UNIX Command Line

. is the pwd .. is the parent directory of present directory

1. cd: cd (address)

* cd ~ takes you to the home directory
* cd .. takes you to parent directory
* cd – takes you to previous working directory

1. pwd: gives the address of the directory you are currently in
2. ls: ls -(flags) (directory on which you are doing ls)

* ls -a shows all hidden files
* ls -l shows all files in a list format
* ls -R lists files recursively (files in directory and the subdirectories, and then keep doing it over and over for the subdirectories)
* ls -X sorts the file alphabetically
* ls -S sorts the files by size
* ls -t sorts the files by time of creation; newest first
* ls -d lists directory but not their content

1. mkdir: mkdir (dirName) creates a new directory

* mkdir -m while creating the directory we can set the permissions for user, group and others
* mkdir -p in case you specify a whole address of the new directory and one of the parents doesn’t exist, the command creates the directory
* mkdir -v prints a message every time a new directory is created by mkdir
* Try something more complex as shown in class i. mkdir -p Music/{Jazz/Blues,Folk,Disco,Rock/{Gothic,Punk,Progressive},Classical/Baroque/Early}

1. echo:

* if you simply write echo “My bad”, the command will send back the message in the double quotes to the terminal itself.
* If you write this message in single quotes or in no quotes at all then the number of spaces in the message won’t be preserved meaning echo hello world and echo ‘hello world’ would display hello world.
* You can use backslash sequences like these to perform some function. To enable their interpretation, you need to add a -e flag before the message you wish to echo
* \b: backspace
* \c: produce no further output
* \n: enters the newline
* \f: form feed formatting
* \t is horizontal tab, \v is vertical tab
* You can use echo to redirect messages to a file; > to overwrite the destination file’s content and >> to append the message in echo to the destination file

1. Redirection > or >> or <

* There are three standard streams: std input (stdin, file descriptor is 0), std output (stdout, fd is 1) and std error (stderr, fd is 2)
* Command1 (>/>>) (file) : whatever output Command1 gives will be redirected into file you specified. If > was used the content of the file will be overwritten by the incoming text. If >> was used then the incoming text will just be appended into the existing content
* < is used to read in from a file

1. Concatenate: cat (text/fileName)

* Will display the contents of the file which you have specified in the command
* cat file1 file2 will display the concatenated content of both files on your terminal.
* -n: numbers lines OVERRIDDEN BY -b: numbers only non-empty lines
* -s: suppresses repeated empty lines
* -v: uses ^ and -M notation -E: uses $ notation -T: shows tabs t=-vT -e=-vE
* In case of a very long file you can use less command and navigate through the file using arrows

1. mv: move files from one location to other or rename them

* -v: displays message to show how is the file moving
* -i: prompt before action
* -n: do not overwrite an existing file
* -f: do not prompt before overwriting
* If you write mv fileName1 fileName2 and the second filename doesn’t exist, it renames the first file to the second file’s name

1. cp: copies files from one location to other

* -i: interactive, gives prompt before overwriting
* -p: preserves timestamps and ownership of the file

1. rm: remove file (permanently)

* -d: remove empty directories
* -r: remove directories and their content recursively
* -i: prompt before every deletion

1. rmdir: removes folder iff it is empty

* equivalent to rm -d
* -i: prompt before every removal
* -v: explain what is being done
* -p: removes directory and it’s ancestors