Bioinformatics and Functional Genomics
Unix at the command line
biol4230 Friday, Jan 19, 2018

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 line to shell script
- downloading sequences using 'curl'

To learn more:

- 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)

Exercises (today):

1. login to interactive.hpc.virginia.edu
2. create a directory for the course ("biol4230")
3. in that directory, create a file
4. edit the file and display it
5. Homework due Monday, 22-Jan-2018 (end of handout)

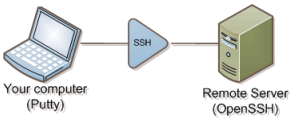
Computing environments

- 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 (Perl, Python, etc.)
  - interoperation between tools only limited by your ability to glue together input/output formats
  - almost entirely free access to tools

UNIX concepts

- Linux (RedHat/Centos, Ubuntu, …), AIX, Solaris, & Mac
- shells: `sh, bash, csh, tcsh, zsh, ksh, ...`
- commands: `ls, cd, more, cat, echo, ...`
- flags and arguments: `ls -l`, or: `cd ~`
- inputs and outputs: `stdin, stdout, stderr`
- redirecting input/output from/to a file `'<from', '>to'`
- piping output/input between commands `'|'`
- environmental variables: `$PATH, $PWD`, etc.
- shebang (`#!`) scripts

Using `interactive.hpc.virginia.edu`

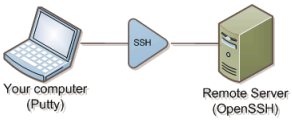


On MacOS:

1. Open "terminal"
2. `ssh your_its_id@interactive.hpc.virginia.edu`

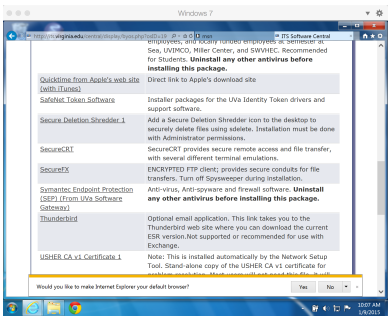
5

Using `interactive.hpc.virginia.edu`



On Windows:

1. Download `"SecureCRT"` from `www.its.virginia.edu/central;` install
2. connect to `interactive.hpc.virginia.edu`

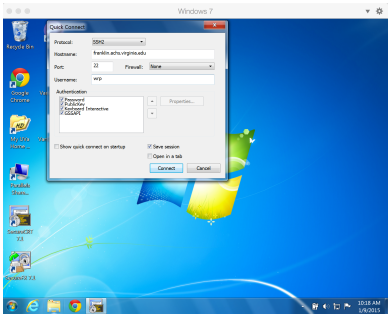


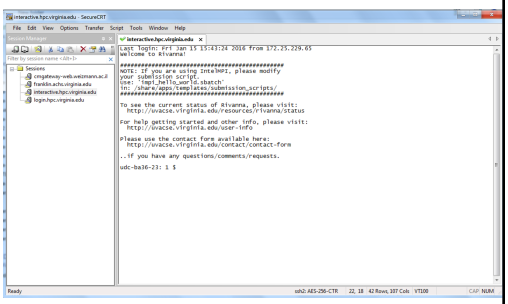
Using `interactive.hpc.virginia.edu`

On Windows:

1. Download `"SecureCRT"` from `www.its.virginia.edu/central;` install
2. connect to `interactive.hpc.virginia.edu`
   (in options/session, terminal=xterm, no delete/backspace mapping)

Windows SecureCRT





Using `interactive.hpc.virginia.edu`

reset eservices password: its.virginia.edu/accounts/changelost.html

After logging in:

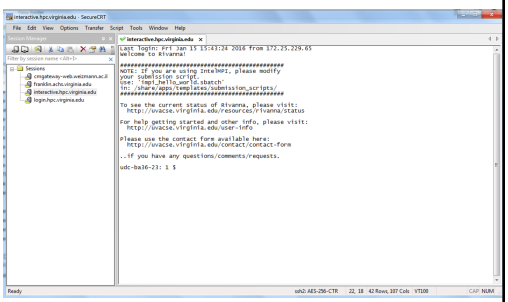
1. to see current location: `pwd`
2. to list files in directory: `ls`
3. logout: `^D` (ctrl-D) or "`exit`"

^C (ctrl-C) for emergencies

MacOS terminal

Windows SecureCRT

```
Last login: Fri Jan 20 08:31:52 on ttys000
d-172-25-36-58: 1 $ ssh interactive.hpc.virginia.edu
udc-ba33-37: 1 $ ls
aat                 fasta-sse2-jarpith.tar.gz   nup
aat.tar             fasta_www1103.tar.Z         papers
bib                 fast_pand                   prodna_lib.c
bin                 gbindex                     prodna_x.c
bioch               ht_password                 psisearch.tar.gz
bioch_old           inactive_src                public_html
biol4230            intel                       random10K.lib
blast2              jones-pam                   random.lib
blast95j            jones-pam.msg               reprints
blast_pan           lav                         RMAIL
blast_pand          lib                         save
bmg_web_update.sh   mail                        scripts
bug_seq             Mail                        seq
clustalw            md3000_install.log          seq.tar
config.c            md3000_install.log~         seq
CRP.pm              mips                        sequence.aa
data                mmap.info                   sgi
data_red            module_load.sh              sjxu.addr
dayhoff             mrtrans                     src
Desktop             ms                          tmp
Devel               msa2.0                      tmp~
```



UNIX file editors – learn one!!

- Use a UNIX editor on UNIX files:
  - `nano` – simple, menu-driven
  - `emacs` vs. `vi`/`vim`
    - powerful, memorize commands
  - do not use: Word, NotePad/WordPad, TextEdit, etc. (they include special, invisible, formatting characters that break programs and scripts)
- every editor has pros and cons (focus on `nano` and `emacs` if starting out)
- UNIX newlines are "`\n`"
  - PC is "`\r\n`"; Mac is "`\r`" (sometimes, but often '`\n`')

file system navigation

- UNIX filenames are "case-sensitive"

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

  - lower case only, only "`a-z_0-9`" (avoid '/', '[]*%$')
  - NO spaces ' ' (they are allowed in filenames, but break commands)
- `ls` – list files
- `cd` – change directory
- `pwd` – print working directory (current dir.)
- `pushd`/`popd` – `cd`, but remember directory in stack
- `basename`/`dirname` – extract filename pieces (also `new_file = ${old_file%%.*}` for `basename` with wild cards).

Practical Computing, Ch. 4

file system manipulation

- `ls` – list files
- `cp` *`from to`* – copy files
- `mv` *`from to`* – move files
- `rm` *`old_file`* – remove files

- `mkdir` *`dir`* – make a new, empty directory
- `cd` *`dir`* – change to directory
- `rmdir` *`dir`* – remove directory

Practical Computing, Ch. 4

file inspection

- `more` – read/browse through a file/stdin
- `head`/`tail` – print first/last N lines
- `diff` – report differences between files
  - useful for sorted lists that are slightly different
- `sort` – sort the lines in the file
- `uniq` – report unique lines
- `cut` – extract specific columns (`cut -f 1`)
- `cat` – dump file contents to stdout
- `grep` – search for matching lines
- `wc` – count words/lines/characters
- `od` – look at the raw data

Practical Computing, Ch. 4

Unix Permissions
(ignore until you cannot read/execute something)

- `chmod` – change permissions on file/dir
  - `u/g/o` – user (you), group, others
  - `r/w/x` – read/write/execute (look inside for directories)
  - `chmod +r data.file` – let others read the file
  - `chmod +x program.pl` – make script "eXecutable", directory searchable
  - `chmod -R go+r .` – recursively let others read files below "." (your current directory)
- `chown` – change owner of file (rarely used)
- `chgrp` – change group of file (also rare)

Unix host (machine) status

- `top/ps` – what is running, how long, how much memory
  - `ps -fu wrp` – tells what programs wrp is running

```
UID        PID  PPID  C STIME TTY          TIME CMD
wrp      21128 21126  0 11:19 ?        00:00:01 sshd: wrp@pts/0
wrp      21129 21128  0 11:19 pts/0    00:00:00 -tcsh
wrp      23615 21496  0 14:54 pts/2    00:00:00 ps -fu wrp
```

- `kill` – force quit a process

```
kill -9 21129
```

  - `-3` -> `^c`, interrupt, `-9` -> nuclear option

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

other UNIX commands

- `ssh/scp` – login/copy to/from remote hosts
- `wget/curl` – download files
- `man` – get help
- `history` – what have I done previously

- `builtins` – list available shell commands
- `which/where` – find path of commands
- `time` – measure how long something take
- `echo/tee` – print/report text
- `gzip/gunzip/bunzip/zcat` – compressed files

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`

globs (referring to more than one file)

- `*` wildcard matching
- `{a,b,c}` multiple choice
- `[a-c],[1-5,9]` range/set choice
- `^` negation

- `ls -l *.bam`
- `ls chr[1-23,X,Y].bed`

environment variables

- `$PATH` – where the shell will go to look for commands
- `$EDITOR` – your default editor
- `$USER` – who you are
- `$SHELL` – what shell you are running
- `$PWD` – your current working dir

- set in your `.bashrc/.cshrc`, `export/setenv`

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/vim**
  - not so simple to use
  - guaranteed to be on any UNIX machine
  - often the default `$EDITOR`
  - how to quit: ?[esc-key] [colon]q![enter]

Beginning emacs

```
bash $ emacs # ^ is ctrl-key, hold down like
  shift
^x^c exit
bash $ emacs filename
type some stuff
^f,^b,^p,^n forwd,back char, prev, next line
^x^s       save it
^x^c exit
bash $
```

Intermediate emacs

```
bash $ emacs get_protein.sh
curl http://www.uniprot.org/uniprot/P09488.fasta
^x^s  -- save it
^x^c  -- exit
# non-ctrl-text simply goes into the file
^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)
^k^k^y^y duplicates a line, each addnl ^y makes
   another duplicate
^x 2, ^x 1, ^x o (multiple windows)
^u     (repeat number)
   ^u10^d - delete 10 chars
    ^u5^y - "yank" five lines
^h (help,^h-t tutorial, ^h-a apropos)
```


Transferring Files (for homework)

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

```
scp file.data your_id@interactive.hpc.virginia.edu:~/biol4230/
```

    - transfer from `interactive.hpc`

```
scp your_id@interactive.hpc.virginia.edu:~/biol4230/file.data .
```

This "." is essential

- Windows:
  - download and use "SecureFX" (menu driven)

Download a set of accessions from www.uniprot.org (one per line) and transfer to `interactive.hpc`

(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

Scripting from the WWW: `wget/curl`

- Most bioinformatic analyses require resources from the web, e.g. sequences, domain information, datasets, etc.
  - The NCBI and EBI resources are usually scriptable; e.g. write a script that takes a set of accessions from a file and get the sequences
  - Often all that is required is to recognize the URL of the information desired

    `https://www.uniprot.org/uniprot/P09488`

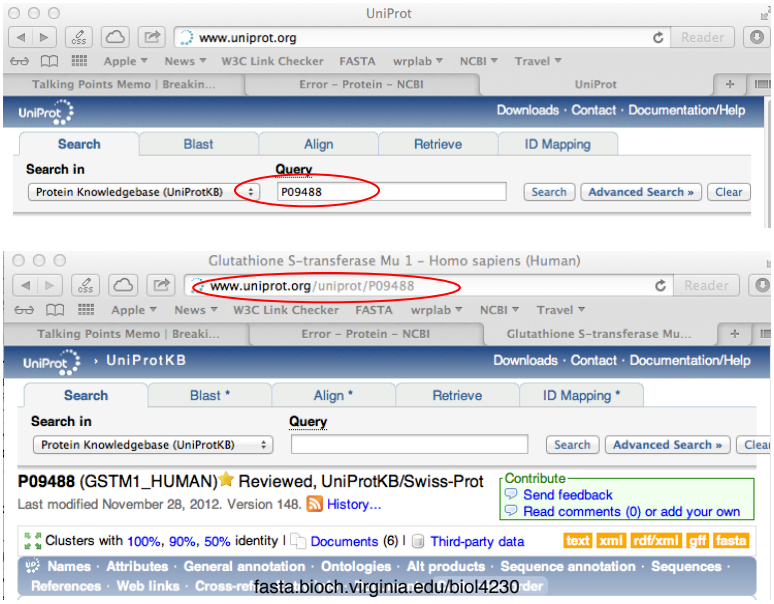
`curl` and `wget` allow you to pull a web page into a file from the command line:

```
curl https://www.uniprot.org/uniprot/P09488.fasta >p09488.fasta
```

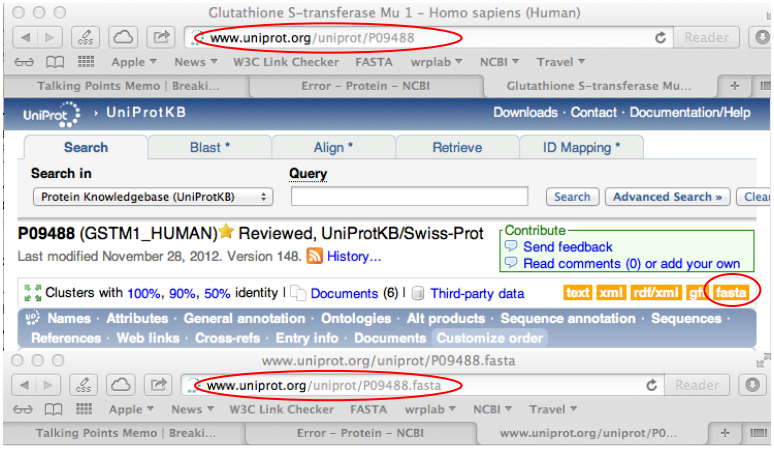
- `curl` needs `"http://", "https://" or "ftp://"`
- Sometimes this is all you need (uniprot); other times more work is required (NCBI)

Practical Computing, Ch. 5, p. 78

Finding a URL (www.uniprot.org)



Finding a URL to download (uniprot)




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

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

Finding a URL to download (NCBI)

curl fails with NCBI downloads

```
curl 'https://www.ncbi.nlm.nih.gov/protein/P09488.3?report=fasta'
```

fails producing lots of javascript but no data

```
curl 'https://www.ncbi...gov/protein/121735?report=fasta&format=text'
```

Also fails

But the way NCBI wants you to do it is:

```
curl "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/
efetch.fcgi?db=protein&id=P09488&rettype=fasta&retmode=text" (all on one line)
# "quotes" required to prevent shell interpretation of & and ?
>P09488.3 RecName: Full=Glutathione S-transferase Mu 1 ...
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNLPYLIDGAHKI
TQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEFEKLKPKYLEELPEKLKLYSE
FLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPNLKDFISRFEGLEKISAYMKSSRFLPRPVFS
KMAVWGNK
```

This is new; NCBI now uses accessions, not GI numbers

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?

Practical Computing, Ch. 6

control flow statements

- `for name in […]; do […] ; done` learn to do this
  - do something for each item in a list
- `if ( … ) ; then […] ; elif ( … ); then […]; else ( … ) fi` better to use a python script
  - 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

To learn bash scripting, try things out

```
sh$ for n in 1 2 3 4 5;
> do
> echo $n
> done
1
2
3
4
5
sh$
```

```
sh$ for n in `cat gstm_hs.accs_v`; do
> echo $n; done
P09488.1
P28161.2
P21266.3
Q03013.1
P46439.2
sh$
```

```
sh$ for n in `cat gstm_hs.accs`
> do
> echo $n
> done
P09488
P28161
P21266
Q03013
P46439
```

```
sh$ for n in `cat gstm_hs.accs_v`; do
> echo ${n%.*};  done
P09488
P28161
P21266
Q03013
P46439
```


Lab work (today)

1. Get an ITS unix account
2. On your computer, login to your account on `interactive.hpc.virginia.edu`
   - Windows: download/install SecureCRT
   - Mac: open terminal

     `slogin unix-id@interactive.hpc.virginia.edu`

   Outside UVA, you MUST use CISCO AnyConnect
3. create a file containing your `$PATH`

   `echo $PATH > path.file`
4. list the contents of the file
5. Make a copy of the file
6. Make a sub-directory (folder) called "`biol4230`"
7. Move the path.copy file into the `biol4230` folder
8. List the contents of the `data` folder
   - save the contents of the `data` folder to a file

Homework (due Monday, Jan. 22, 12 noon)

1. Download a protein sequence from the NCBI (`www.ncbi.nlm.nih.gov`) in FASTA format (try `GSTM1_HUMAN or NP_000552`)
2. Transfer the sequence file from your computer to `interactive.hpc.virginia.edu` using scp (Mac) or SecureFX.
3. Move the transferred file into the "hw1" folder in the "`biol4230/`" folder (directory)
4. From the NCBI, download the accessions for Human RefSeq glutathione transferases to a file on your laptop;
   - search for: "`glutathione S-transferase AND human[orgn] AND srcdb_refseq[prop]`"
   - Use the "send to" link to download the **`accession list`** and then transfer the **`accession list`** to interactive.hpc
   - also download from the NCBI the **`accession list`** for human glutathione transferases from SwissProt (`srcdb_swiss_prot[prop]`)
5. Use the "grep" command to identify the refseq accessions beginning with NP_, and put them in a separate file (gst_refseq.NP_only)

Homework (due Monday, Jan. 22, 12 noon)

6. Use the shell script: `~wrp/biol4230/hwk1/get_ncbi_acc.sh` to transfer the sequences referred to by the RefSeq accessions to a single glutathione transferase library file (hint, use ">" to send results to a file).

   `~wrp/biol4230/hwk1/get_ncbi_acc.sh your_refseq_acc_file > gstm.library`
7. From the Uniprot web site, identify a set of Human reviewed (SwissProt), glutathione S-transferases. Use the "download" link to download the list of accessions
8. NCBI SwissProt accessions have version numbers, while accessions downloaded from the UniProt site do not. Use the script: `~wrp/biol4230/hwk1/no_version.sh` to remove the version numbers from the NCBI SwissProt Accessions and generate a file of accessions without version numbers.

   `~wrp/biol4230/hwk1/no_version.sh your_refseq_acc_file > refseq_accs.no_version`
9. Write a shell script to compare the list of NCBI SwissProt human GST's with the Uniprot SwissProt human GSTs by:
   a. sorting both files using the "`sort`" program, generating two sorted files:`ncbi_swiss.srt` and `uniprot_swiss.srt`

      ```
      sort ncbi_swiss.no_version > ncbi_swiss.srt
      sort uniprot_swiss.accs > uniprot_swiss.srt
      ```

   a. compare the two files using "diff": `diff ncbi_swiss.srt uniprot_swiss.srt` and save the results to a file
10. Pick one sequence that is in the NCBI:SwissProt list but not in the UniProt list, and one in the Uniprot list but not the NCBI list, and suggest a reason why each sequence is missing from the other list (name the accession and the explanation).

Describe programs (shell scripts) and put answers in:
`biol4230/hwk1/hwk1.notes`

Homework (due Monday, Jan. 22, 12 noon)

Put all the scripts and results files in a directory on interactive.hpc called "`biol4230/hwk1`" and be certain that I can read it.

```
cd                 # go to home directory
chmod go+rx .      # make it readable by others
chmod go+rx biol4230/  # make biol4230 readable
chmod go+rx biol4230/hwk1  # and hwk1
chmod go+r biol4230/hwk1/* # and files in hwk1
```