Unix at the command line

Goals of today's lecture:

- introduction to the unix command line
- unix file manipulation
  - ls, cp, mv, mkdir, cd, pwd, more/less, head, tail
- other unix commands
  - cut, curl, grep, man
- using a Unix editor (emacs)
- from command to shell script

1

For more information?

- Practical computing (HD), ch. 4,5,7,9
- Unix and Perl primer: korflab.ucdavis.edu/Unix_and_Perl/

(we will be using Python, not Perl)

- Learn Python the Hard Way: learnpythonthehardway.org/book/
- Think Python (collab) www.greenteapress.com/thinkpython/thinkpython.pdf

Exercises:

1. Open the Mac "terminal" app (`/Applications/Utilities/terminal.app`)
2. Create a directory: `ecg`
3. in that directory, create a file (gst.accs) containing Uniprot GST accs
4. edit the file and display it
5. Use the "curl" command to download a sequence
6. write a file of shell (bash) commands to download those sequences the sequences from "gst.accs" from Uniprot

2

Computing environments

- UNIX computing: the command line
  - "shell" environment, built-in tools
  - infinitely extensible: download/install tools
    - most bioinformatics algorithms/tools are implemented as UNIX command line utilities or libraries
    - or, write your own algorithms/tools from scratch
  - highly automatable by scripting (sh, python, etc.)
  - interoperation between tools only limited by your ability to glue together input/output formats
  - almost entirely free access to tools
- demo

3

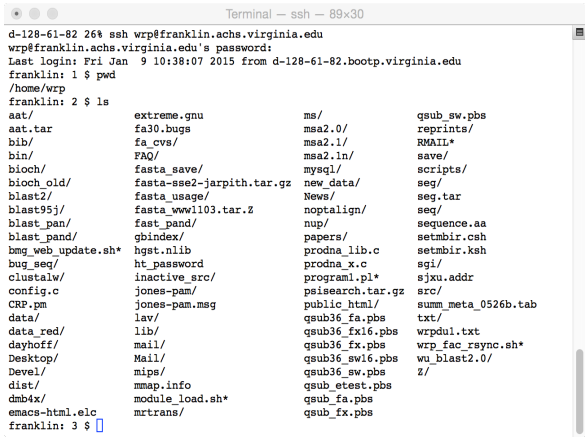
Using the Unix "terminal"

After logging in:

1. to see current location: `pwd`
2. to list files in directory: `ls`
3. logout: `^D` (ctrl-D) or "`exit`"

^C (ctrl-C) for emergencies

MacOS terminal



4

UNIX file editors

- UNIX newlines are "`\n`"
  - PC is "`\r\n`"; Mac is "`\r`" (sometimes);
- Use a UNIX editor on UNIX files:
  - `nano`
  - `emacs` vs. `vi`/`vim`
  - do not use: Word, NotePad/WordPad, TextEdit, etc.
- every editor has pros and cons (focus on nano and emacs if starting out)

5

filesystem navigation

- UNIX filenames are "case-sensitive"

```
seq.file != Seq.file
```

  - lower case only, only "a-z_0-9" (avoid '/', '[]')
- `cd` – change directory
- `pwd` – print working directory (current dir.)
- `ls` – list files
- `pushd`/`popd` – `cd`, but remember stack
- `find` – search through filesystem
- `basename`/`dirname` – extract filename pieces

6

filesystem manipulation

- `cp` – copy files
- `mv` – move files
- `rm` – remove files
- `rmdir` – remove directories
- `touch` – make a new, empty file
- `mkdir` – make a new, empty directory

7

file inspection

- `more` – read/browse through a file/stdin
- `cat` – dump file contents to stdout
- `head`/`tail` – print first/last N lines
- `od` – look at the raw data
- `sort` – sort the lines in the file
- `uniq` – report unique lines
- `cut` – extract specific columns
- `grep` – search for matching lines
- `wc` – count words/lines/characters

8

UNIX permissions

- `chmod` – change the permissions on a file/dir
- `chown` – change the ownership of a file/dir
- `chgrp` – change the group of a file/dir

the UNIX `$PATH`

Unix uses the $PATH variable to find programs.
Programs in the $PATH can be found by name:

- `blastp -help`
- `echo $PATH`

```
.:/home/wrp/bin:/seqprg/bin:/usr/NX/bin:/usr/kerber
os/bin:/usr/local/bin:/bin:/usr/bin
```

9

UNIX host status

- `top/ps` – what processes/apps are running
- `kill` – force-quit running processes/apps

- `df -h` – available disk resources
- `du` – disk space usage

10

other UNIX commands

- `builtins` – list available shell commands
- `which/where` – find path of commands
- `time` – measure how long something take
- `echo/tee` – print/report text
- `wget/curl` – download files
- `gzip/gunzip/bunzip/zcat` – compressed files
- `ssh/scp` – login/copy to/from remote hosts
- `history` – what have I done previously
- `man` – get help

11

redirection, pipes, replacements

- > - redirect `stdout` into file, replace existing
- >> - redirect `stdout` into file, appending
- | - redirect/pipe stdout to stdin of next command
- `` `backticks` `` - replace with captured `stdout`

12

UNIX editors: learn (at least) one

- `nano`
  - simple, easy
  - no mouse, use arrow keys
  - how to quit: ctrl-X (all commands at screen bottom)
- `emacs`
  - not so simple to use
  - incredibly versatile, customizable, programmable
  - how to quit: ctrl-X ctrl-C
- `vi`
  - not so simple to use
  - guaranteed to be on any UNIX machine
  - often the default `$EDITOR`
  - how to quit: [colon]q![enter]

13

Beginning emacs

```
sh> emacs
^x^c exit
sh> emacs filename
type some stuff
^f,^b,^p,^n forwd,back char, prev, next
  line
^x^s      save it
^x^c exit
sh>
```

14

Intermediate emacs

```
sh> emacs myscript.py
^s, ^r search forward, reverse
^a, ^e start, end of line
 esc = M-
M-<, M-> start, end of buffer
M-%   query-replace
^k kill-line (and put in kill buffer)
^k^k  delete line and linefeed (EOL)
^y (yank – insert kill buffer)
^x 2, ^x 1, ^x o (multiple windows)
^u    (repeat number)
^h (help,^h-t tutorial, ^h-a apropos)
```

15

Transferring Files

- Always initiate transfer from desktop machine (franklin.achs.virginia.edu has a "known" name and address, your laptop does not)
- MacOS:
  - open terminal
  - cd to directory with data file

  `scp file.data your_id@franklin.achs.virginia.edu:~/bioinfo/`
- Windows:
  - download and use "SecureFX" (menu driven)

Download a set of accessions from www.uniprot.org (one per line) and transfer to franklin.achs

16

(bash) shell scripts

- files ending with .sh suffix
- shebang: `#!/bin/bash` or `#!/bin/sh`
- useful to capture (potentially long) history of UNIX commands into a reproducible analysis
  - you will always need to repeat your analysis
  - you will never remember all the necessary steps
- with some modification, your script can be made generic, and reusable for other data

17

Downloading sequences (from the command line)

```
Uniprot – use accession: P09488  (not GSTM1_HUMAN)

curl http://www.uniprot.org/uniprot/P09488.fasta

>sp|P09488|GSTM1_HUMAN Glutathione S-transferase Mu 1 OS=Homo sapiens
GN=GSTM1 PE=1 SV=3
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL
PYLIDGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEF
EKLKPKYLEELPEKLKLYSEFLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPN
LKDFISRFEGLEKISAYMKSSRFLPRPVFSKMAVWGNK
```

18

shell scripts are commands

- shell scripts can simply be copies of commands you have run:

```
curl http://www.uniprot.org/uniprot/P09488.fasta > gstm1_hs.fa
curl http://www.uniprot.org/uniprot/P28161.fasta > gstm2_hs.fa
curl http://www.uniprot.org/uniprot/P21266.fasta > gstm3_hs.fa
curl http://www.uniprot.org/uniprot/Q03013.fasta > gstm4_hs.fa
curl http://www.uniprot.org/uniprot/P46439.fasta > gstm5_hs.fa
```

- if `download_uniprot_gstm.sh` contains those five lines, you get the same result:

```
sh download_uniprot_gstm.sh
```

  - what would happen if you did not send the "`curl`" output to a specific file name?
  - how would you put all these sequences in one file?

19

control flow statements

- `for name in` *`[…]`* `; do` *`[…]`* `; done`
  - do something for each item in a list
- `if` *`[…]`* `; then` *`[…]`* `;`
  `elif` *`[…];`* `then` *`[…];`*
  `else` *`[…]`* `fi`
  - specify behavior depending on conditions
- variables (name) and loops (for) reduce typing:

```
sh $ for acc in P09488 P28161 P21266 Q03013 P46439 ;
> do curl http://www.uniprot.org/uniprot/${acc}.fasta;
> done
sh $ for acc in `cat gstm.accs`
> do curl http://www.uniprot.org/uniprot/${acc}.fasta;
> done
```

backtics

20

alternative scripting languages

- Perl
  - once the mainstay of WWW/CGI programming
  - long history == lots of reusable packages
- Python
  - extremely popular (used in this class, ?easier? to learn)
- …

21

Exercises:

1. Open the Mac “terminal” app (`/Applications/Utilities/terminal.app`)
2. Create a directory: `ecg (mkdir ecg; cd ecg)`
3. in that directory, create a file (gst.accs) containing Uniprot GST accs
4. edit the file and display it
5. Use the "curl" command to download a sequence
6. write a file of shell (bash) commands to download those sequences the sequences from “gst.accs” from Uniprot

22