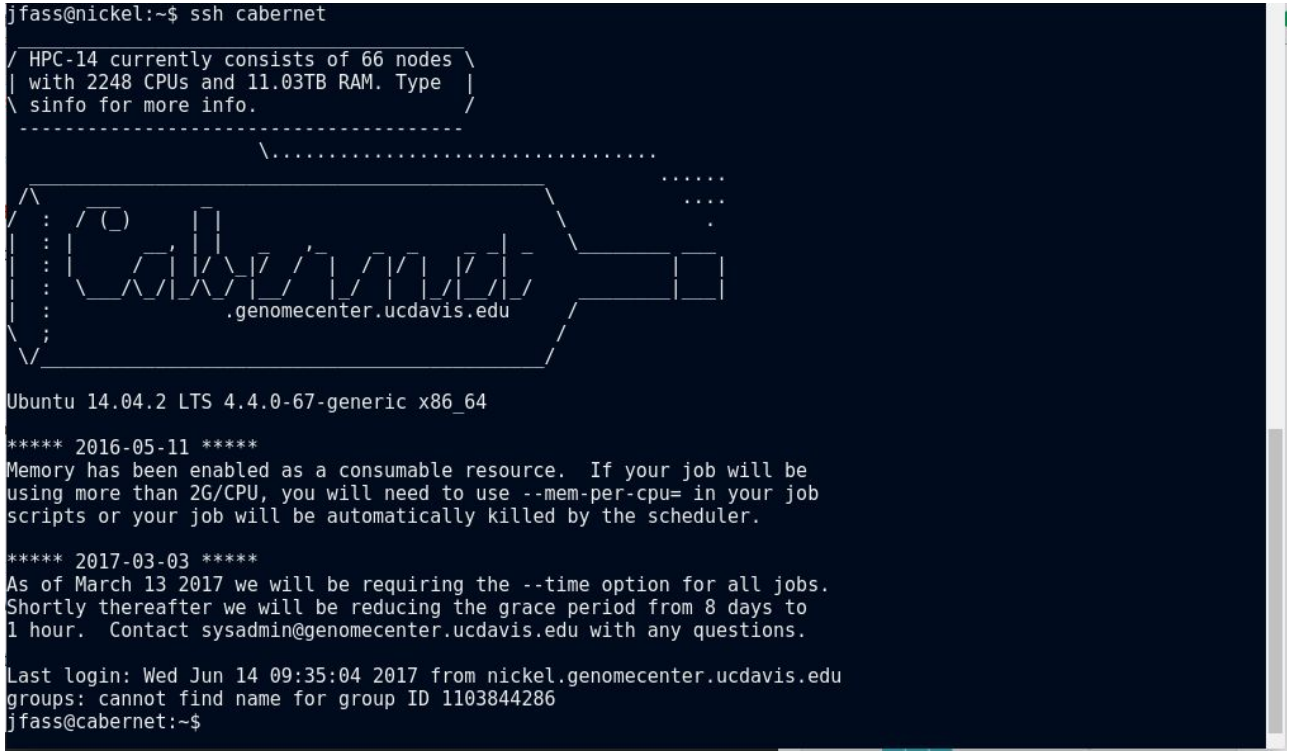
Intro to the Command Line

J Fass | 19 June 2017

The Terminal



The Terminal

```
jfass@nickel:~$ ssh cabernet
  ________________________________________________          ......
 /\    ___        _                                \           ....
/  :  / (_)      | |                                \            .
|  : |      __, | |   _   ,_    _  _    _ _| _   \_______ ___
|  : |     /  | |/ \_|/  /  |  / |/ |  |/  |                |   |
|  :  \___/\_/|_/\_/ |__/   |_/  |  |_/|__/|_/    ________|___|
|  :                .genomecenter.ucdavis.edu     /
\  ;                                              /
 \/_____________________________________________/
[...]

jfass@cabernet:~$
```

The Terminal

```
jfass@nickel:~$ ssh cabernet
  ________________________________________________          ......
 /\    ___        _                                \           ....
/  :  / (_)      | |                                \            .
|  : |      __, | |   _   ,_    _  _    _ _| _    \_______ ___
|  : |     /  | |/ \_|/  /  |  / |/ |  |/  |                |   |
|  :  \___/\_/|_/\_/ |__/   |_/  |  |_/|__/|_/    ________|___|
|  :               .genomecenter.ucdavis.edu     /
\  ;                                             /
 \/_____________________________________________/
[...]

jfass@cabernet:~$ <ctrl>-l
```

The Terminal

<ctrl>-l or -k …
Clears terminal, start at top

See also: reset

The Terminal

<-- prompt (includes $ and one space after)

Huge # of possible configurations; in this case:

<uname>@<hostname>:<pwd>$<space>

(pwd = **p**resent **w**orking **d**irectory)

Command Line Basics

```
jfass@cabernet:~$ <type command here>

jfass@cabernet:~$ pwd<enter>

/home/jfass
```

Follow command with <enter>

E.g. “pwd” … lists your **p**resent **w**orking **d**irectory

Command Line Basics - Getting Out!

```
jfass@cabernet:~$ wrong command <control-c>

jfass@cabernet:~$ # looks like this:

jfass@cabernet:~$ wrong command ^C

jfass@cabernet:~$

jfass@cabernet:~$ sleep 1000

^C

jfass@cabernet:~$
```

<control-c>: escape from entering a command …

… kill a running command (“sleep” actively counts off the specified number of seconds before letting you do anything else)

Command Line Basics - Getting Out!

```
jfass@cabernet:~$ R

[...]

> <control-d>

Save workspace image? [y/n/c]: n

jfass@cabernet:~$
```

<control-d> … escape from some interactive sessions (R, python, …)

(R is a powerful data-centered, statistical computing language)

Command Line Basics - Getting Out!

```
jfass@cabernet:~$ yes | more

[...]

<q>

jfass@cabernet:~$
```

q = **q**uit:

Escape from paginators! (less, man, etc.)

("yes" says "y" until killed … it's a dinosaur)

("|" is the pipe character … we'll explore it more soon)

("more" shows you a page of text, then waits for you to hit <space> to show another)

Command Line Basics - Getting Out!

```
jfass@cabernet:~$ exit
```

“exit” kills the current shell: the program that’s interpreting your commands for the operating system.

Command Line Basics - Where Am I?

```
jfass@cabernet:~$ ls

[...]

jfass@cabernet:~$ pwd

[...]
```

list file in the pwd

present **w**orking **d**irectory

Command Line Basics - Options!

```
jfass@cabernet:~$ ls -R

[...]

[...]

<control-c>
```

list recursively

What did I just do???

Command Line Basics - Read The Manual (RTM)!

```
jfass@cabernet:~$ man ls

[...]

<up, down arrows>

[...]

<q>
```

man <command> consults the **man**ual that exists for basic, OS commands. Any software author can write a “man page” for their software, but most scientific software authors don’t.

Command Line Basics - Options, options, options!

```
jfass@cabernet:~$ ls -l

jfass@cabernet:~$ ls -a

jfass@cabernet:~$ ls -l -a

jfass@cabernet:~$ ls -la

jfass@cabernet:~$ ls -ltrha
```

man ls …

Can combine single letter options …

list **a**ll files (in pwd), in **l**ong format, in **r**everse **t**ime order with **h**uman readable file sizes

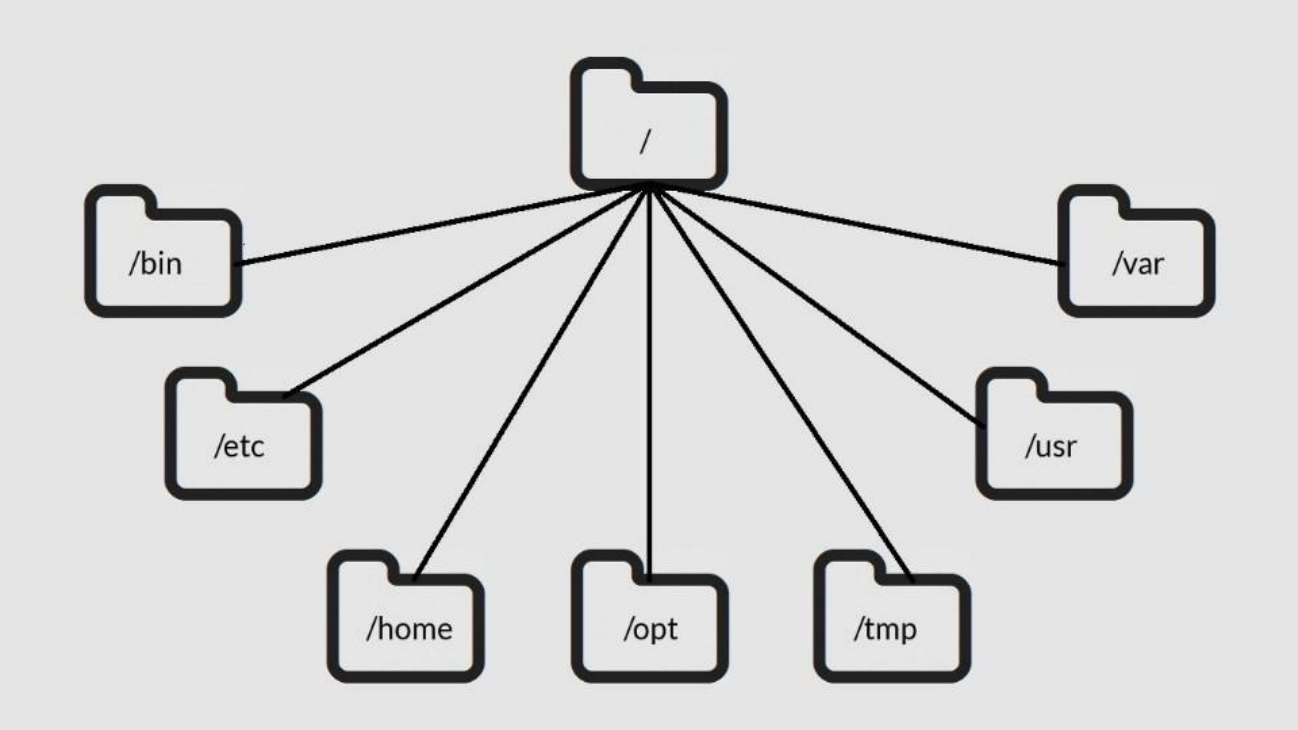
Command Line Basics - Directory Structure

```
jfass@cabernet:~$ ls -R

[...]

./R/x86_64-pc-linux-gnu-library/3.3/BH/include

[...]

<control-c>
```

‘/’ separates directories

Names can include many characters, but avoid spaces and other weird stuff.

Command Line Basics - Directory Structure



www.linuxtrainingacademy.com

Command Line Basics - ‘.’ and ‘..’

```
jfass@cabernet:~$ ls -a

.

..

.bash_history

[...]
```

“.” = pwd

“..” = up one level

Don’t be confused between use of “.” and filenames that start with “.” … the latter are valid filenames, that are just “hidden” unless you use the “ls” command’s “-a” option.

Command Line Basics - Absolute/Relative Address

```
jfass@cabernet:~$ ls /home/jfass/

[...]

jfass@cabernet:~$ ls ./

[same ...]

jfass@cabernet:~$ ls ../jfass/

[same ...]

jfass@cabernet:~$ ls ../jfass/../jfass/../jfass/

[yup, same ...]
```

‘.’ = pwd

‘..’ = up one level

Command Line Basics - <Tab> Completion

```
jfass@cabernet:~$ ls /home/jfas<tab>

jfass@cabernet:~$ ls /home/jfass/
```

<tab> will literally save your life. Hours of it.

A single <tab> auto-completes when it's possible (when only a single possible completion exists).

Command Line Basics - <Tab> Completion

```
jfass@cabernet:~$ ls /home/j<tab>

jfass@cabernet:~$ ls /home/j

jfass@cabernet:~$ ls /home/j<tab>

jacob/     [...]
jagadish/
jagomez/                                    jwbucha/

[...]
```

<tab> will literally save your life. Hours of it.

Two <tab>s in a row will show you all the possible completions, when there wasn't a *single* one for the single <tab> to use.

Command Line Basics - <Tab> Completion

```
jfass@cabernet:~$ ls /h<tab>

jfass@cabernet:~$ ls /home/<tab>

jfass@cabernet:~$ ls /home/<tab>

jfass@cabernet:~$ ls /home/j<tab>

jfass@cabernet:~$ ls /home/j<tab>

jfass@cabernet:~$ ls /home/jf<tab>

jfass@cabernet:~$ ls /home/jfass/
```

Use it!

Watch out for RSI ...

Command Line Basics - Create and Destroy

```
jfass@cabernet:~$ mkdir temp

jfass@cabernet:~$ cd temp/

jfass@cabernet:~/temp$ echo “Hello, world!” >
first.txt

jfass@cabernet:~/temp$ cat first.txt

jfass@cabernet:~/temp$ rm first.txt

jfass@cabernet:~/temp$ cd ../

jfass@cabernet:~$ rmdir temp
```

Create a directory

Change directories

Put text into a file

Con**cat**enate file to screen

Remove file

Up and out

Remove (empty) directory

Command Line Basics - Pipe and Redirect

```
jfass@cabernet:~$ mkdir CLB; cd CLB/

jfass@cabernet:~/CLB$ echo “first” > test.txt

jfass@cabernet:~/CLB$ echo “second” > test.txt

jfass@cabernet:~/CLB$ cat test.txt

jfass@cabernet:~/CLB$ echo “third” >> test.txt

jfass@cabernet:~/CLB$ cat test.txt
```

“>” redirects the output from one command to a file, instead of the screen.

“>” replaces

“>>” appends

Command Line Basics - Pipe and Redirect

```
jfass@cabernet:~/CLB$ cut -c 1-3 test.txt

jfass@cabernet:~/CLB$ cat test.txt | cut -c 1-3



jfass@cabernet:~/CLB$ cat test.txt > cut -c 1-3
```

“cut” cuts lines of text.

“|” pipes output from one command to be the input of another command.

“>” is wrong here … what does this command do?

Command Line Basics - Pipe and pipe and pipe …

```
jfass@cabernet:~/CLB$ cat test.txt

jfass@cabernet:~/CLB$ cat test.txt | cut -c1-3

jfass@cabernet:~/CLB$ cat test.txt | cut -c1-3 |
grep s
```

Pipes allow us to build up compound operations, filtering and changing data as we go.

“grep” searches for matches to regular expressions (patterns)

Command Line Basics - History

```
jfass@cabernet:~/CLB$ history

jfass@cabernet:~/CLB$ history | head

jfass@cabernet:~/CLB$ history | tail

jfass@cabernet:~/CLB$ history | tail -n 15

jfass@cabernet:~/CLB$ history | less
```

Since we often develop long commands through trial and error, it helps to see and access what we've done.

In "less," up and down arrows, pgUp, pgDn, and "q" to exit. Also, "/pattern" searches for pattern each <enter>. "n" and "N" for next and previous match, "g" and "G" for beginning and end of file / stream.

Command Line Basics - History

```
  558  cat test.txt | cut -c1-3 | grep s
  559  history
  560  history

jfass@cabernet:~/CLB$ !560
```

"!#" repeats command # from your history.

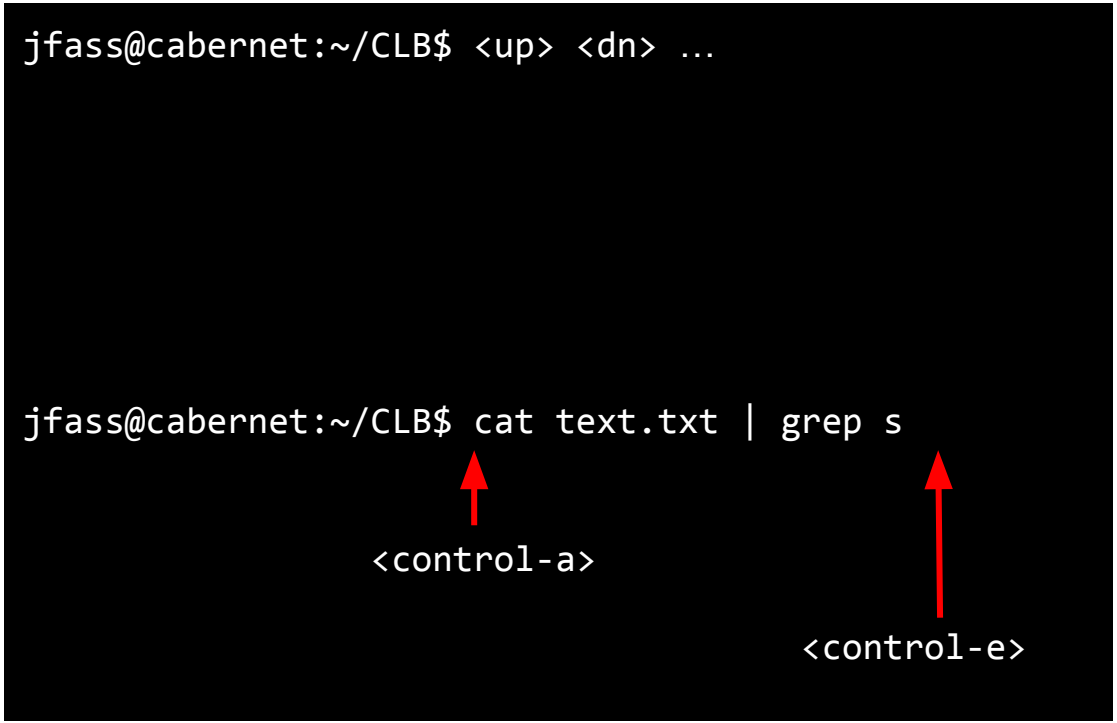
Command Line Basics - History Search

```
jfass@cabernet:~/CLB$ <control-r>

(reverse-i-search)`first': echo "first" > test.txt
```

<control-r>text triggers a recursive search for “text” from your history. After finding the most recent match that you like, use <control-r> again to get to an earlier match (and again, and …).

<enter> executes the command; left or right arrow fills the command line with the command but allows you to edit it before running it.

Command Line Basics - History Search



And, by the way, the up and down arrows take you backwards and forwards through your history of commands. Reach one you like, and start editing.

Also, by the way, <control-a> and <control-e> bring you to the beginning and end of your command.

Command Line Basics - Editing Commands

```
jfass@cabernet:~/CLB$ blah blah blah

jfass@cabernet:~/CLB$ blah blah <control-k>blah

jfass@cabernet:~/CLB$ blah blah

jfass@cabernet:~/CLB$ blah blah <control-w>

jfass@cabernet:~/CLB$ blah
```

Left arrow to before the last "blah," then <control-k> … kills text from here 'til the end of the line.

Now, <control-w> … kills *preceding* word.

Command Line Basics - Compression

```
jfass@cabernet:~/CLB$ gzip test.txt

jfass@cabernet:~/CLB$ file text.txt.gz

jfass@cabernet:~/CLB$ gunzip test.txt.gz

jfass@cabernet:~/CLB$ bzip2 test.txt; bunzip2
test.txt.bz2
```

Compress big files using “gzip,” “bzip2.” Bzip2 compresses smaller, but takes longer.

(“file” gives you info about the *type* of file you’re looking at)

Command Line Basics - Archives

```
jfass@cabernet:~/CLB$ wget
ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/PhiX/I
llumina/RTA/PhiX_Illumina_RTA.tar.gz

jfass@cabernet:~/CLB$ tar -xzvf
PhiX_Illumina_RTA.tar.gz
```

Large directory trees may be compressed as “tarballs” … see “tar.”

Let’s grab one and expand it.

Command Line Basics - Forced Removal

```
jfass@cabernet:~/CLB$ rm -rf PhiX
```

This is a dangerous one. Remove a file / directory, do it **r**ecursively to all sub-directories, and **f**orce removal (by-pass confirmation questions).

Caution is warranted. There's no Trash Bin, and no gauranteed way to recover deleted files.

Command Line Basics - Wildcard Characters

```
jfass@cabernet:~/CLB$ tar -xzvf
PhiX_Illumina_RTA.tar.gz

jfass@cabernet:~/CLB$ ls
PhiX/Illumina/RTA/Sequence/*/*.fa

[...]
```

Let's re-unarchive that tarball, to have something to look at.

List all files a few directories down that end in ".fa" …

Command Line Basics - Wildcards and Find

```
jfass@cabernet:~/CLB$ find . -name “*.fa”

[...]



jfass@cabernet:~/CLB$ find . -name “*.f?”
```

“*” can fill in for anything in a filename, *except* “/” … there’s a more appropriate command to use when you don’t know which directory level the files you’re looking for are at: “find”

“?” is like “*,” except only fills in for a single character.