Unix shell use

Navigation commands

* pwd – print working directory
* ls – list directories
  + ls –F - ‘folder’:lists only the sub directories in that directory
  + ls –a - ‘all’ : shows all directories, including those that start with .
  + ls --help - tells you all the possible commands
* ls Desktop - tells you all the files in Desktop (assuming Desktop is a directory in where you are now)
* cd .. –go up a directory *to where you currently are*!
* ../ - refers to the directory up a level

Creating, removing and modifying files and directories

* mkdir dirname -dirname should have no spaces – use letters, numbers, numbers, dots, underscores
* rm filename.txt -remove or delete that file
  + rm –r dirname -to remove the contents of the directory, and the directory, recursively. Be careful! if you rm / -r, it will totally wipe your computer!!
  + rm –r –I dirname -goes through every file in that directory and asks you if you want to remove it before you remove it
* touch filename.txt -makes an empty file
* mv oldfilename.txt newfilename.txt -‘move’:renames the file
  + mv oldfilename.txt newfilename.txt –i -‘interactive’: asks you if you want to overwrite that newfilename
  + mv oldfile.txt dirname -moves the file to that directory
* cp oldfilename.txt newfilename.txt -‘copy’: makes a copy with that new name
  + cp oldfile.txt ../newfile.txt -makes a copy of that file in a directory up a level
* \*.txt -refers to all files in the current directory
* ? -any single character. Eg. P??? matches PATH, PANT, etc.

Pipes and Filters

* wc file.txt -‘word count’ – number of lines(-l), characters (-c) and words (-n)
  + wc file.txt > newfile.txt -writes the results to a new file
* firstcmd | secondcmd -pipe – redirects the output of the firstcmd to the secondcmd

Loops

* for variablename in directory
* do
* program $variablename
* echo line -makes it print the contents of that line, in a for loop
* done

Random programs

* head –n
* echo
* cat file.txt - prints contents of a text file to the screen

Name folders with “year-month-day”, with leading zeros

cntrl-C –kills

or :q

Git

setting up a git repository

1. mkdir directoryName
2. cd directoryName
3. git init
4. git status -check status–are there any uncommitted files?

tracking changes

* after you have made/changed a file….

1. git add filename.txt
   * adds things to a ‘staging area’
2. git commit –m “message about what version this is”
   * git commit –m “message” --author=”Name name <email>”
   * commits everything in the staging area to a version copy
   * Every time you want to save a change, you have to add and commit
   * when you make a change to that file, and save a new copy, git status will know that there is now a copy that hasn’t been committed. You must commit this new copy every time you save it

* git log -shows you a log of all the commits and the messages
* git diff -tells you the difference between current and last version

Viewing changes

* git diff HEAD~1 file.txt
  + To see only a change in a specific file at a specific time point, where 1 is the number of version files ago. If you leave out the ~1, it will show you the last change
  + git diff 94538sdefhj file.txt
    - can view a specific commit version, based on that commit code
* To view an old version:
  + git checkout HEAD~1 file.txt
  + cat file.txt
  + git checkout master file.txt -master refers to the most recent version
  + Be very careful – if you checkout a version without commiting your most recent version, you will lose this most recent version, even if you have saved it to disk!
* git log --patch file.txt - shows the log with version numbers, as well as the differences in each version