Unix, Perl and BioPerl

I: Introduction to Unix for Bioinformatics

George Bell, Ph.D.
WIBR Bioinformatics and Research Computing
• Training
  - Train Whitehead scientists on the use of bioinformatics and graphics tools
• Education
  - Teach courses about theory behind bioinformatics tools and graphics concepts
• Consulting
  - Advise scientists on ways of analyzing data and designing graphics images
• Collaboration
  - Build new bioinformatics tools
  - Use bioinformatics tools to analyze research data
  - Publish papers in the area of bioinformatics with Whitehead scientists
Introduction to Unix for Bioinformatics

- Why Unix?
- The Unix operating system
- Files and directories
- Ten required commands
- Input/output and command pipelines
- Supplementary information
  - X windows
  - EMBOSS
  - Shell scripts
Objectives

• Get around on a Unix computer

• Run bioinformatics programs “from the command line”

• Design potential ways to streamline data manipulation and analysis with scripts
Why Unix (for me)?

- **GEISHA**, the *Gallus gallus* (chicken) EST and in situ hybridization (ISH) database
Why Unix (in general)?

- Features: multiuser, multitasking, network-ready, robust
- Others use it – and you can benefit from them (open source projects, etc.)
- Good programming and I/O tools
- Scripts can be easily re-run
- Types: Linux, Solaris, Darwin, etc.
- Can be very inexpensive
Why Unix for Bioinformatics?

- Good for manipulating lots of data
- Many key tools written for Unix
- Don’t need to re-invent the wheel
- Unix-only packages: EMBOSS, BioPerl
- Unix tools with other OSs: Mac (OS X) & PC (Cygwin)
Unix O.S.

- **kernel**
  - managing work, memory, data, permissions
- **shell:**
  - working environment and command interpreter
  - link between kernel and user
  - choices: tcsh, etc.
  - History, filename completion [tab], wildcard (*)
  - Shell scripts to combine commands
- **filesystem**
  - ordinary files, directories, special files, pipes
WIBR BaRC systems

Training

hebrides
(Solaris; 2 CPUs)
with storage and filesystem
/home/username

Research

storage and filesystem:
/home/username

barra (Linux; 4 CPUs)

Linux cluster: ~21 x 2 CPUs
Logging in

• ssh (secure shell; for encrypted data flow)
  
  `ssh -l user_name hebrides.wi.mit.edu`

• passwd: to change your passwd

• logging out
  `logout`
Intro to files and directories

• Arranged in a branching tree
• Root of tree at “/” directory
• User elvis lives at /home/elvis
  (on ‘hebrides’)
• No spaces allowed
• Full vs. relative pathnames
  – At his home, Elvis’ home dir is “.”
  – To get to /home/gidget, go up and back down:
    (../gidget relative to   /home/elvis)
• Anywhere, your home directory is “~”.

Unix, Perl, and BioPerl © Whitehead Institute, 2005
Intro to Unix commands

• Basic form is
  command_name options argument(s)
  examples:
  mv old_data new_data
  blastall -p blastn -i myFile.seq -e 0.05
    -d nt -T T -o myFile.out

• Use history (↑, ↓, !num) to re-use commands
• Cursor commands: ^A(beginning) and ^E (end)
• To get a blank screen:  clear
• For info about a command:  man command
Key commands p. 1

• Where am I?
  ```
  elvis@hebrides[1] % pwd
  /home/elvis
  ```

• What’s here?
  ```
  elvis@hebrides [2] % ls
  A01.fa
  ```
  ```
  elvis@hebrides [3] % ls -a
  .  .cshrc  A01.fa
  ..  .twmrc
  ```
  ```
  elvis@hebrides [4] % ls -l
  -rw-r--r--  1 elvis musicians  1102 Jun 19 10:45 A01.fa
  ```
Key commands p. 2

• Change directories:
  cd ../gidget
  /home/gidget

• Make a new directory:
  mkdir spleen

• Remove a directory (needs to be empty first):
  rmdir spleen
File permissions

• Who should be reading, writing, and executing files?
• Three types of people: user (u), group (g), others (o)
• 9 choices (rwx or each type of person; default = 644)
  0 = no permission          4 = read only
  1 = execute only          5 = r + x
  2 = write only            6 = r + w
  3 = x + w                  7 = r + w + x

• Setting permissions with chmod:
  chmod 744 myFile or chmod u+x myFile
  -rwxr--r-- 1 elvis musicians 110 Jun 19 10:45 myFile
  chmod 600 myFile
  -rw------- 1 elvis musicians 110 Jun 19 10:45 myFile
Key commands p.3

• Copying a file:
  cp [OPTION]... SOURCE DEST
  Ex: cp mySeq seqs/mySeq

• Moving or renaming a file:
  mv [OPTION]... SOURCE DEST
  Ex: mv mySeq seqs/mySeq

• Looking at a file (one screenful) with ‘more’
  Ex: more mySeq
  (Spacebar a screenful forward,
  <enter> a line forward; ^B a screenful back; q to exit)
Key commands (summary)

ssh    mkdir    cp
pwd    mvdir    mv
ls    chmod    more

cd

To get more info (syntax, options, etc.):
man command
Input/output redirection

- Defaults: stdin = keyboard; stdout = screen
- To modify, 
  \texttt{command < input\texttt{File} > output\texttt{File}}
- input examples
  \texttt{sort < my\_gene\_list}
- output examples
  \texttt{ls > file\_name} (make new file)
  \texttt{ls >> file\_name} (append to file)
  \texttt{ls foo >& file\_name} (stderr too)
Pipes (command pipelines)

- In a pipeline of commands, the output of one command is used as input for the next

- Link commands with the “pipe” symbol: |
  ex1: `ls *.fa | wc -l`
  ex2: `grep ‘^>’ *.fa | sort`
Managing jobs and processes

• Run a process in the foreground (fg):
  `command`

• Run a process in the background (bg):
  `command &`

• Change a process (fg to bg):
  1. suspend the process: `^Z`
  2. change to background: `bg`
Managing jobs and processes (cont.)

• See what’s running (ps)
```
elvis@hebrides[1]% ps -u user_name
```

```
  PID  TTY      TIME  CMD
22541  pts/22   0:00  perl
22060  pts/22   0:00  tcsh
```

• Stop a process:
```
kill PID
```
```
ex: kill 22541
```
Text editors

• emacs, vi (powerful but unfriendly at first); pico

• nedit, xemacs (easier; X windows only)

• desktop text editors (BBEdit; TextPad) + sftp
Supplementary information
**X Windows**

- method for running Unix graphical applications
- still allows for command-line operation
- see help pages for getting started
- some applications with extensive graphics:
  - EMBOSS
  - R
  - Matlab
  - ClustalX + TreeView
- Requires a fast network/internet connection
EMBOSS

• The European Molecular Biology Open Software Suite
• List of programs at http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/
• ex: Smith-Waterman local alignment (water)
• Programs have two formats: interactive and one-line
• Conducive to embedding in scripts for batch analysis
• Traditionally command-line but web interfaces are becoming available
EMBOSS examples

- **needle**: Needleman-Wunsch global alignment
  
  ```
  needle seq1.fa seq2.fa -auto 
  -outfile seq1.seq2.needle
  ```

- **dreg**: regular expression search of a nucleotide sequence
  
  ```
  dreg -sequence mySeq.tfa -pattern GGAT[TC]TAA -outfile mySeq_dreg.txt
  ```
Shell script example

#!/bin/csh
# alignSeqs.csh: align a pair of sequences

# Check to make sure you get two arguments (sequence files)
if ($#argv != 2) then
  echo "Usage: $0 seq1 seq2"; exit 1
endif

# Local alignment
set localOut=$1.$2.water.out
water $1 $2 -auto -outfile $localOut
echo Wrote local alignment to $localOut

# Global alignment
set globalOut=$1.$2.needle.out
needle $1 $2 -auto -outfile $globalOut
echo Wrote global alignment to $globalOut
Some other helpful commands

- **rm**: remove (delete) files  
  **ex:** `rm myOldfile`
- **cat**: concatenate files  
  **ex:** `cat *.seq > all_seq.tfa`
- **alias**: create your own command shortcuts  
  **ex:** `alias myblastx blastall -p blastx -d nr`
- **find**: find a lost file (ex: look for files with the .fa extension)  
  **ex:** `find . -name \*.*.fa`
- **diff; comm**: compare files or lists
- **sort**: sort (alphabetically/numerically) lines in a file
- **uniq**: get list of non-redundant lines
- **grep**: search a file for a text pattern
- **tar**: combine files together for storage or transfer
- **sftp**: transfer files between machines
- **gzip & gunzip**: compress or uncompress a file
Summary

• Why Unix?
• The Unix operating system
• Files and directories
• Ten required commands
• Input/output and command pipelines
• X windows, EMBOSS, and shell scripts
Exercises

• compress, move, and uncompress sequence files
• make a multiple sequence file
• create a BLAST database
• run BLAST on your database
• extract a sequence from the database