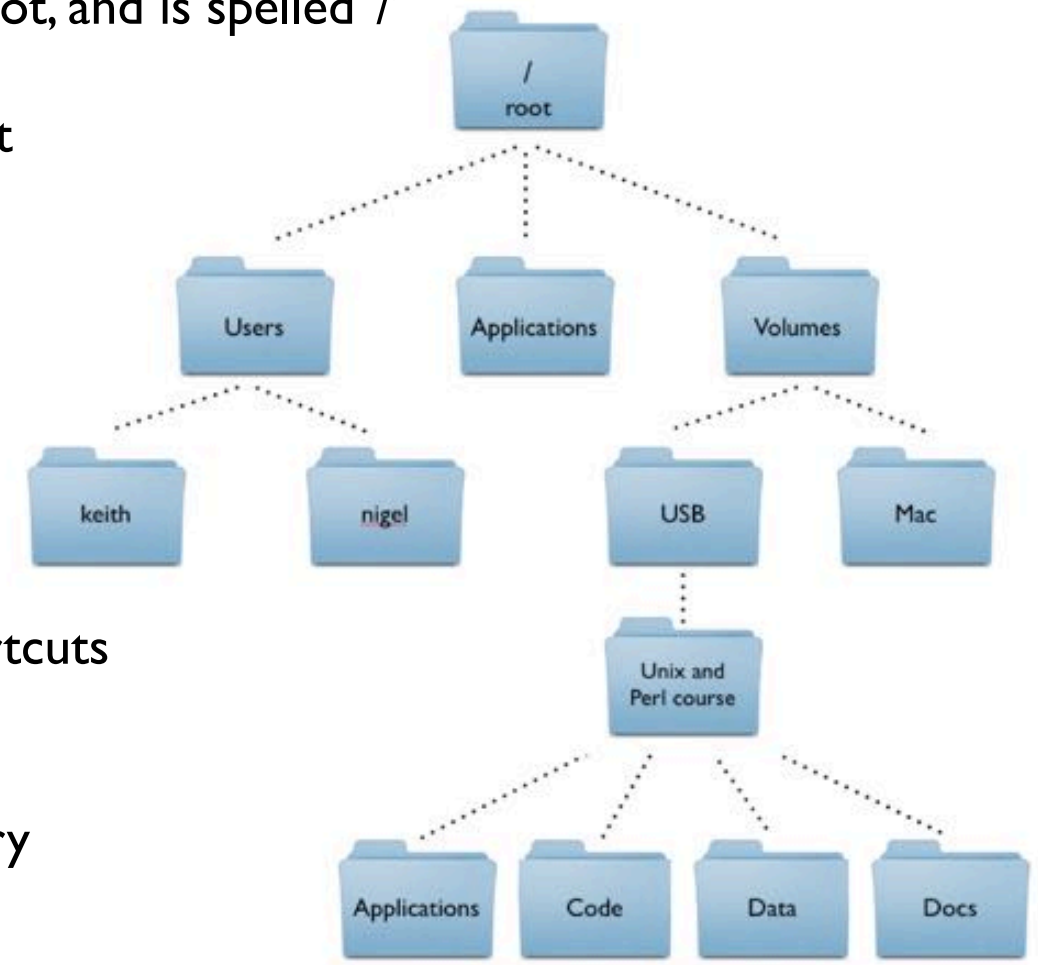
Files are stored in nested directories (folders) that form a tree

- The top of the tree is called the root, and is spelled '/'

- Your home directory (on mac) is at
  /Users/username

- Command line tools are at
  /bin/
  /usr/bin/
  /usr/local/bin/

- A few special directories have shortcuts
  ~ = home directory
  ~bob= bob's home directory
  . = current working directory
  .. = parent directory
  - = last working directory



Working with the shell

- The shell is interactive and will attempt to complete your command as soon as you press enter

```
$ pwd
/Users/mschatz

$ ls
Desktop/     Library/     Public/      bin/     Documents/   Movies/
Downloads/   Music/       Dropbox/     Pictures/
```

- Here are a few shortcuts that will make your life easier

Command	Effect
Left/Right arrow	Edit your current command
Up/Down arrow	Scroll back and forth through your command history
Control-r	Search backwards through your command history
history	What commands did I just run?
Control-c	Cancel the command
Control-u	Clear the current line
Control-a, Control-e	Jump to the beginning and end of the line

Working with files and directories

```
## Create a work directory
$ cd Desktop
$ mkdir human_analysis
$ ls
$ cd human_analysis/

## Download the annotation of the human genome
$ curl -O http://schatzlab.cshl.edu/teaching/2013/hg19.gff.gz

## See how big it is
$ ls –l
-rw-r--r--  1 mschatz  staff  24904770 Sep  2 22:48 hg19.gff.gz

## See how big it is in a human readable way
$ ls –lh
-rw-r--r--  1 mschatz  staff    24M Sep  2 22:48 hg19.gff.gz

## Make a copy
$ cp hg19.gff.gz hg19.2.gff.gz
$ ls

## Rename the copy with the move command
$ mv hg19.2.gff.gz hg19_2.gff.gz
$ ls

## delete the copy
$ rm hg13_2.gff.gz
$ ls
```

Careful what you delete!

Working with (compressed) text files

```
# uncompress compressed files with gunzip
$ gunzip hg19.gff.gz
$ ls -lh
total 1065808
-rw-r--r--  1 mschatz  staff   520M Sep  2 22:48 hg19.gff
# Notice it is >20 times large

## look at the first few lines using the command head
$ head hg19.gff
##gff-version 3
#!gff-spec-version 1.20
#!processor NCBI annotwriter
#!genome-build Genome Reference Consortium GRCh37.p13
#!genome-build-accession NCBI_Assembly:GCF_000001405.25
#!annotation-source NCBI Homo sapiens Annotation Release 105
##sequence-region NC_000001.10 1 249250621
##species http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=9606
NC_000001.10 RefSeq   region   1    249250621    .    +    .
    ID=id0;Name=1;Dbxref=taxon:
•   31267

## How many lines are in the file?
$ wc -l hg19.gff
 1937161 hg19.gff

## page through the file
$ less hg19.gff
```

Working with annotations with grep

```
## Find just the BestRefSeq annotations
$ grep BestRefSeq hg19.gff | less
$ grep BestRefSeq hg19.gff | wc -l
  922144

## Save it to a new file
$ grep BestRefSeq hg19.gff > hg19.BestRefSeq.gff

$ ls -lh
total 1599120
-rw-r--r--  1 mschatz  staff   260M Sep  2 23:11 hg19.BestRefSeq.gff
-rw-r--r--  1 mschatz  staff   520M Sep  2 22:48 hg19.gff

## Count the number of genes
$ grep gene hg19.BestRefSeq.gff | wc -l
  922144

## That doesnt look right, lets focus on column 3
$ cut -f3 hg19.BestRefSeq.gff | sort | uniq -c
387524 CDS
458708 exon
26705 gene
38536 mRNA
2729 ncRNA
1620 primary_transcript
  21 rRNA
6301 transcript
```

26,705 annotated genes,
458,708 annotated exons

17 exons / gene on average

Working with annotations with grep (cont)

```
## Have to ensure the whole field is gene with tabs on either side
$ grep '\tgene\t' hg19.BestRefSeq.gff | wc -l
   26705

## Save the genes to a file
$ grep '\tgene\t' hg19.BestRefSeq.gff > hg19.BestRefSeq.gene.gff


## Save it to a new file
$ grep BestRefSeq hg19.gff > hg19.BestRefSeq.gff



## Count genes per chromosome
$ cut -f1 hg19.BestRefSeq.gene.gff | sort | uniq -c | sort -nrk1 | head -3
2426 NC_000001.10
1632 NC_000019.9
1510 NC_000002.11
```

Should we be surprised that chromosome 1 has the most genes?

```
## How many chromosomes total
$ cut -f1 hg19.BestRefSeq.gene.gff | sort | uniq -c | sort -nrk1 | wc -l
     217
```

Why are there so many chromosomes?

Programming Basics: Loops

- A bash script is just a list of commands

```
$ cat simple_script.sh
#!/bin/sh

echo "Hello, World”
echo "Shall we play a game?”

$ chmod +x simple_script.sh
$ ./simple_script.sh
```

[What does this do?]

- Things get interesting when we add variables and loops

```
$ cat loop_script.sh
#!/bin/sh

for chrom in NC_000001.10 NC_000002.11 NC_000003.11
do
  echo Searching $chrom
  grep $chrom hg19.BestRefSeq.gene.gff > $chrom.BestRefSeq.gene.gff
done

$ chmod +x loop_script.sh
$ ./loop_script.pl
```

[What does this do?]

Unix Review

Command	Output
man	Look up something in the manual (also try Google)
ls	List the files in the current directory
cd	Change to a different directory
pwd	Print the working directory
mv, cp, rm	Move, copy, remove files
mkdir, rmdir	Make or remove directories
cat, less, head, tail, cat	Display (parts) of a text file
echo	Print a string
sort, uniq	Sort a file, get the unique lines
grep	Find files containing X
chmod	Change permissions on a file
wc	Count lines in a file
| (pipe), > (redirect)	Send output to a different program, different file

Challenges

- Where is TP53 located? Where is NRAS? Where is SRY?
- How many genes are annotated with “tumor” or “oncogene”?
- Create a file with the RefSeqGenes for each chromosome and sort them by file size

Programming Resources

- Much like learning a new spoken language, computer languages have their own syntax and grammar that will be unfamiliar at first, but get easier and easier over time
  - There are many ways to accomplish the same task
  - You can quickly become a data magician

- The way to learn a new computer language is to practice speaking it
  - The ~30 commands you have seen today can be combined together into an infinite number of combinations
  - Lots of good resources available online:
    - http://www.molvis.indiana.edu/app_guide/unix_commands.html
    - http://tldp.org/LDP/abs/html/index.html
    - http://stackoverflow.com/
    - http://google.com

WARNING: Computers are very unforgiving

- 'rm –rf /' <= delete every file on your computer
- 'cp junk.doc thesis.doc' <= overwrite your thesis with junk.doc
- 'cat results.partial > results.all' <= oops, should have appended with >>

Bonus

Files and permissions

- Every file has an owner and a group, you can only read/write to a file if you have permission to do so

```
$ pwd
/Users/mschatz/Desktop/Unix_and_Perl_course/Data/Arabidopsis

$ ls -l
total 193976
-rw-r--r--@ 1 mschatz  staff  39322356 Jul  9  2009 At_genes.gff
-rw-r--r--@ 1 mschatz  staff  17836225 Oct  9  2008 At_proteins.fasta
-rw-r--r--@ 1 mschatz  staff  30817851 May  7  2008 chr1.fasta
-rw-r--r--@ 1 mschatz  staff  11330285 Jul 10  2009 intron_IME_data.fasta
```

- These files can be read by anyone, but only written by me
  - Change permissions with 'chmod'

```
$ chmod g+w At_*
$ man chmod
```

- Programs and scripts have the execute bit set

```
$ ls -l /bin/ls
-r-xr-xr-x  1 root  wheel  80688 Feb 11  2010 /bin/ls*
```

Editing Files

- You can open files from the shell using "regular" applications by their extension

```
$ cp At_genes.gff At_genes.gff.txt
$ open At_genes.gff.txt
$ open .
$ open /Applications/Microsoft\ Office\ 2011/Microsoft\ Word.app/
```

- It is often helpful (or necessary) to edit files within the terminal

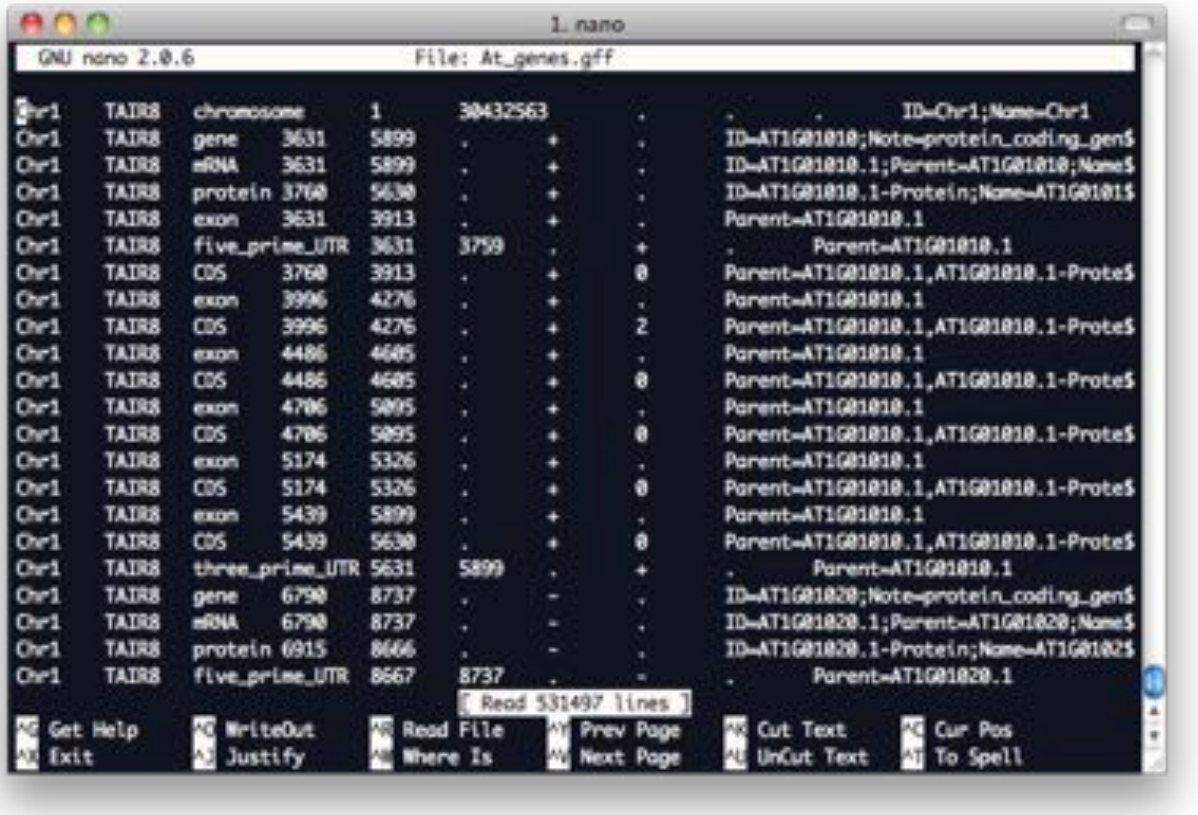
```
$ nano At_genes.gff
```

Basic nano commands

- Type to make edits
- Arrows to move
- Control-O to save
- Control-X to exit
- Control-G for help

Advanced text editors:

- vi
- emacs



Background Processes

- Any number of processes can run in the background
  - Use the ampersand (&) to launch a process into the background
  - Alternatively use control-z to pause a process, then use 'bg'

```
$ du -a /
(control-c to cancel)

$ du -a / | sort -nrk1 > ~/filesizes.txt
(control-z to stop)
$ bg
$ du –a / | sort –nrk1 > ~/filesizes.txt.2 &
```

- List running jobs associated with this shell

```
$ jobs
$ fg %1
(control-z to stop)
$ bg
```

- Kill off run-away commands

```
$ ps
$ kill 61110
$ kill -9 61110
```

61110 is the process id I want to kill
kill -9 for really stubborn processes

Monitoring Processes

- Unix systems can run many commands and by many users at once
  - Especially useful for commands that run for a long time
  - Especially useful for servers that have special resources

```
$ ps
  PID TTY           TIME CMD
60820 ttys000    0:00.30 /bin/bash

$ ps aux | head -3
USER       PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
root     21527   1.7  0.1  3129268   5692   ??  Ss   11Jul12 679:00.75 /
Library/Application Support/iStat local/iStatLocalDaemon
mschatz  62928   1.6  1.4  2986576 119648   ??  S    31Jul12 895:05.37 /
System/Library/CoreServices/SystemUIServer.app/Contents/MacOS/SystemUIServer
```

- Monitor use of the system

```
$ top
(press q to quit)
```

Working with remote servers

- Use SSH to connect to a remote server

```
$ ssh mschatz@bnbdev1.cshl.edu
```

- The server runs UNIX, and the standard commands are available

```
$ ls –l | sort –nrk5 | head -3
$ who
```

- There are special lab directories for CSHL users (> 1PB of storage total)

```
$ df –h /data/schatz* /data/wig*
```

- Your lab may have special commands available

```
$ ls /data/schatz/software/bin/
$ /data/schatz/software/bin/samtools
```

- Typing out the full path for every command is a pain, edit your bashrc

```
$ nano ~/.bashrc
```

(at the bottom add: export PATH=~/bin:/data/schatz/software/bin/:$PATH)
Control-o to save

See: http://intranet.cshl.edu/it/bluehelix/ for details on the shared cluster

Programming Basics: Conditionals

- Conditionals and loops let us work over any number and type of file

```
$ cat conditional_script.sh
#!/bin/sh

for filename in `/bin/ls * | grep –v ”.sh"`
do
  type=`echo $filename | cut -f2 -d'.'`
  echo "Processing $filename, type is $type"
  echo "=================="

  if [[ $type == "fasta" ]]
  then
    protein_count=`grep -c '>' $filename`
    hypo_count=`grep -c hypothetical $filename`
    echo "$filename has $protein_count proteins, $hypo_count are hypothetical"
  elif [[ $type == "gff" ]]
  then
    echo "$filename stats"
    cut -f3 $filename | sort | uniq -c
  else
    echo "Unknown file type"
  fi

  echo "=================="
  echo
done
```

The backtics `<cmd>`
Let us run commands
inside of other commands

[What does this do?]

Programming Basics: Arguments

- The shell defines a few special variables to specify input

```
$ cat argument_script.sh
#!/bin/sh

if [[ $# -lt 2 ]]
then
  echo "USAGE: argument_script.sh proteinsfile type_1 .. type_n"
  exit
fi

echo "Script was run as: $0"
echo "First argument is: $1"
echo "Second argument is: $2"

proteinsfile=$1
shift

while [ $# -gt 0 ]
do
  type=$1
  shift
  count=`grep '>' $proteinsfile | grep -c $type`
  echo "There are $count $type proteins in $proteinsfile"
done

$ ./argument_script.sh At_proteins.fasta F-box GTP-binding hypothetical
```

$# stores number of arguments

$0 has script name
$1-$9 have first 9 arguments

Use shift to access arguments

Loop until there are no more types to consider

Programming Basics: Functions

- A function is a reusable block of code

```
$ cat function_script.sh
#!/bin/sh

function log()
{
  date=`date`
  echo "$date :: $*"
}

function processFasta()
{
  file=$1
  log "Processing fasta: $file"
  num=`grep -c '>' $file`
  log "There are $num sequences"
}

function processGFF()
{
  file=$1
  log "Processing gff: $file"
  num=`wc -l $file`
  log "There are $num records"
}

for file in `/bin/ls *`
do
  log "Processing $file"

  type=`basename $file | cut -f 2 -d'.'`

  if [[ $type == "fasta" ]]
  then
    processFasta $file
  elif [[ $type == "gff" ]]
  then
    processGFF $file
  else
    log "Unknown filetype $type"
  fi
done
```

Scripting Challenges

1. Create 1000 files named mutantA.X.txt with X in [1,1000] that contain the numbers 1 to X

   mutantA.1.txt: 1

   mutantA.2.txt: 1 2

   mutantA.3.txt: 1 2 3

   …

2. Rename 1000 files named mutantA.X.txt to mutantB.X.txt?

   mutantA.1.txt => mutantB.1.txt

   mutantA.2.txt => mutantB.2.txt

   mutantA.3.txt => mutantB.3.txt

   …

3. Identify the files in the given directory that contain a specified keyword and copy them to a specified directory

   ./find_special.sh search_directory 976 destination_directory

   => cp search_directory/mutantB.976.txt destination_directory

   => cp search_directory/mutantB.977.txt destination_directory

   => cp search_directory/mutantB.978.txt destination_directory

   ...

Programming Review

Variables & Arguments

```
names=Mike
names="$names Justin"
names="$names Mickey"
echo $names

echo "There are $# arguments: $*"
shift
echo "The second argument is $1"
```

Conditionals

```
if [[ $type == "fasta" ]]
then
    num=`grep –c '>' $file`
    echo "There are $num seqs"
elif [[ $type == "gff" ]]
then
    num=`wc -l $file`
    echo "There are $num records"
else
    echo "Unknown file type"
fi
```

Loops

```
rm authors.txt
for name in Mike Justin Mickey
do
    echo $name >> authors.txt
    c=`cat authors.txt | wc -l`
    while [ $c -gt 0 ]
    do
        echo $name $c
        c=`echo $c-1 | bc`
    done
done
```

Functions

```
function log()
{
  date=`date`
  echo "$date :: $*"
}
for name in Mike Justin James
do
  log "Processing $name"
  echo $name >> authors.txt
  log "Done with $name"
done
```