UNIX Commands

You have accounts on Gordon

Use ssh to log in to gordon using the username you created with XSEDE:

```
$ ssh phage@gordon.sdsc.edu
The authenticity of host 'gordon.sdsc.edu (198.202.104.118)' can't be established.
RSA key fingerprint is 5e:47:dc:37:3e:2e:62:2a:31:57:41:a3:35:d1:82:22.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'gordon.sdsc.edu,198.202.104.118' (RSA) to the list of known hosts.
Password:
Last login: Thu Sep 12 14:42:53 2013 from kerner.ucsd.edu
Rocks 6.1 (Emerald Boa)
Profile built 13:17 19-Aug-2013

Kickstarted 13:49 19-Aug-2013

                            WELCOME TO
_______________________________   _________            _________
__  ___/__  __ \_  ___/_  ____/   __  ____/___________________  /____________
_____ \__  / / /____ \_  /        _  / __ _  __ \_  ___/  __  /_  __ \_  __ \
____/ /_  /_/ /____/ // /___      / /_/ / / /_/ /  /   / /_/ / / /_/ /  / / /
/____/ /_____/ /____/ \____/      \____/  \____//_/    \__,_/  \____//_/ /_/

[phage@gordon-ln1 ~]$
```

Basic file system navigation

`pwd`

- **p**rint **w**orking **d**irectory

```
~$ pwd
/home/phage
```

`cd` *`path`*

- **c**hange **d**irectory
- changes the current working directory to *path.*

List directory contents:

`ls` *`path`*

- **lis**t
- displays the contents of *path*, or the current working directory if none is specified

`ls -l`

- long format (including sizes and permissions)

`ls -a`

- show hidden files that start with '.'

`ls -lhtr`

- long, “human readable” sizes, sort by most recent first

File manipulation

`cp` *`a b`*

- **c**o**p**y
- copy *a* to *b* (files or directories)

`mv` *`a b`*

- **m**o**v**e
- moves *a* to *b* (files or directories). Note that moving a file is the same as renaming it.

`mkdir` *`directory`*

- **m**a**k**e **dir**ectory
- creates *directory*

`rm` *`file`*

- **r**e**m**ove
- deletes *file*

`rmdir` *`directory`*

- **r**e**m**ove **dir**ectory
- deletes *directory*. Note that it only works if the directory is empty.

`rm -r` *`directory`*

- recursively deletes *directory* and its contents

Viewing and creating text files

`cat` *`file`*

- con**cat**enate
- concatenates the contents of *file* and displays them to the screen (stdout)

`less` *`file`*

- displays the contents of *file* to the screen, one page at a time

Redirecting and piping

`cat > file`

- concatenates the input (stdin) and *redirects* it to a file (instead of stdout).
- '>' (and '<') is a redirection symbol.

echo me too >> *file*

- adds the line “me too” to the end of *file*.
- '>>' is a redirection symbol for appending.

`ls -lah * | less`

- list the contents of every directory in the current path, in long form, with human readable sizes and *pipe* the output to less so it displays one screen at a time.
- '|' is the pipe symbol. It redirects the output of the first command to the input of the second command

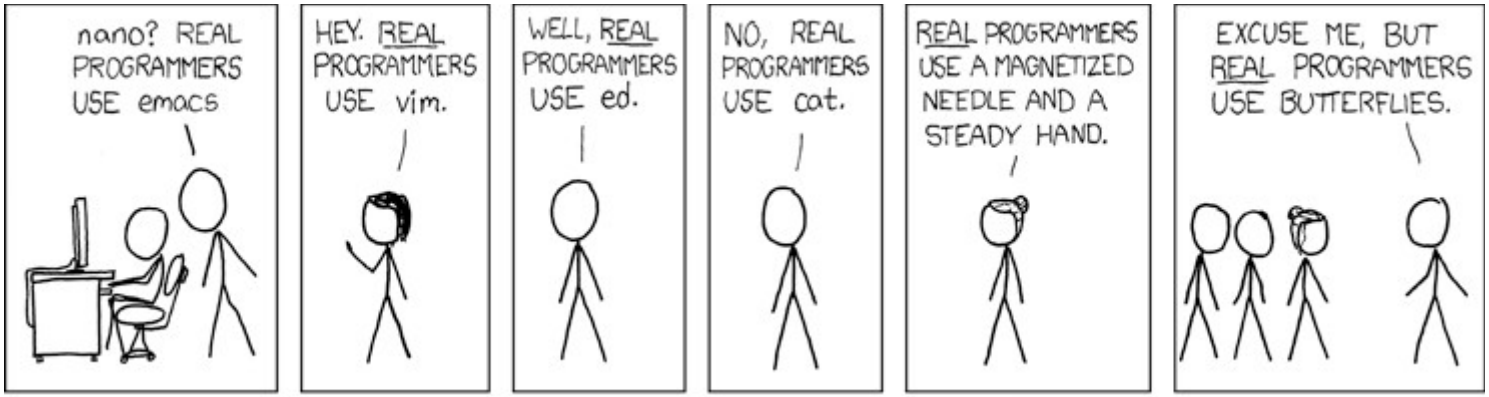
Redirecting and piping

- This technique is *extremely* powerful
- By chaining together the inputs and outputs of small programs you can accomplish just about anything with a UNIX command prompt.
- In fact, this is the core philosophy of the UNIX system tools:
  - small, reusable tools instead of large, monolithic programs.
  - e.g. *grep*, *sed*, *awk*, etc.

Text editing

- **Many choices: emacs vs. vim vs. ?**
- We'll use GNU nano since it's lightweight, easy to use and commonly installed (including on Gordon):

```
nano <filename>
```



http://xkcd.com/378/

Text editing

Some commands:

- Ctrl-K cut entire line
- Ctrl-U paste entire line
- Ctrl-W search
- Ctrl-\ search and replace
- Ctrl-R read file (and insert it at cursor)
- Ctrl-O write file
- Ctrl-X quit
- Ctrl-C cancel

Paths

`locate` *`pattern`*

- finds file matching *pattern* (doesn't search home directory)

`sudo updatedb`

- updates the 'locate' database

These two commands give you an error:

`ifconfig`

- tells you about your internet connections, including IP addresses

`which ifconfig`

- tells you the path to the command 'ifconfig'

WHY? because ifconfig is not in our path

```
$ locate ifconfig
/sbin/ifconfig
```

Option 1: Include full path in command

$ /sbin/ifconfig

Option 2:

A) Add these lines to ~/.bash_profile:

```
PATH=$PATH:/sbin/
export path
```

B) Exit and Login again, or run "source ~/.bash_profile"

But it won't let me!

`chmod 755 file`

- **ch**ange **mo**de
- sets the permissions of *file* to allow anyone to read / execute. Many other possible combinations.

`chown phage file`

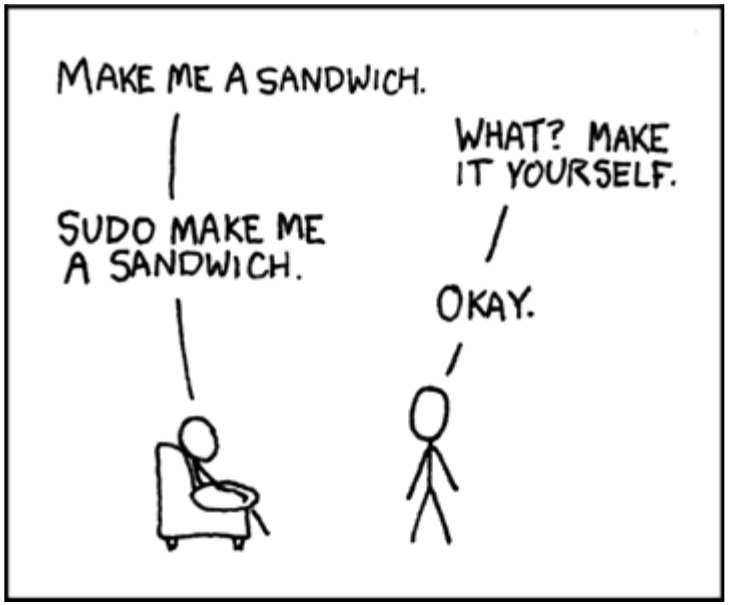
- set the owner of *file* to phage

`sudo command`

- **su**peruser **do**
- execute *command* as superuser / root
- "With great power there must also come . . .

  great responsiblity!"

  - Stan Lee



http://xkcd.com/149/

There's no room!

`du -hs` *`directory`*

- **d**isk **u**sage
- check how much space *directory* is taking up (in human readable form, with subtotals)

df -h

- report free disk space

Is it running?

top

ps aux | grep *phrase*

Is it running?

`ps aux`

- **p**rocesse**s**
- list all running processes with their pid

`ps aux | grep` *`process_name`*

- search list of processes for *process_name*

`kill` *`pid`*

- sends a TERM signal to *pid* (found with ps)

`top`

(also htop - fancy, pretty version)

- interactive program for viewing processes, cpu usage, memory, etc.
- and terminating them with extreme prejudice
- answers the commonly-asked question: “what's slowing my system to a crawl?”

If you do something by accident, maybe you can fix it ...

Ctrl-C quits / cancels things

Ctrl-Z puts things in background!!! oops

(⇨ fg ('fore-ground') allows you to restore)

Tar

tape **ar**chive

tar cvfz *archive_name path*

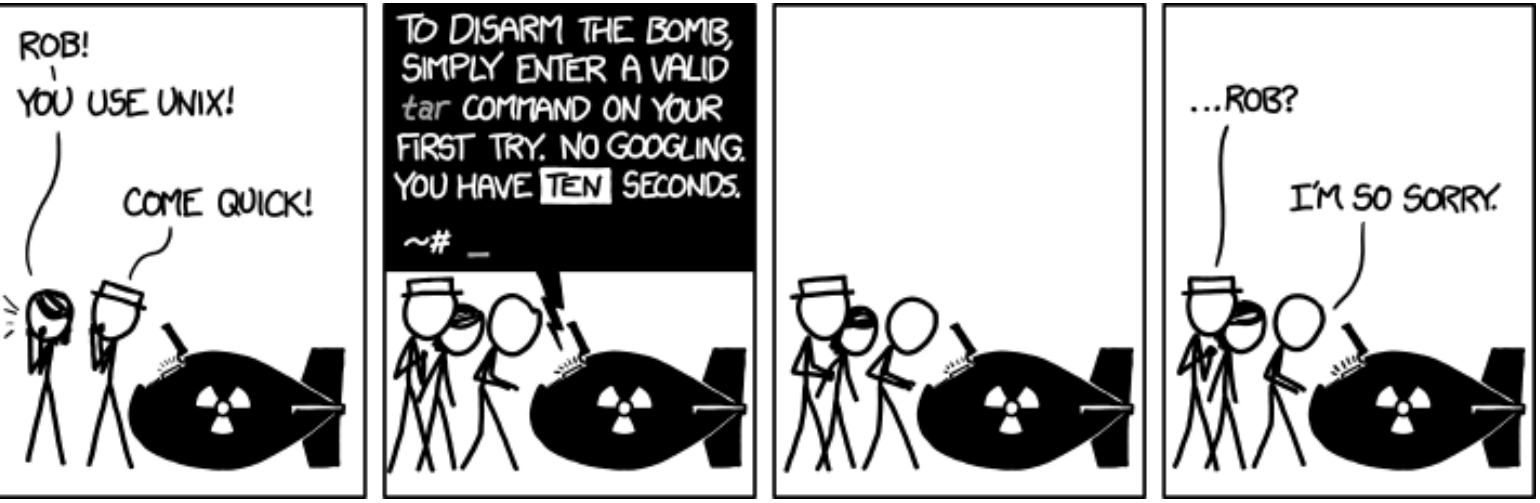
- **c**reate a gzip compressed archive with *archive_name* from the items in *path*

tar cvfj *archive_name path*

- **c**reate a bzip2 compressed archive with *archive_name* from the items in *path*

tar *x*vf *archive_name*

- e**x**tract the contents of *archive_name*



http://xkcd.com/149/

Installing stuff

sudo apt-get install

Installing from source: good luck!

sudo ./configure

sudo make

sudo make install

Other basic commands

`uname -a`

- display the OS and version

`ip addr`

- display IP addresswa

`ln -s` *`target link_name`*

- create a symbolic link (shortcut) named *link_name* to *target*

`wget` *`URL`*

- download whatever *URL* points to

`sftp` *`user@host:path`*

- securely download (over ssh) from a remote system

"Advanced" stuff

Now that we know our way around, how do we make use of Linux?

Run in the background / 'parallel'

`nohup` *`command`* `&`

- **no h**ang**up**
- run *command* in the background and keep running even after logging out

`screen`

- starts up a screen, which is like a new login
- to 'detach': Ctrl-a Ctrl-d

`screen -list`

- list available screens

`screen -r`

- re-attaches screen if only one exists

`screen -r` *`foo`*

- re-attaches screen matching *foo*

Shell scripting

You can put multiple commands together in one file.

For example, create a file **'test.sh'** that has these commands:

mkdir foo

mv foo bar

Then run:

```
bash test.sh
```

There should now be a directory called 'bar'

You can even write complex programs with flow control and data structures in shell scripts . . .

File manipulation

grep *pattern file* # ***global regular expression:*** *-n, -A, -B options helpful*

wc -l *file* # number of lines in file

cut -f 3 *file* # takes the 3rd column

cut -f 2-4 cut -f 3,5

head *file*

tail -n 40 *file* # last 40 lines

how many unique values in column 3 of file?

cut -f 3 *file* | sort | uniq | wc -l

cat a >> tmp

cat b >> tmp # concatenate files one after another

paste a b > c # puts the columns together (e.g. cbind in R)

https://xkcd.com/208/

AWK: File manipulation like a boss

 # general:

awk '/*regular expression/ { do stuff: default is to print}*'

awk -f *script_file* -F *field_separator* 'OFS="\t" BEGIN{a=0} { print $1}'

 # examples

awk '/>/' *file* # prints out headers only, from a FASTA

 # add 'Gene Length' to a .bed file

awk '{print $3-$2}' foo.bed > foo.geneLengths

paste foo.bed foo.geneLength > foo2.bed

 # Awk example 2: Print protein-coding genes that don't duplicate the same position as the previous one

 # file foo.bed:

```
Chr     Pos     Name
chr1    1234    NM_123
chr1    3465    NM_1234
chr1    3465    NM_456
chr3    3454    NR_345
chrX    2345    NM_578
```

```
# script.awk
BEGIN {chr=""; pos=0;}
      /NM/ && NR > 1 {
            if(!(chr== $1 && pos ==$2)) {print}
            chr=$1;pos=$2  }
```

awk -f script.awk foo.bed | less