MM 2090 : Introduction to Scientific Computing

A List of useful Shell commands :

(written by Anirrudh Ramesh[a], [a]Dept. of Metallurgical and Materials Engg., IIT Madras)

1. Replacing a text from all the files in a directory and sub-directories

```
find ./ -type f -exec sed -i -e 's/apple/orange/g' {} \;
```

(This command replaces the string 'apple' by 'orange')

2. Changing all file names in a folder
   Just use bash, no need to call external commands.

```
for file in *.png
  do
    mv "$file" "${file/_h.png/_half.png}"
  done
```

For those who need that one-liner:

```
for file in *.png; do mv "$file" "${file/_h.png/_half.png}"; done
```

Ex: This command changes files with .png extension by changing '_h.png' with ' _half.png'

05_h.png
06_h.png
to
05_half.png
06_half.png

Ex:
```
for file in *.out ; do mv "$file" "${file/avg.out/avg_1.out}"; done
```

changes a file named "RVE_avg.out" to "Rve_avg_1.out"

3. Finds all the files in all the subdirectories with extension .ps and prints them

```
find . -name "*.ps" -print
```

4. Copies a file Anirrudh.f90 from directory 01 and pastes it to to all the directories (with 2 ...) in the directory one level up.

```
[skumara1@ghosh-login 01]$ find .. -type d -name "??" -print -exec cp Anirrudh.f90'{}' ';'
```

5. Obtaining the absolute path of the directory/file

```
find $(pwd)/ -type d -name "*.scheduler" -print > ./all_direct.txt
```

Finds all the directories with the name .scheduler and prints their absolute path to all_direct.txt

6. CONVERT dos formatted file to UNIX format

```
dos2unix -k -o filename
```

7. Opening a web page in shell

`explorer http:\\www.google.com` (or) `explorer http://www.google.com` (opens google.com in the system default browser)

explorer www.google.com open windows explorer

explorer opens windows explorer

8. Obtain the wifi password from windows cmd

`netsh wlan show profiles` --- shows all the wifi profiles in the system

Eg: AapKaWiFi

Detroit Airport Wi-Fi etc.

```
netsh wlan show profiles "Detroit Airport Wi-Fi" key=clear
netsh wlan show profiles AapKaWiFi key=clear
```

See the key content in the security settings displayed on the screen for the password (Only the currently connected wifi password can be displayed)

9. Linux kernel version

```
uname -a
```

10. Disk space remaining and used in linux

```
df -ah
```

11. Size of a directory without displaying all the contents

```
du -sh
```

12. Check for open ports on a linux machine

```
netstat
netstat -tulpn
```

13. Check CPU usage of a process

`ps aux | grep nginx` (or) `top`

14. Mounting a disk or USB

```
ls /mnt
mount /dev/sda2 /mnt
```

15. Check for existing mounts

```
mount
```

16. Faster way to transfer files

```
rsync -ru /source/directory/* username@domain.net:/destination/directory
```

17. Find a directory and cd into it

```
find ./ -type d -name "output" -print -exec bash -c "cd '{}'; exec bash" \;
```

This command searches directories named "output" and cd's into the first instance of such a Directory

18. Speed up the transfer with compression

A super cool option to speed up the transfer to save time and bandwidth.
All you need to do is use the C option to enable compression.
The files are compressed on the fly and decompressed on the destination

```
$ scp -vrC ~/anirrudh/@....:/media/C/
```

19. Preserving file attributes

```
$ scp -C -p ~/......
```

20. copying files using rsync

```
$ rsync -ru /source/directory/* username@domain.net:/destination/directory
```

21. grep for searching

`grep def *` (searches for all files which contain def string)
`grep "r..y" *` (searches for files which contain a string that starts with r and ends with y with exactly 2 characters in between)
`grep -c ruby *` (counts the number of occurrences of the string ruby in each of the all the files in the current directory)
`grep -n ruby readme.md` (prints the line number containing the string ruby in the file readme.md)

NOTE: By default, grep performs a case insensitive search. To perform case sensitive search we use -i flag as below.

```
grep -i ruby readme.md
```

Recursive Search (to search the current directory and all its sub-directories)

```
grep -R "ruby"
```

Modify the search and search only in .yml files in the cur directory and all its sub directories

```
grep -R --include="*.yml" "ruby"
```

22. Sorting files by size

`ls -lhS` (l for listing in long form: h for human readable form for size : S for sorting)
sorting by last modified time
`ls -lt` (l for long form listing: t for sorting by last modified time)
`ls -lr` (reverses the sort: i.e. oldest file will be displayed as the first file in the directory)

23. `cat fileName` (displays the contents of the file ‘fileName’ on the screen)

24. `ls -a ~ | grep _ | sed "s/_/-/g"` (This command performs three operations)

1. Lists all the files (including hidden) in the home directory (These are input to the next command)
2. Second command lists all the files/directories that contain an '_'
3. Third command replaces '_' with '-' in all the directories outputted from second command

<u>Writing to a file</u>
`ls -a ~ | grep _ > underscores.txt` (writes all file/directory names that contain an _ to underscores.txt file)
<u>Reading from a file</u>
we can use `<` to read from a file

25. **<u>Note:</u>** Ack and Ag are much faster than grep (grep is installed on every system. However Ack and Ag needs to be installed explicitly. Therefore,
usually people use grep)
(Speed: Ag>>Ack>grep)
(Refer to: http://conqueringthecommandline.com/book/ack_ag for more details)

26. Using Find to search (if we want to search for files using criteria other than the file's contents. For Ex.: filename,directory name, path name etc.)
`find . -name model.rb` (lists all the files with name model.rb in the cur directory)
`find . -name \*model.rb` (lists all the files whose name ends with model.rb. Here \ escapes *)
`find . -path \*session\*` (lists both directories and files whole path contains the term session)
./actionpack/test/dispatch/request/session_test.rb (a file)
./actionpack/lib/action_dispatch/middleware/session (a directory)
`find . -path \*session\* -type f` (finds only files)
`find . -path \*session\* -type d` (finds only directories)
And/or Expressions
`$ find . -path \*session\* -type f -name \*mem\*` (finds directories whose path contains session and files whose name contains mem)
./actionpack/lib/action_dispatch/middleware/session/mem_cache_store.rb
./actionpack/test/dispatch/session/mem_cache_store_test.rb
`$ find . \(-name \*.jpg -or -name \*.png\) -type f` (files whose extension is either .jpg or .png)
`$ find . -not -path \*t\* -type f` (files whose path does not contain the letter 't' anywhere in their path)
same as `find . \! -path \*t\* -type f`
`find ./guides -type f -name \*.yml -print -delete` (deletes all the files whose extension is .yml)

27. Using ps to know about the running processes
`ps` (lists processes that are being run in the current terminal (tab))
`ps u` (user-oriented output listing processes run by the current user (myself) in all tabs)
`ps -e` (display all processes by all users)
`ps u -e` (display all processes by all users in a user-oriented form)
`ps -U user` (display processes by user name)

`ps -O %mem` (for customizing the displayed properties of the processes)
`ps -m -O %mem -U user` (sort processes by memory usage)
`ps -r -O %cpu -U user` (sort processes by CPU usage)

28. **Using sed**

`sed "s/[aeiou]/*/" anirrudh.txt` (replaces the first vowel with * in each line of the file anirrudh.txt)
`sed "s/[aeiou]/*/g" anirrudh.txt` (replaces all the vowels with * in the file anirrudh.txt) (here g represents global: therefore all vowels are replaced)
`sed "s/cha/chi/2" anirrudh.txt` (replaces the second occurence of ‘cha’ in each line with ‘chi’)

All of the above replacements are displayed on the screen (standard output) (the entries in the files remain unchanged). We can change
the file entries "in-place" using -i flag

`sed -i "s/cha/chi/g" anirrudh.txt` (Now the contents of the file are changed and no output is displayed on the screen (standard output))
`sed -i.tmp "s/cha/chi/g" anirrudh.txt` (creates a backup file named anirrudh.txt.tmp with its contents being same as the contents of anirrudh.txt file before any changes are made.
We can use this file to verify the changes and if we are not satisfied with the changes, we can delete anirrudh.txt and rename the back-up file with anirrudh.txt and then re-do the editing)

`sed "s/happy/Merry/i" anirrudh.txt` (case-insensitive search. replaces all instances of the word ‘happy’ regardless of its case, to the word ‘Merry’) (output will be printed on the screen)
(again, -i is needed to change the file in-place)

`sed "/cha/s/./*/g" anirrudh.txt` (replaces every character in a line, that matches cha, with the character *)
`sed "s/[aeiou]/\u&/g" anirrudh.txt` (capitalizes all the vowels found)
here ‘&’ indicates a match found from the search part of the script
‘\u’ will convert whatever comes after it (here & - which indicates a match) to uppercase
likewise ‘\l’ will convert whatever that comes after it to lowercase

`sed -n "s/name/Mark/" anirrudh.txt` (will not print anything to the console because of the -n flag)
`sed -n "s/name/Mark/p" anirrudh.txt` (prints only one line of the file, that searches and replaces ‘name’ with ‘Mark’)
`sed -n "s/name/Mark/pw anirrudh.txt" anirrudh1.txt` (writes only the modified lines to a separate file named anirrudh1.txt)
`sed -n '1~2p' anirrudh.txt` (displays, starting at line 1 and prints every second line - can be helpful for excel sheets)
`sed "/name/d" anirrudh.txt` (deletes lines that contain the word ‘name’)

Conquering the Command Line:

$ sed 's/<[^>]*>//g' Example.html | sed '/^$/d' (removes HTML tags and the blank lines)
A more elegant way of doing this: (use ; to run multiple sed commands)
$ sed 's/<[^>]*>//g;/^$/d' example.html (removes HTML tags and the blank lines)

$@ = stores all the arguments in a list of string
$* = stores all the arguments as a single string
$# = stores the number of arguments

29. **Using the tar tool:**

tar -c Examples (creates a tar and displays the attributes such as file permissions,file's owner,group etc. to the screen) (here we can also pass a list of files to be put in the tar folder)
tar -cf Examples.tar Examples (creates a tar file)
tar -cvf Examples.tar Examples (verbose mode)
Note: We can pass any number of directories or files to the above tar command.
find . -name *.jpeg | tar -cvf jpeg.tar -T - (-T reads in a list of file names from another file. Here we use - in place of the
name of a file. This tells tar that it should read in the list of files from the find query
we piped in)

tar -tf foo.tar (lists the contents of the archived tar file foo.tar : we used -t flag for this)
tar -rvf foo.tar GemFile (append a file GenFile to the tar file foo.tar : we used -r flag for this)
tar -uvf foo.tar version.rb (updating the files that are already in the tar file foo.tar)
tar -xvf foo.tar (extracts all the files from the foo.tar file)
tar -xvf foo.bar rails.gemspec (extracts a specific file rails.gemspec from the foo.tar file)
tar -xvf foo.bar -C ./myfolder (extracts to a different folder. create the folder beforehand)
tar -cf foo.tar foo (compresses)
tar -czf foo.tar.gz foo (gzip - significant reduction in size: .gz is specified to indicate that that file is gzipped)
gzip (-z) : bzip2 (-j) : Unix compress (-Z)
most common : considerably slower : faster but produces larger archives
tar -xvzf ar.tar.gz (Extracting from compressed files : -z uncompresses and then -x extracts the files)

30. cat file1.txt file2.txt > file3.txt (combines two files and writes them to a third file)
cat -b file1.txt (prints out line numbers)
tail anirrudh.txt (displays the last 10 lines of the file anirrudh.txt)
tail -n 5 anirrudh.txt (displays the last 5 lines)
tail -f anirrudh.txt (displays output as it written to the file anirrudh.txt - Watching a live file)
wc -w anirrudh.txt (counts of number of words in a file)

wc -l anirrudh.txt (counts number of lines in the file)
grep "<" Example.html | wc -l (gives the number of occurrences of < in the file Example.html)

31. cat a.txt b.txt c.txt d.txt >> abcd.txt
Note: ">>" and ">" are called append symbol. They are used to append the output to a file and not on standard output. ">" symbol will delete a file already existed and create a new file hence for security reason it is advised to use ">>" that will write the output without overwriting or deleting the file.

32. Bash internal Variables
$BASH - path to bash binary itself
$BASH_ENV - env variable pointing to a Bash start up file to be read when a script is invoked
$BASH_SUBSHELL - variable indicating the subshell level
$BASHPID - process ID of the current instance of the Bash
$BASH_VERSINFO[n] - 6 element array containing version information about the installed release of Bash. same as $BASH_VERSION but a bit more detailed
$BASH_VERSION - version of bash installed in the system
$CDPATH - A colon seperated list of search paths available to the cd command, similar in function to the $PATH variable for binaries.

33. cat -n sample.txt - prints lines in the text file with line numbers
cat -b sample.txt - prints lines in the text file with numbers only for non-blank lines
cat -v sample.txt - echoes lines along with non-printable characters for EOL (End Of Line) using ^ notation
NOTE: tac is the reverse of cat - it lists the file contents from end to beginning
rev - reverses each line of the file
chattr - changes file attributes For ex. chattr +i filename marks the file as immutable. The file can't be modifies,linked to, or deleted,not even by the root. This file attribute can be set or removed only by root.
Similarly an option marks the file as append only.
If a file has the u (undelete) attribute set, then when it is deleted, its contents can still be retrieved.
If a file has the c (compress) attribute set, then it will automatically be compressed on writes to disk, and uncompressed on reads.

34. head anirrudh.txt - prints the first 10 lines of the text file
head -n 20 anirrudh.txt - prints the 20 lines of the text file
head -n -0 anirrudh.txt - to view the entire file
tail -n 30 anirrudh.txt prints the last 30 lines of the text file
less anirrudh.txt - if the file is too long

35. grep -nr string fileName - gives the line numbers of the string 'string' in the file 'fileName'
cat -n fileName - displays contents along with line numbers

`grep -inE --color=auto -nr PATCH.inp *` - highlights the string PATCH.inp in all the files of the current directory

grep flags:

-n returns line number
-i ignores case
-v invert match - i.e display lines that does not contain the given string
-x - select the matches that exactly match whole line
-c count number of matches
-L files without match
-l files with match
-q quiet (do not write anything to screen). i.e. exit immediately if any match is found
-H print the file name for each match
-d skip - skips directories while searching
-d recurse OR -r - includes all the files under all the directories in the search
--exclude=GLOB - to exclude files whose basename matches with GLOB. We can also use wild cards
--exclude-from=FILE - exclude all the files contained in the file FILE
--exclude-dir=DIR - excludes directories named DIR from the search
--include=GLOB - search only files whose name matched GLOB

36. `sed "s/hello/hi/g" file1 > file2` - replace hello with hi globally
`sed "s/cat[s]*/sheep/g" file` - replace cat or cats with sheep
`sed "s/\bjames\b/jim/g" file` - match james on word boundaries
`sed "d/+-/-+/" file1 > file2` - swap + and - signs in file

37. `sort data.txt > out.txt` - sort alphabetically, write out.txt
`sort -n data.txt > out.txt` - sort numerically (gen. format)
`sort -n -k 2 data.txt > out.txt` - sort num. by 2nd column
`sort -r -n data.txt > out.txt` - num reverse sort on 1st column
`grep 'hello' data.txt | sort | more` - show sorted output

38. recursively search all the sub directories using grep
`grep -rl "solution" *` - searches all the sub directories and displays all the FILES that contain the string 'solution'
`grep -r "solution" *` - searches all the sub directories and displays all the LINES in all the FILES that contain the string 'solution'
`grep -r "solution" --include=*.{inp,cc}` - seacrches only files with extensions .inp and .cc

39. Delete all the blank lines in a file
`sed '/^$/d' file`

40. Delete all the lines that start with a particular number/letter
`sed -e '/^[a-z]/ d' HS-100-5-new.dat >HS-100-5.dat` (case sensitive) (removes all the lines that start with a lower case letter)