AUTOMATED INITIATION

From MSI

On bootup (from ~)

bash msi\_start.sh

On bootdown (from ~)

bash msi\_leave.sh

SETUP STUFF

Get to Biller Lab Projects folder: /mnt/c/users/Molly/Desktop/Biller\ Lab/Projects

COMMON DIRECTORIES

/: root

~: home

/bin: basic programs

/dev: files connecting to devices like keyboard, mouse, screen

/etc: configuration files

/tmp: temporary files

/home/Molly: home directory ($HOME)

COMMON SYMBOLS

$ variable

$ this command must be executed in terminal

# this line is a comment

./ current directory

\t tab

| pipe (redirect output into a new command)

> redirect (redirect output into a file; will overwrite existing files)

>> append (redirect output and adds it to an existing file)

Wildcards

\* zero or more characters (no leading dot, so hidden files are ignored)

Example: ls \*.txt (returns directory contents that contain \_\_\_.txt)

? placeholder for any single character (no leading dot)

# comment

\n line break character in unix

\r\n line break character in windows

\ escape character; can change the meaning of the following character

-h or --help after command, tells you info about the command

SYNTAX

argument 🡪 command argument

multiple arguments 🡪 command argument1 argument2

option (eg: x) 🡪 command -x

option 🡪 command --option

multiple options (eg: x, y z) 🡪 command -xyz

a space in a directory 🡪 proceed the space with a back slash (\)

same line of code but show on another line 🡪 /\n

for loop 🡪

for [blank] in [blank]

do [blank]

done

bash script.sh $1 $2 $3 (ex: bash script.sh flatfish silversides sciaenids)

SHORTCUTS

tab: autocomplete

up/down arrows: navigate history of commands

ctrl+A: go to beginning of line

ctrl+E: go to end of line

ctrl+L: clear the screen

ctrl+U: clear line before cursor

ctrl+K: clear line after cursor

ctrl+C: kill the command that is currently running

ctrl+D: exit current shell

alt+F: move cursor fwd one word

alt+B: move cursor back one word

CREATE A SCRIPT

touch [filename.sh] 🡪 create script file

gedit [filename.sh] & 🡪 open file in text editor

FUNCTIONS

echo: print

\*echo $HOME: print location of home directory

cal: print calendar

man command: access manual of a command

\*q: close manual

unix man page [COMMAND]: search online for manual

cd: change directory

\*argument: which directory to change to

\*cd ..: move up one directory

\*cd /: move to root directory

\*cd ~: move to home directory

\*cd –: go back to the directory you previously visited (like back in browser)

pwd: print working directory

ls: list files and subdirectories in the current directory

\*ls -a: all files

\*ls -l: long list with details on permissions, user, owner, size, etc

First letter 🡪 d: directory, -:file

Second, third, and fourth letter 🡪 user permissions; r: read, w: write, x: execute

Fifth, sixth, seventh letter 🡪 group permissions

Rest of the letters 🡪 other people’s permissions

Includes 🡪 user, group, size, date

\*ls -lh: file size in human readable units

less: progressively print a file to the screen

\*option: -S makes it so it word wraps

\*ctrl+F: jump a screen fwd

\*ctrl+B: jump a screen bkd

\*q: quit

Chmod

\*change permissions

mv: move or rename a file/directory

\*arguments: thing to move, where to move it

\*or use the new and old file names to rename

touch: update last access of file, or create new file

rm: remove file

cat: concatenate and print files (turns the text into a text stream, brings into active memory)

\*argument: file name(s)

mkdir: make directory

\*option: -p = create nested directories (separate with /)

rmdir: remove directory

\*only works on empty directories

wc: line, word, and byte(character) count of a file

\*option: -l = line count only

sort: sort lines of a file and print result

\*options: -n = numerical, -r = reverse order, -k = sort by particular column

uniq: show only unique lines of a file

\*lines must be sorted first

\*option: -c count of occurance for each unique element

file: determine type of file

\*can be used to figure out what the line terminator character is

head: print the head (first few lines) of a file

\*option: -n = number of lines to print (follow n by a space and a number)

-Add a – or + before the number to do all besides for certain numbers of lines

tail: print the tail

\*tail -n +2: don’t include header

history: prints recent history of commands

\*pipe into tail for last 10, pipe into less to browse

diff: show the differences between two files

cp: copy from argument 1 to argument 2

\*Can add a slash behind argument 2 followed by a new name for the file

\*Add a -r option to copy everything recursively (not just the files but directories too)

\*Add -f option for force… don’t know why

Redirect text into file: command > filename

Append stdout to file: command >> filename

Redirect contents of file to stdin: command < filename

tr: translate

\*arguments: what to find, what to change it to

\*example: read in info from a file1, output changed into a file2

tr “,” “;” < [file1] > [file2]

| pipe output into an input for a new command

cut: grab rows

\*option: -d followed by “what separates rows”

\*option: -f followed by which columns to display

grep: find all lines of a file matching a given pattern

\*arguments: search term, file

\*can match one of many strings: “[string1]\|[string2]”

\*option: -c count lines containing match, -w match only full words, -i case insensitive

\*option: -B (followed by #) print # lines before match, -A print # lines after match

\*option: -n show line number of match, -v print all lines excluding matches

paste: adding columns (pasting columns together)

find: find files in the system

\*arguments: directory to search, search term

\*option: -name finds specific file name, -iname ignores case of file name

Can include wildcards to search for files containing search term

\*option: -maxdepth (followed by #) searches only dirs immediately below specified

\*option: -not excludes certain files

\*option: -type (followed by a letter) search for a certain file kind

d: directory

sudo: gives administrator privileges, goes in front of command

gedit: open file in a text editor

\*argument: which file to open

\*following it with “&” allows you to open the editor in the background

open: open file with default text editor

\*argument: file to open

\*following it with “&” allows you to open the editor in the background

bash: run a script

\*argument: script to run

\*can skip by bash by using chmod +rx [filename]

$PATH 🡪 path variable

\*Searches for corresponding executable files when command is invoked

\*Can append directory name to PATH by editing .bash\_profile

\*Add path to program to PATH variable: export PATH=$PATH:[PATH TO PROGRAM]

which: identify the path to a program

history: list the last commands you executed

time [COMMAND]: time the execution of a command

wget [URL]: download the webpage at [URL]

open: open file/directory with default program

\*xdg-open in ubuntu

\*start in windows git bash

rsync: synchronize files locally or remotely

tar: uncompress files and directories

zip: compress and package files and directories

awk: filtering based on content of column

\*just lines where the value of a column (columnnumber) is > value

-arguments: ‘ $columnnumber > value ‘ [file]

\*can do some math (ex: rows where column 5 is greater than 95 and column 6 is greater

than ninety percent of column 2

awk ‘ $5 > 95 && $6 > $2 \* 0.9 ‘ [file]

\*compounding

&& and

|| or

sed: complicated! Make changes to a file (ex: search and replace)

\*search and replace

-arguments: ‘s/lookingforthis/changingtothis/’ [file]

\*only changes the first occurrence in the line unless: ‘s/thing/thing/**g**’

xargs: pass a list of arguments to other commands

nano: open in text editor

\*argument: which file to open

$(): return a value

Does whatever is inside the parentheses and then assigns the output to variable

Changing the value of a variable

VARIABLE=1375

VARIABLE=$(echo 1375)

FILES=$(ls \*)

ACCESS CLUSTER

ssh -F $HOME/eofe-cluster/linux/config eofe8.mit.edu

\*from home directory

\*password: pro2cyano

Directories on cluster

/home/billerlab

- working space for source code, scripts,hand-edited files.

- This is backed up daily and has a quota of 100G

/nobackup1c/users/billerlab

- lustre parallel file system for parallel I/O.

- This has a 1TB quota and is not backed up.

/pool001/billerlab

- NFS file system if you need it.

- This has a 1TB quota and is not backed up.

Other commands

module avail: see list of available software

module add: add module (must redo each time you login or run a job; gcc/5.4.0)

sinfo: show which partitions you can submit to

squeue -u billerlab: check the status of your jobs in the queue

scontrol show job [<jobid>]: check the status of a specific job

Slurm scheduler

srun [<flags>] [commandname]

Example: This job is requesting 1x Centos7 node with 16 CPUs and 1x GPU for

one hour.

srun -N1 -n16 --time=1:00:00 --gres=gpu:1 --constraint=centos7 –

partition=sched\_mit\_hill ". /etc/profile.d/modules.sh;module add python/3.6.3; python3 mycode.py"

sbatch [script.sh]: submit batch jobs to the slurm scheduler

Example: This job is requesting 1x Centos7 node with 16 CPUs and 1x GPU for

one hour.

#!/bin/bash

#SBATCH -N 1

#SBATCH -n 16

#SBATCH --time=1:00:00

#SBATCH --gres=gpu:1

#SBATCH --constraint=centos7

#SBATCH --partition=sched\_mit\_hill

. /etc/profile.d/modules.sh

module add python/3.6.3

python3 mycode.py

Conda

\*Purpose: regulating and universalizing package installs

-Anaconda (larger, more complete) and miniconda (simpler)

\*Getting started:

-Download by finding the command for the proper version online

\*Commands

source path/.bashrc: activate conda (base)

conda deactivate: leave conda environment (leave conda I in base)

conda create: create a new conda environment

-n environment\_name: set name of environment to environment name

python=python\_version: set version o python to run in this environment

conda activate environment\_name: activate your new conda environment

conda env list: list existing conda environments (\*=current environment)

python –version: see what version o python is being run

conda env remove -n environment\_name: remove environment

\*can’t be inside it, so conda deactivate first

conda install package\_name: install a new package

-y don’t ask me for any confirmation

-c channel\_name: indicate which channel to install from

package\_name=version number: pick a particular version of the package

conda uninstall package\_name: uninstall package

conda search package\_name: search for all versions o a package in conda

-c channel\_name: indicate which channel to install from

conda install -c high -c med -c low pacakge\_name: set channel priority during install

conda config --add channels low; conda config --add channels med; conda config --add

channels high: set channel priority ahead of installation

conda config --get channels: see channel list and priority list

conda create -n env\_name -c high -c med -c low package python=version: create

environment, set channel priority, install package, set python version in one line