Meghana D Duttargi

241059042

LINUX OS & SCRIPTING LAB

M.E – CYBER SECURITY

Topic Name:

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

Explore file structure

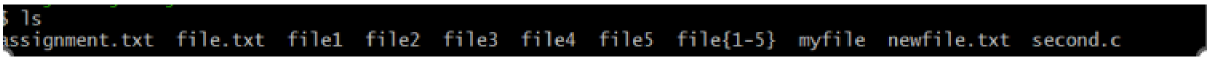
1. **/** - Root directory: The top-level directory of the file system.
2. **/bin** - Binary files: Contains essential command binaries for all users.
3. **/boot** - Boot files: Contains boot loader files and the Linux kernel.
4. **/dev** - Device files: Contains device files for all hardware devices.
5. **/etc** - Configuration files: Contains system-wide configuration files and scripts. *
6. **/lib** - Shared libraries: Essential shared libraries and kernel modules.
7. **/proc** - Process information: Virtual file system for process and system info.
8. **/sbin** - System binaries: Essential binaries for system administration.
9. **/tmp** - Temporary files: Stores temporary files, cleared upon reboot.
10. **/var** - Variable files: Logs, spools, and temporary files for various processes.

File Types

File Type	Represented by (Hint: ls)	Role	How to create	How to check	Location
Regular file	-	Store data, text, or binary information.	touch file	ls -l	/home
Text file	-	Store plain text.	touch file.txt	ls -l	/home
Compressed file		Store compressed data.	gzip file	file file.gz	/var
Image		Store image data.	convert	file image	/images
Directory	d	Organize files.	mkdir dir	ls -ld	/
Block file	b	Represent block devices.	mknod	ls -l	/dev
Character file	c	Represent character devices.	mknod	ls -l	/dev
Socket file	s	Facilitate communication between processes.	socket	ls -l	/run
Pipe file	p	Facilitate inter-process communication.	mkfifo	ls -l	/tmp

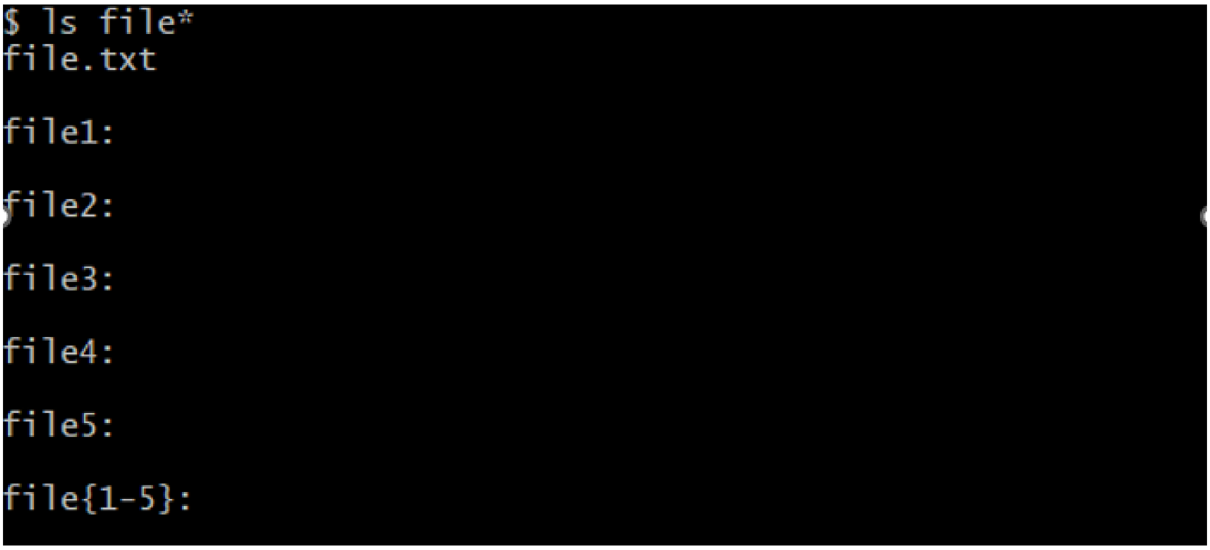
1. Globbing

1. List all files starting with file



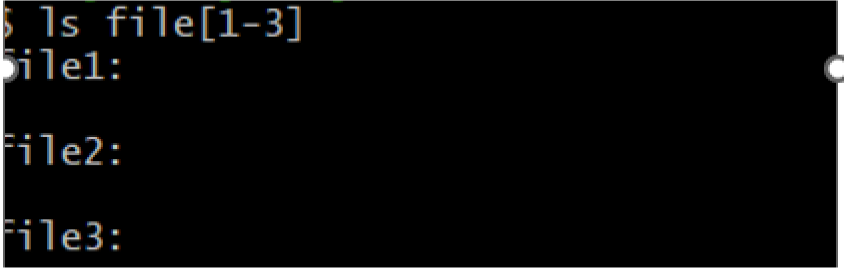
```
$ ls
assignment.txt  file.txt  file1  file2  file3  file4  file5  file{1-5}  myfile  newfile.txt  second.c
```

2. List all files starting with File



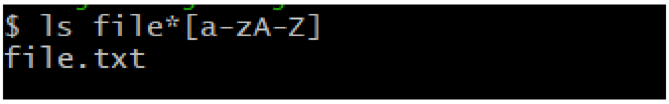
```
$ ls file*
file.txt

file1:

file2:

file3:

file4:

file5:

file{1-5}:
```

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



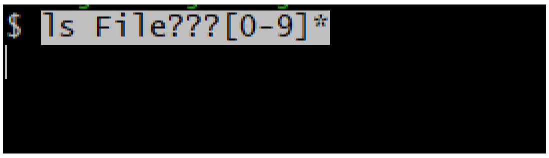
```
$ ls file[1-3]
file1:

file2:

file3:
```

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



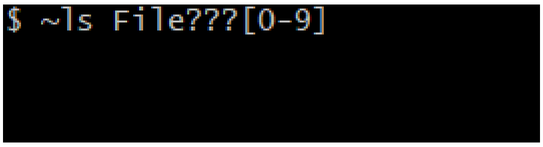
```
$ ls file*[a-zA-Z]
file.txt
```

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



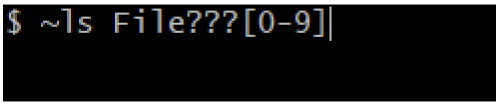
```
$ ls File???[0-9]*
```

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



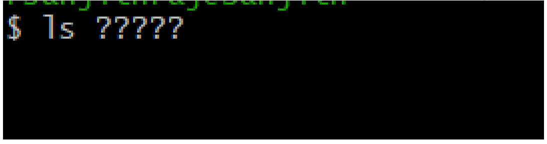
```
$ ~ls File???[0-9]
```

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



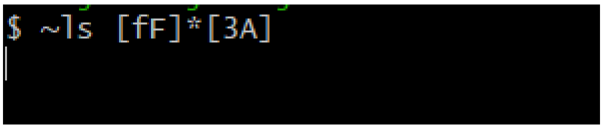
```
$ ~ls File???[0-9]
```

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



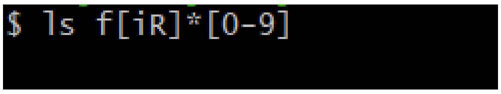
```
$ ls ?????
```

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



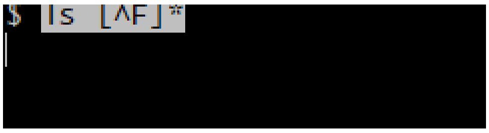
```
$ ~ls [fF]*[3A]
```

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



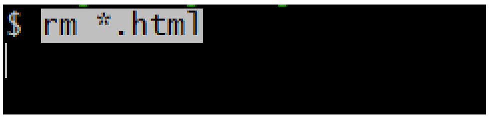
```
$ ls f[iR]*[0-9]
```

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



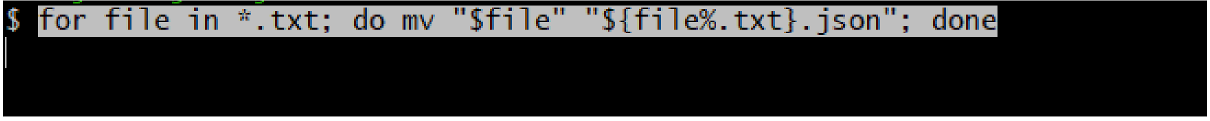
```
$ ls [^F]*
```

12. Remove all the *.html



```
$ rm *.html
```

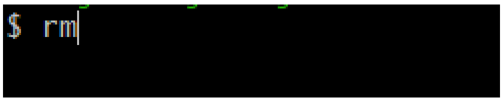
13. Rename *.txt to *.json



```
$ for file in *.txt; do mv "$file" "${file%.txt}.json"; done
```

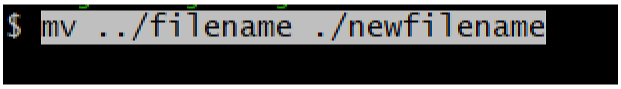
2. Absolute path and relative path

Rm



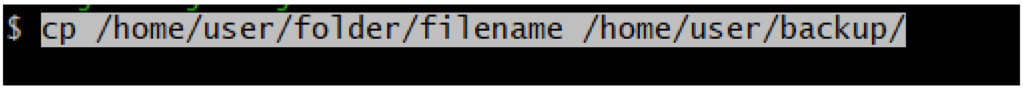
```
$ rm
```

Mv



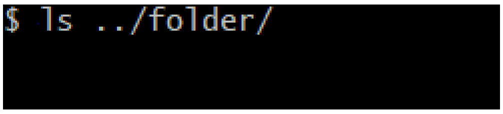
```
$ mv ../filename ./newfilename
```

Cp



```
$ cp /home/user/folder/filename /home/user/backup/
```

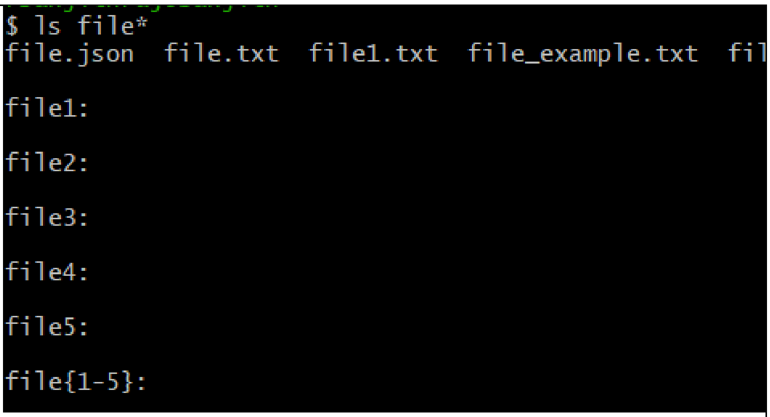
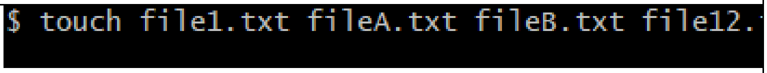
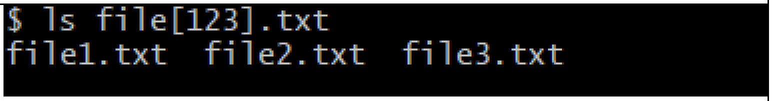
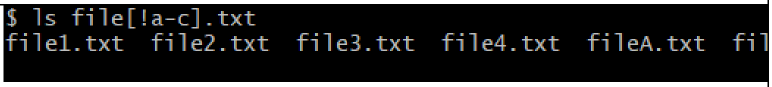
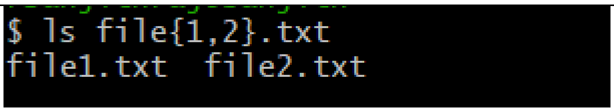
Ls



```
$ ls ../folder/
```

3. Wildcards

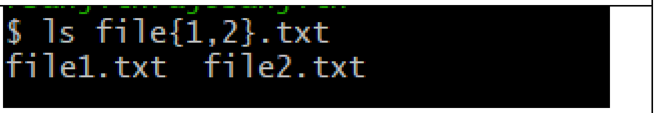
Notation	Use	Example	Screenshot

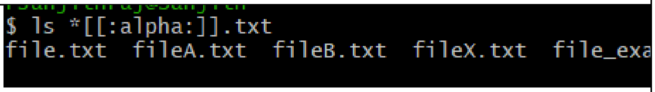
*	Matches any number of characters	ls file* lists all files starting with "file".	 $ ls file* file.json file.txt file1.txt file_example.txt fil file1: file2: file3: file4: file5: file{1-5}:
?	Matches a single character	ls file?.txt lists files like file1.txt, fileA.txt	 $ touch file1.txt fileA.txt fileB.txt file12.
[ ]	Matches any single character within the brackets	ls file[123].txt lists file1.txt, file2.txt, file3.txt	 $ ls file[123].txt file1.txt file2.txt file3.txt
[!]	Matches any character that is not a member of the set characters	ls file[!a-c].txt lists files not starting with a, b, or c.	 $ ls file[!a-c].txt file1.txt file2.txt file3.txt file4.txt fileA.txt fil
{ }	Matches any one of the comma-separated	ls file{1,2}.txt lists file1.txt and file2.txt.	 $ ls file{1,2}.txt file1.txt file2.txt

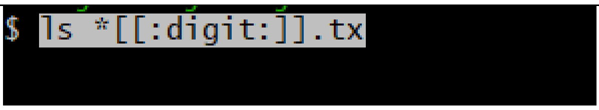
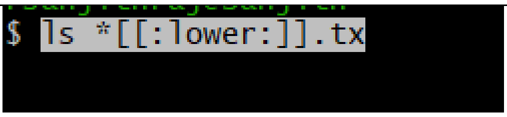
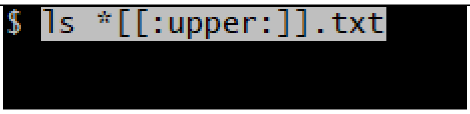
	patterns within the braces		

More on Character class

Notation	Use	Example	Screenshot

[:alnum:]	Matches any alphanumeric character (letters and digits)	ls *[[:alnum:]].txt	
			`$ ls file{1,2}.txt`
			`file1.txt  file2.txt`

[:alpha:]	Matches any alphabetic characte	ls *[[:alpha:]].txt	
			`$ ls *[[:alpha:]].txt`
			`file.txt  fileA.txt  fileB.txt  fileX.txt  file_exa`

	r (letters only)		
[:digit:]	Matches any digit (0-9)	ls *[[:digit:]].txt	 `$ ls *[[:digit:]].tx`
[:lower:]	Matches any lowercase letter	ls *[[:lower:]].txt	 `$ ls *[[:lower:]].tx`
[:upper:]	Matches any uppercase letter	ls *[[:upper:]].txt	 `$ ls *[[:upper:]].txt`

4. change permission

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

```
chmod 700 /work/readme.txt
```

2. 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.

```
chmod 764 /work/readme.txt
```

3. 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.

```
chmod 4755 /bin/bash
```

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

```
chmod u+rwx,g-rwx,o-rwx
/work/readme.txt
```

5. 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.

```
chmod 700 /work/readme.txt
```

```
chattr +i /work/readme.txt
```

6. 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.

```
chmod 2755 /work/readme.txt
```

7. 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.

```
ls -l
```

8. What are the default permissions for the new file?

```
chmod 444 chmod.exercises
```

9. What was the command to view the file permissions?

```
chmod 640 chmod.exercises
```

10. Change chmod.exercises permissions to -r--r--r—

```
chmod 751 chmod.exercises
```

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

```
chmod 444 filename
```

12. What was the command for changing the file permissions to -r--r--r--?

```
chmod 444 filename
```

13. Change chmod.exercises permissions to -rw-r-----

```
chmod 640 chmod.exercises
```

14. Change the file permissions to match the following:
    a. owner: Read and Write
    b. group: Read
    c. other: no permissions (None)

```
chmod 640 filename
```

15. What was the command for changing the file permissions to -rw-r-----?

```
chmod 640 filename
```

16. Change chmod.exercises permissions to -rwxr-x—x

```
chmod 751 chmod.exercises
```

17. Change the file permissions to match the following:

a. owner: Read, Write and Execute
b. group: Read and Execute
c. other: Execute

```
chmod 751 filename
```

18. What was the command for changing the file permissions to -rwxr-x--x?

```
chmod 751 filename
```