

**Topic Name:**

The main aim of this lab session is to provide hands-on experience on

- Explore file structure
- File management commands
- Absolute path and Relative path
- Globbing - Scripting

**File Structure**

1. Under the root directory there are many files like /bin , /boot , /dev , /etc , ....  
Find out the importance of those files  
Example : /etc is for user account details

S.No	Directory	Usage
1	/	Root directory
2	/bin	Binary files
3	/boot	To store files necessary for boot process
4	/dev	Provide access to peripheral devices such as hard disks, to resources on peripheral devices such as disk partitions, and pseudo devices such as random number generator
5	/etc	A central location for storing system configuration files and directories that are essential for the proper functioning of the system and its installed applications
6	/home	A personal working space for all the users except root
7	/lib	A personal working space for all the users except root
8	/proc	To obtain information about the system and to change certain kernel parameters at runtime
9	/sbin	Primarily used by the system administrator for system maintenance and management tasks.
10	/tmp	To provide a space for temporary files generated by running programs and processes
11	/var	Used by busy applications such as accounting, mail, and the print spooler.

12	/mnt	Mount points to removable or temporary files storage
13	/opt	Providing a clear and hierarchical organization of installed software

2. In Linux, there are three different files

Regular file

Directory

Special file

Block file

Character file

Socket file

Pipe file

Fill the below table:

File Type	Represented by (Hint ls )	Role	How to create	How to check	Location	Screen shot
Regular file		In computing, a regular file is a type of file that contains user-created data in a specific format	Touch filename	NA		
-Text file	-	These files contain plain text and are usually saved with a .txt extension.	Touch filename.txt	NA		
- Compressed file	-	These files contain one or more files that have been compressed to reduce their size.	Zip [options] [zipfile] [file1] [file2]	NA		
-Image	-	These files contain digital images.	NA	NA		

Directory	d	A directory is a special type of file that serves to store a list of file names and their associated metadata, organizing files within a hierarchical structure known as the directory tree	mkdir <filename>	NA		
Block file	b	A block file, also known as a block device file, represents a type of device that allows for buffered access to hardware devices.	NA	NA	‘/dev’	
Character file	c	Character files, also known as character device files, are a type of special file that provides a way for the operating system to communicate with hardware devices.	NA	NA	‘/dev’	
Socket file	s	Socket files are a critical component of IPC in Linux, enabling efficient communication between processes on the same machine.	NA	NA	‘/tmp’	
pipe file		Pipes are an essential mechanism for IPC in Linux, facilitating	NA	NA		

		efficient data transfer between processes.				
--	--	--	--	--	--	--

### 3. Globbing

- a. Go back to CYS
- b. Create multiple sub directories using single command

LS

Unit1

command  
glob

Unit2

command  
grep

Unit3

constructs

```

kali@kali: ~/CYS
File Actions Edit View Help
(kali@kali)-[~/CYS]
$ mkdir -p CYS CYS/LS/Unit1/{command,glob} CYS/LS/Unit2/{command,grep} CYS/LS/Unit3/Constructs
(kali@kali)-[~/CYS]
$ tree CYS
CYS
├── LS
│   ├── Unit1
│   │   ├── command
│   │   └── glob
│   ├── Unit2
│   │   ├── command
│   │   └── grep
│   └── Unit3
│       └── Constructs
└── 10 directories, 0 files
(kali@kali)-[~/CYS]
$

```

- c. Navigate to unit1/glob

```

(kali@kali)-[~/CYS/CYS/LS]
$ cd Unit1/glob

```

- d. Create the following files :

Commands.txt  
Commands1.txt  
Commands2.txt  
page1.html  
page2.html

```

page3.html
file1 file10
file11 file2
File2 File3
file33
fileAB filea
fileA
fileAAA
file(
file 2

```

```

kali@kali: ~/CYS
File Actions Edit View Help
(kali@kali)~-[~/CYS]
$ touch Commands.txt Commands{1..2}.txt page{1..3}.html file{1..2} file{0..1} File{2..3} file33 file33 fileAB filea fileA fileAAA file\ file\2

```

i. List all files starting with file

```

(kali@kali)~-[~/CYS]
$ ls file*
'file(' file1 file10 file11 file2 file33 filea fileA fileAAA fileAB

```

ii. List all files starting with File

```

(kali@kali)~-[~/CYS]
$ ls File*
File2 File3

```

iii. List all files starting with file and ending in a number.

```

(kali@kali)~-[~/CYS]
$ ls file*[0-9]
file1 file10 file11 file2 file33

```

iv. List all files starting with file and ending with a letter

```

(kali@kali)~-[~/CYS]
$ ls file*[a-zA-Z]
filea fileA fileAAA fileAB
(kali@kali)~-[~/CYS]
$

```

v. List all files starting with File and having a digit as fifth character.

```

(kali@kali)~-[~/CYS]
$ ls File???[0-9]*
ls: cannot access 'File???[0-9]*': No such file or directory

```

vi. List all files starting with File and having a digit as fifth character and nothing else.

```

(kali@kali)~-[~/CYS]
$ ls File???[0-9]
ls: cannot access 'File???[0-9]': No such file or directory

```

vii. List (with ls) all files starting with a letter and ending in a number.

```
(kali@kali)-[~/CYS]
$ ls [a-zA-Z]*[0-9]
file1 file10 file11 file2 File2 File3 file33
```

viii. List (with ls) all files that have exactly five characters.

```
(kali@kali)-[~/CYS]
$ ls ?????
'file(' file1 file2 File2 File3 filea fileA
```

ix. List (with ls) all files that start with f or F and end with 3 or A.

```
(kali@kali)-[~/CYS]
$ ls [fF]*[3A]
File3 file33 fileA fileAAA
```

x. List (with ls) all files that start with f have i or R as second character and end in a number.

```
(kali@kali)-[~/CYS]
$ ls f[iR]*[0-9]
file1 file10 file11 file2 file33
```

xi. List all files that do not start with the letter F.

```
(kali@kali)-[~/CYS]
$ ls -l [!F]*
zsh: event not found: F]
```

xii. Remove all the \*.html

```
(kali@kali)-[~/CYS]
$ rm *.html
```

xiii. Rename \*.txt to \*.json

```
(kali@kali)-[~/CYS]
$ ls
Commands1.txt Commands.txt 'file(' file10 file2 File3 filea fileAAA
Commands2.txt CYS file1 file11 File2 file33 fileA fileAB

(kali@kali)-[~/CYS]
$ for file in *.txt; do mv "$file" "${file%.txt}.json"; done

(kali@kali)-[~/CYS]
$ ls
Commands1.json Commands.json 'file(' file10 file2 File3 filea fileAAA
Commands2.json CYS file1 file11 File2 file33 fileA fileAB

(kali@kali)-[~/CYS]
$
```

#### 4. Absolute path and relative path

Use rm, mv, cp, ls with absolute path and relative path as per your choice.

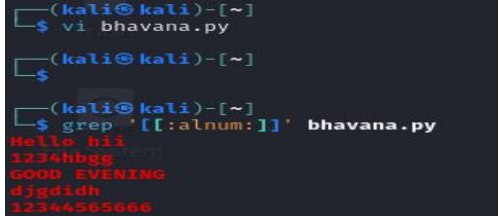
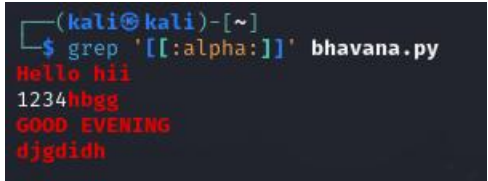



Command	absolute path	relative path
ls	<pre>(kali@kali)-[~] \$ ls /home/kali/bhavana 1.py file1.txt  (kali@kali)-[~] \$ ls bhavana 1.py file1.txt</pre>	<pre>(kali@kali)-[~] \$ ls /home/kali/bhavana 1.py file1.txt  (kali@kali)-[~] \$ ls bhavana 1.py file1.txt</pre>

rm	<pre>(kali@kali)-[~/bhavana] \$ rm /home/kali/bhavanagowda</pre>	<pre>(kali@kali)-[~/bhavana] \$ rm bhavanagowda</pre>
mv	<pre>(kali@kali)-[~/bhavana] \$ mv /home/kali/bhavana/file1.txt /home/kali/bhavanagowda</pre>	<pre>(kali@kali)-[~/bhavana] \$ mv 1.py bhavanagowda</pre>
cp	<pre>(kali@kali)-[~/bhavana] \$ cp /home/kali/bhavana/bhavanagowda /home/kali/anusha</pre>	<pre>(kali@kali)-[~/bhavana] \$ cp bhavanagowda bhavana</pre>

## 5. Wildcards

Notation	Use	Example	Screenshot
*	Used to match one or many character	ls *	<pre>(kali@kali)-[~/CYS] \$ ls * Commands1.json  Commands.json  file1  file11  file2  file33  fileA  fileAB Commands2.json  file(  file10  file2  file3  filea  fileAAA</pre>
?	Used to match only one character	ls Commands?.json	<pre>(kali@kali)-[~/CYS] \$ ls Commands?.json Commands1.json  Commands2.json</pre>
[ ]	Used to match a single character from a set of specified characters.	ls file[1-3]	<pre>(kali@kali)-[~/CYS] \$ ls file[1-3] file1  file2</pre>
[! ]	Matches any character that is not a member of the set characters	ls file[!1]	<pre>(kali@kali)-[~/CYS] \$ ls file[!1] file2</pre>
{ }	Used to generate multiple arguments by separating the values with commas	Echo Commands{1..2}.txt	<pre>(kali@kali)-[~/CYS] \$ echo Commands{1..2}.txt Commands1.txt  Commands2.txt</pre>

#### More on Character class

Notation	Use	Example	Screenshot
<code>[:alnum:]</code>	Matches any alphanumeric character	grep <code>'[:alnum:]'</code> bhavana.py	 <pre> (kali@kali)-[~] \$ vi bhavana.py (kali@kali)-[~] \$ (kali@kali)-[~] \$ grep '[:alnum:]' bhavana.py Hello hii 1234hbgg GOOD EVENING djgdidh 12344565666 </pre>
<code>[:alpha:]</code>	Matches any alphabetic character	grep <code>'[:alpha:]'</code> bhavana.py	 <pre> (kali@kali)-[~] \$ grep '[:alpha:]' bhavana.py Hello hii 1234hbgg GOOD EVENING djgdidh </pre>
<code>[:digit:]</code>	Matches any numeric digit (0-9).	grep <code>'[:digit:]'</code> bhavana.py	 <pre> (kali@kali)-[~] \$ grep '[:digit:]' bhavana.py 1234hbgg 12344565666 </pre>
<code>[:lower:]</code>	Matches any lowercase alphabetic character	grep <code>'[:lower:]'</code> bhavana.py	 <pre> (kali@kali)-[~] \$ grep '[:lower:]' bhavana.py Hello hii 1234hbgg djgdidh </pre>
<code>[:upper:]</code>	Matches any uppercase alphabetic character (A-Z)	grep <code>'[:upper:]'</code> bhavana.py	 <pre> (kali@kali)-[~] \$ grep '[:upper:]' bhavana.py Hello hii GOOD EVENING </pre>

#### 4. change permission



- a) Change the permission set of /work/readme.txt so that only the user (owner) can read, write, and execute it. Use absolute mode.

```
(kali㉿kali)-[~/work]
$ chmod 700 readme.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rwx----- 1 kali kali 0 Aug 11 07:01 readme.txt
```

- b) Change the permission set of /work/readme.txt so that any user can read it, the group can read/write to it and the user (owner) can read/write/execute it. Use absolute mode.

```
(kali㉿kali)-[~/work]
$ chmod 764 readme.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rwxrw-r-- 1 kali kali 0 Aug 11 07:01 readme.txt
```

- c) Change the permission set of /bin/bash so that only the user (owner) can read/write/execute, group, and any user can execute it. However, whenever anyone executes it, it should run with the privileges of the owner user. Use absolute mode.

```
(kali㉿kali)-[/home]
$ chmod 711 /bin/bash

chmod: changing permissions of '/bin/bash': Operation not permitted
```

- d) Change the permission set of /work/readme.txt so that only the user (owner) can read, write, and execute it. Use relative mode.

```
(kali㉿kali)-[~/work]
$ chmod u+rwx,go-rwx readme.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rwx----- 1 kali kali 0 Aug 11 07:01 readme.txt
```

- e) Change the permission set of /work/readme.txt so that any user can read it, the group can read/write to it and the user (owner) can read/write/execute it. Use relative mode.

```

(kali㉿kali)-[~/work]
$ chmod u=rwx,g=rw,o=r readme.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rwxrw-r-- 1 kali kali 0 Aug 11 07:01 readme.txt

```

- f) Change the permission set of /work/readme.txt so that only the user (owner) can read/write/ execute, group, and any user can execute it. However, whenever anyone executes it, it should run with the privileges of the group. Use absolute mode.

```

(kali㉿kali)-[~/work]
$ chmod 711 readme.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rwx--x--x 1 kali kali 0 Aug 11 07:01 readme.txt

```

- g) Change the permission set of /work/readme.txt so that only the owner can rename or delete this file while maintaining the existing permissions. Use absolute mode.

```

(kali㉿kali)-[~/work]
$ sudo chattr +i readme.txt

(kali㉿kali)-[~/work]
$ lsattr readme.txt
---i-----e----- readme.txt

```

- h) What are the default permissions for the new file?

```

(kali㉿kali)-[~/work]
$ touch file1.txt

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rw-rw-r-- 1 kali kali 0 Aug 11 07:26 file1.txt

```

- i) What was the command to view the file permissions?

```

(kali㉿kali)-[~/work]
$ ls -l
total 0
-rw-rw-r-- 1 kali kali 0 Aug 11 07:26 file1.txt
-rwx--x--x 1 kali kali 0 Aug 11 07:01 readme.txt

(kali㉿kali)-[~/work]
$

```

- j) Change chmod.exercises permissions to -r--r--r

```

(kali㉿kali)-[~/work]
$ touch chmod.exercises

(kali㉿kali)-[~/work]
$ chmod 444 chmod.exercises

(kali㉿kali)-[~/work]
$ ls -l chmod.exercises
-r--r--r-- 1 kali kali 0 Aug 11 07:30 chmod.exercises

```

k) Change the file permissions to Read only for the owner, group and all other users.

```

(kali㉿kali)-[~/work]
$ chmod 444 chmod.exercises

(kali㉿kali)-[~/work]
$ ls -l chmod.exercises
-r--r--r-- 1 kali kali 0 Aug 11 07:30 chmod.exercises

(kali㉿kali)-[~/work]
$ chmod u=r,g=r,o=r chmod.exercises

(kali㉿kali)-[~/work]
$ ls -l chmod.exercises
-r--r--r-- 1 kali kali 0 Aug 11 07:30 chmod.exercises

```

l) What was the command for changing the file permissions to -r--r--r--?

```

(kali㉿kali)-[~/work]
$ chmod 444 chmod.exercises

```

m) Change chmod.exercises permissions to -rw-r-----

```

(kali㉿kali)-[~/work]
$ ls -l chmod.exercises
-rw-r--r-- 1 kali kali 0 Aug 11 07:30 chmod.exercises

(kali㉿kali)-[~/work]
$ chmod u=rw,g=r,o-r chmod.exercises

(kali㉿kali)-[~/work]
$ ls -l chmod.exercises
-rw-r----- 1 kali kali 0 Aug 11 07:30 chmod.exercises

```

n) Change the file permissions to match the following:

- a. owner: Read and Write
- b. group: Read
- c. other: no permissions (None)

```
(kali@kali)-[~/work]
$ ls -l chmod.exercises
-rw-r--r-- 1 kali kali 0 Aug 11 07:30 chmod.exercises

(kali@kali)-[~/work]
$ chmod u=rw,g=r,o-r chmod.exercises

(kali@kali)-[~/work]
$ ls -l chmod.exercises
-rw-r----- 1 kali kali 0 Aug 11 07:30 chmod.exercises
```

o) What was the command for changing the file permissions to -rw-r-----?

```
(kali@kali)-[~/work]
$ chmod 640 chmod.exercises
```

p) Change chmod.exercises permissions to -rwxr-x--x

```
(kali@kali)-[~/work]
$ chmod 751 chmod.exercises

(kali@kali)-[~/work]
$ ls -l chmod.exercises
-rwxr-x--x 1 kali kali 0 Aug 11 07:30 chmod.exercises
```

q) Change the file permissions to match the following:

- owner: Read, Write and Execute
- group: Read and Execute
- other: Execute

```
(kali@kali)-[~/work]
$ chmod 751 chmod.exercises

(kali@kali)-[~/work]
$ ls -l chmod.exercises
-rwxr-x--x 1 kali kali 0 Aug 11 07:30 chmod.exercises
```

r) What was the command for changing the file permissions to -rwxr-x--x?

```
(kali@kali)-[~/work]
$ chmod 751 chmod.exercises
```

Evaluation :

Marks : 10 (Deadline : 4 – Originality :3 – Completeness :3 )

Deadline: 06.08.2024

In life there are no shortcuts. All things are connected. For success there is no fast lane.  
Work hard. Focus your energy, practice, remain honest, Truthful, loyal and committed.

-unknown

