Tutorial 2

Registration Code for CS108-T3 SAFE course → 03RXPXSI

<u>Start Attendance. Mark your attendance, please, in SAFE.</u>

1) **To be covered:-**

- Redirection and pipes
- Process Management
- Access Control
- Different Users
- Regular Expressions
- Some other commands
- Exercises

2) **Redirection symbols (>, >>):-**

- Standard input (stdin; fd is 0)
- Standard output (stdout, fd is 1)
- Standard error (stderr, fd is 2); used when an error has occurred

>: Output of a command redirected to a file
- command > file same as command 1> file (stdout redirected, stderr still screen)
- command 2> file (send stderr to file, stdout is screen)
- command 2> error.txt 1> out.txt (send both to different files)
- command > file 2>&1 (send both to same file)
- command 2> /dev/null (suppress error messages)
  - /dev/null is a special file that discards anything written to it

>> : same as above but append instead of replace

echo “Hi” > you_will_never_see_a_file_with_this_name
(No output comes in the terminal)
echo “There” >> you_will_never_see_a_file_with_this_name

```
(If a file with this name doesn't exist, a new file will be created.)
(If a file existed, using > will write over all the content of the old file.)
rm this_file_have_a_very_long_name (This gives an error and some message in terminal)
rm this_file_have_a_very_long_name 2> /dev/null (No error message but error still exists)
echo "Hey there" 2> temp.txt (Nothing saved in file, and some output message comes)
zip -r output.zip input > /dev/null (No output messages)
```

****Peculiar** → FYI, 2>> also exists.

<: stdin of "command" read from "file.txt

– command < file.txt

```
g++ main.cpp -o main 2> /dev/null; ./main < input.txt 2> /dev/null
```

3) Pipe (|) symbol:-

| : Redirects the output of one command as the input to another command (pipe).

```
man proc | grep -i "executable"
cat temp.txt | head -n 1
ls -Rl ~/Desktop | grep -e ".*\.pdf" | sort
```

****Imp** → There can be multiple pipes in one command.

4) ps command:-

ps : process status, displays a header line, followed by lines containing information about all of your processes that have controlling terminals.

```
ps
ps aux (literally all processes running on the machine)
```

****Interesting but not important** → ps -eo "%P %p %c" (PPID, PID, command executed)

5) kill command:-

Kill: can terminate a process if it hangs and not responding

```
kill <pid> OR kill -9 <pid> (absolute death)
```

6) chmod command:-

“chmod” command helps change access permissions for files you own

- chmod [OPTIONS] MODE FILE(s)/directory(s)
- Only root, the file owner or user with sudo privileges can change the permissions of a file
- Symbolic Mode:
  - u - The file owner.
  - g - The users who are members of the group.
  - o - All other users.
  - a - All users, identical to ugo
  - - Removes the specified permissions.
  - + Adds specified permissions.
  - = Changes the current permissions to the specified permissions
    - If no permissions are specified after the = symbol, all permissions from the specified user class are removed
  - E.g. chmod g=r filename; chmod a-x filename;

Numeric mode:

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- no permissions = 0
- chmod 640 samplefile (octal notation, user has r+w, group has read, others have none)
- chmod -R 700 dirname (Recursively set read, write, and execute permissions to the file owner and no permissions for all other users on a given directory)

```
chmod 777 temp.txt OR chmod 755 temp.txt
chmod -R 777 temp_dir
```

7) sudo and su commands:-

The sudo allows a permitted user to execute a command as the superuser or another user, as specified by the security policy.

The su utility requests appropriate user credentials and switches to that user ID (the default user is the superuser). A shell is then executed.

sudo su (completely switches to the root user)
sudo cat /etc/security/audit_control (example of a protected file)

8) grep command:-

The grep utility searches any given input files, selecting lines that match one or more patterns.

grep “Hey” temp.txt (Case sensitive) (can catch “Hey”, “Heyya”, “Hey!!!”)
grep -i “hey” temp.txt (Case insensitive)
echo “ABC.def” | grep -i “abc\.def”
echo “Heyya” | grep -w “Hey” (No output, searches for Hey as a word, not a part of other)
man man | grep -c “exec” (Tells the count)

Regex:- (V. Imp.)

Meta-characters (some of them are listed below):

- ^ : start of a line (NOTE: Can also mean “not” if used inside [])
- $: end of line
- . : match any single character
- \ : escape a special character
- | : logical or operation i.e. match a particular character set on either side
- * : search for a character that occurs zero or more times as defined by the preceding character
- + : search for a character that occurs one or more times as defined by the preceding character
- ? : search for a character that occurs zero or one time as defined by the preceding character
- \d : represents any single numeral, 0 through 9
- \s : represents space

A quantifier is a syntactic structure in regular expressions that indicates the number of times a character occurs in sequence in the input text.
Some of them are listed below:

- {n} : the preceding character needs to occur exactly n times
- {n,} : the preceding character needs to occur at least n times
- {n,m} : the preceding character needs to occur between n and m times

Groups and ranges:

- (<def>) : a group of characters declared according to a specific definition
- [<range>] : match any character from range of given characters in the []
    - [0-9] : match any digit from 0-9
    - [a-z] : match any lowercase letter from a-z
    - [A-Z] : match any uppercase letter from A-Z
- [^<range>] : match any character not in the range of given characters in the []
    - [^adA-Z] : match any character which is not a, d or lies in A-Z

echo “kavya 123” | grep -e “^[a-zA-Z0-9 _]+$”
echo “kavya_123” | grep -e “^[a-zA-Z0-9 _]+$”
echo “kavya@123” | grep -e “^[a-zA-Z0-9 _]+$” (No output means regex didn’t match)
echo "hi" | grep -E "^(hi|bye)$"
echo "bye" | grep -E "^(hi|bye)$"

****Note** → Use this website for regex practice:- https://regexone.com/

9) cut command:-

cut out selected portions of each line of a file
Some of the useful options are:

- ➢ -d : Used to specify field delimiter character (default is tab)
- ➢ -f : Used to specify the fields desired in the output and separated in the input by field delimiter character.

echo “Kavya,81,92,78” | cut -d ‘,’ -f 1,2
echo “Kavya,81,92,78” | cut -d ‘,’ -f 1-3
(By default, the delimiter is considered a single whitespace.)

10) wc command:-

The wc utility displays the number of lines, words, and bytes contained in each input file
Some of the useful options are:

- ➢ -l : Used to get the number of lines in each input file
- ➢ -w : Used to get the number of words in each input file

Example: By default wc outputs #lines, #words, #characters present in the input files.

ls -l | wc -l

11) sort command:-

The sort utility sorts text and binary files by lines. A line is a record separated from the subsequent record by a newline (default).
Some of the useful options are:

- -t : Used to specify field separator character.
- -r : Used to sort in reverse order.

12) zip, unzip, and tar commands:-

- zip is a compression and file packaging utility for Unix
- To zip a folder recursively, we will use the following command
  `zip -r <zip file> <folder>`
- unzip - list, test and extract compressed files in a zip archive
- To unzip, we will use the following command
  `unzip <zip file>`

- tar creates and manipulates streaming archive files.
- To tar a folder, we will use the following command
  `tar -cvzf <output file> <folder>`
- To untar, we will use the following command
  `tar -xvzf <tar file>`

13) find command:-

As the name suggests, it can *find* anything.
find . -name "*.pdf"
find test/ -type f
find . -perm 755

14) which command:-

Tells the location of the executable file.
which ls
which cd (gives error)
Not all commands are executables :-), think about it.