

Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre



A Lab Report
on
“Operating System Lab-I”

[Code No.: COMP 307]

Submitted by

Miraj Sapkota

Roll No.: 46

Submitted to

Ms. Rabina Shrestha

Department of Computer Science and Engineering

December 10, 2025

Questions

Q1: What is Linux?

Linux is an open-source operating system based on the Unix architecture. It manages hardware resources, executes commands, and provides a secure multi-tasking environment. It powers servers, desktops, embedded devices, and even supercomputers. Some of the most popular Linux distributions are ArchLinux, Ubuntu, RedHat, etc.

Q2: The Linux Hierarchical File System

Linux uses a hierarchical file structure that begins at the root directory /. Everything in Linux is a file or a directory, and all paths originate from /. Common directories include:

- /home – User home directories
- /bin – Essential command binaries
- /etc – Configuration files
- /usr – User utilities and applications
- /var – Logs, caches, temporary data

Q3: Importance of Linux commands in Operating Systems

Linux commands are critical because they provide a direct and powerful interface to the operating system. They allow users and administrators to navigate the file system, manage files and directories, monitor system performance, and automate tasks through scripting. Unlike graphical interfaces, command-line commands are faster, use fewer resources, and provide more precise control. Mastering these commands enhances efficiency, troubleshooting capabilities, and overall understanding of how the OS operates, making it indispensable for system administrators, developers, and power users.

Linux/Unix Commands

1. **pwd**

The `pwd` command prints your current working directory. It tells you exactly where you are located inside the Linux file system. This is extremely useful when navigating through multiple folders, working in scripts, or verifying paths before executing commands that affect files. Since Linux uses a hierarchical file structure starting at the root `/`, `pwd` helps ensure you never get lost.

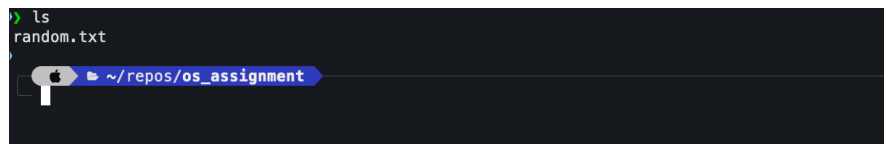
```
> pwd
/Users/miraj/repos/os_assignment
```



2. **ls**

The `ls` command lists all files and directories in your current location. It gives a quick overview of the contents of a folder and is one of the most frequently used commands. By default, it shows only visible items (non-hidden files).

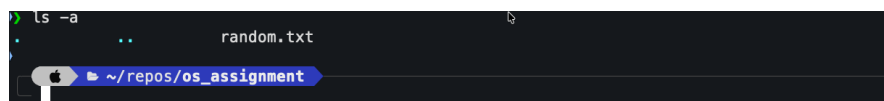
```
> ls
random.txt
```



3. **ls -a**

This version of `ls` displays all files, including hidden ones. Hidden files in Linux start with a dot (`.`), such as `.bashrc` or `.config`. These files usually store configurations and preferences.

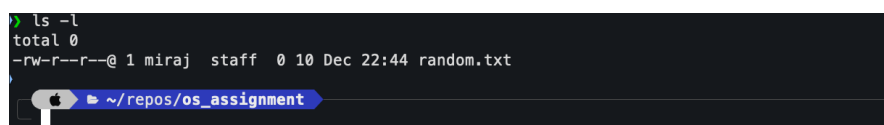
```
> ls -a
.  ..  random.txt
```



4. **ls -l**

The `-l` option displays a long, detailed listing. It includes file permissions, owner, group, size, and last modification time. This format is essential for understanding access rights and managing file security.

```
> ls -l
total 0
-rw-r--r--@ 1 miraj  staff  0 10 Dec 22:44 random.txt
```



5. **cd**

The `cd` command lets you move between directories. It is used to navigate the Linux filesystem. You can move into subdirectories, return to the parent directory using `cd ..`, or jump to a specific absolute path.

```
> cd
```

6. **mkdir**

mkdir creates a new directory. It is commonly used to organize files by grouping them into folders. You can also create multiple folders at once, or even nested folders using **mkdir -p**.

```
> mkdir lab
> ls
lab      random.txt
```

7. **rmdir**

This command removes an empty directory. It cannot delete directories that contain files. It is useful for cleaning up folder structures or removing temporary empty folders. To remove the folder there must be a folder that is created.

```
> rmdir lab
> ls
random.txt
```

8. **rm**

The **rm** command deletes files permanently (no recycle bin). It should be used carefully because deleted files are not easily recoverable. You can also remove multiple files at once.

```
> ls
random.txt
> rm random.txt
> ls
```

9. **rm -r folder_name**

The **-r** option stands for recursive deletion. It removes a directory and everything inside it — files, subfolders, and all. This is powerful and potentially dangerous, so double-check the directory before executing.

```
> ls lab/dummy.txt
lab/dummy.txt
> rm -r lab
> ls
```

10. **touch**

touch is used to create an empty file or update the timestamp of an existing file. It is commonly used in scripting or when preparing placeholder files.

```
➤ touch random.txt
➤ ls
random.txt
```

🍏 ~/repos/os_assignment

11. **cat**

The cat command reads and displays the content of a file directly in the terminal. It is also used to combine files or create files using output redirection.

```
➤ vi random.txt
➤ cat random.txt
hi whats up
```

12. nano, vi, jed

These are terminal-based text editors. nano is beginner-friendly, vi (or vim) is a powerful editor popular among developers, and jed provides a lightweight interface. They allow editing, writing, and saving files directly from the terminal.

```
hi whats up
```

13. **ср**

`cp` copies files from one place to another. You can also copy directories using the `-r` option. This command is essential for backups, duplication, and organizing files.

```
> ls destination
> cp random.txt destination
> ls destination
random.txt
```

```
~/repos/os_assignment
```

14. **mv**

mv allows you to move or rename files and directories. Renaming is simply a move within the same folder. It's used for reorganizing or updating file names.

```
> mv destination/random.txt .
> ls
destination random.txt
```

```
~/repos/os_assignment
```

15. **locate**

This command searches for files by name. It uses a system database, which makes the search extremely fast. It's ideal for finding misplaced files.

```
> locate random.txt
/Users/miraj/repos/os_assignment/random.txt
```

```
~/
```

16. **echo**

Prints text or variable values to the terminal. It is commonly used in scripts to produce messages, debug values, or write text into files using redirection.

```
> echo "hello"
hello
```

```
~/
```

17. **uname -a**

Shows complete system information, including kernel version, machine architecture, hostname, and operating system. Useful for debugging or checking system specs.

```
> uname -a
Darwin M1.local 25.1.0 Darwin Kernel Version 25.1.0: Mon Oct 20 19:32:47 PDT 2025; root:xnu-12377.41.6~2/RELEASE_ARM64_T8103 arm64
```

```
~/
```

18. **df -h**

Displays disk usage in human-readable format (MB/GB). It shows total size, used space, available space, and mounted filesystems. Handy for monitoring storage.

Filesystem	Size	Used	Avail	Capacity	used	ifree	%used	Mounted on
/dev/disk3s1s1	228Gi	11Gi	59Gi	17%	451k	615M	0%	/
devfs	204Ki	204Ki	0Bi	100%	706	0	100%	/dev
/dev/disk3s6	228Gi	2.0Gi	59Gi	4%	2	615M	0%	/System/Volumes/VM
/dev/disk3s2	228Gi	7.6Gi	59Gi	12%	1.4k	615M	0%	/System/Volumes/Preboot
/dev/disk3s4	228Gi	6.4Mi	59Gi	1%	107	615M	0%	/System/Volumes/Update
/dev/disk1s2	500Mi	6.0Mi	482Mi	2%	1	4.9M	0%	/System/Volumes/xarts
/dev/disk1s1	500Mi	5.7Mi	482Mi	2%	30	4.9M	0%	/System/Volumes/ISCPreboot
/dev/disk1s3	500Mi	1.5Mi	482Mi	1%	69	4.9M	0%	/System/Volumes/Hardware
/dev/disk3s5	228Gi	147Gi	59Gi	72%	2.3M	615M	0%	/System/Volumes/Data
map auto_home	0Bi	0Bi	0Bi	100%	0	0	-	/System/Volumes/Data/home

19. `ps -u $USER`

Lists all currently running processes for your user. It displays process IDs, CPU usage, memory usage, and command names. Very useful for identifying unnecessary or stuck processes.

```
ps -u $USER
  PID TTY          TIME CMD
 501   ??        0:15.88 /usr/sbin/distnoted agent
 501   ??        0:21.04 /usr/sbin/cfprefsd agent
 501   ??        1:30.23 /usr/libexec/UserEventAgent (Aqua)
 501   ??        0:05.71 /usr/sbin/universalaccessd launchd -s
 501   ??        0:09.16 /usr/libexec/knowledge-agent
 501   ??        0:00.76 /usr/libexec/pboard
 501   ??        0:05.76 /usr/libexec/containermanagerd --runmode=agent --user-container-mode=current --bundle-container-mode=proxy --systemd
 501   ??        0:51.24 /System/Library/PrivateFrameworks/BiomeStreams.framework/Support/BiomeAgent
 501   ??        0:03.65 /usr/libexec/lsd
 501   ??        1:19.27 /System/Library/CoreServices/talagentd
 501   ??        1:04.23 /System/Library/CoreServices/WindowManager.app/Contents/MacOS/WindowManager
 501   ??        0:30.91 /System/Library/CoreServices/Dock.app/Contents/MacOS/Dock
 501   ??        2:30.70 /System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter
```

20. `top`

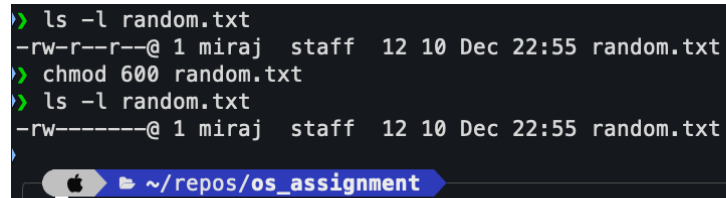
Displays real-time system activity. It shows active processes, CPU load, memory usage, and system uptime. It's one of the most important performance-monitoring commands.

```
Processes: 407 total, 2 running, 405 sleeping, 2064 threads
Load Avg: 1.81, 2.46, 2.39 CPU usage: 2.95% user, 3.42% sys, 93.61% idle SharedLibs: 481M resident, 96M data, 61M linked.
MemRegions: 0 total, 0B resident, 0B private, 657M shared. PhysMem: 7533M used (978M wired, 2868M compressor), 98M unused.
Vms: 166T vsize, 17G framework vsize, 398012(0) swaptins, 505342(0) swapouts. Networks: packets: 10937520/11G in, 3648534/828M out.
Disks: 5896218/112G read, 1863534/48G written.
23:08:52

PID   COMMAND             %CPU   TIME    #TH   #WO   #PORT MEM    PURG   CMPSR   PGRP   PPID   STATE   BOOSTS   %CPU_ME %CPU_OTHRS UID   FAULTS
0      kernel_task          10.9   71:59.80 573/8 0      0      47M    0B      0B      0      0      running 0[0]     0.00000 0.00000 0      67860
5778   ifern2               7.2    01:43.34 9      6      339    144M   80K     66M-   5778   1      sleeping *0[665+] 0.05501 0.01054 501    129037+
666    corespotligh        5.1    00:56.26 6      4      317+   22M+   0B      11M-   666    1      sleeping *204[33] 0.95380 0.00000 501    638252+
7005   top                 5.0    00:00.84 1/1     0      29      6401K  0B      0B      7005   5786   running  *0[1]     0.00000 0.00000 0      6439+
6892   com.apple.We        4.3    01:15.27 7      1      79+    416M+  0B      216M-  6892   1      sleeping 0[11459] 0.00000 0.00000 501    166743+
418    WindowServer        4.0    01:43:30 21     6      2827   512M-  6080K+  110M   418    1      sleeping *0[1] 0.04217 0.44561 88     6540991+
420    analytcsd          2.5    00:20.24 3      2      400+   7233K+ 384K+   6672K- 420    1      sleeping *54[1] 0.00000 2.54531 263    142863+
568    com.apple.Ap        1.1    10:58.19 6      5      210    2784K  0B      1104K   568    1      sleeping 0[1] 0.00000 0.00000 270    52664
5516   spotlightkno       1.0    00:09.35 9      7      84+    12M+   0B      8864K- 5516   1      sleeping 0[15] 0.00000 0.95380 501    42614+
349    fseventsds         0.9    01:22.75 11     1      162    4753K  0B      2032K   349    1      sleeping *0[1] 0.00000 0.00000 0      260975+
951    com.apple.We        0.8    28:05.12 51     5      583-   144M-  15M     396M   951    1      sleeping *0[23590] 0.00000 0.00000 501    6391905
420    cfprefsd           0.7    00:22.71 3      2      658+   4049K+ 0B      1904K- 420    1      sleeping *0[10966+] 0.00000 0.69925 0      144575+
```

21. **chmod**

Modifies file permissions. Permissions control who can read, write, or execute a file. `chmod` is essential for running scripts, securing files, and managing access rights.

A terminal window with a dark background. It shows a sequence of commands and their outputs. The first command is 'ls -l random.txt', which outputs a line with permissions '-rw-r--r--', owner '1 miraj', group 'staff', date '12 10 Dec 22:55', and filename 'random.txt'. The second command is 'chmod 600 random.txt'. The third command is 'ls -l random.txt', which outputs a line with permissions '-rw-----', owner '1 miraj', group 'staff', date '12 10 Dec 22:55', and filename 'random.txt'. At the bottom, there is a terminal bar with an Apple logo icon, a folder icon, and the path '~/repos/os_assignment'.

```
> ls -l random.txt
-rw-r--r--@ 1 miraj  staff  12 10 Dec 22:55 random.txt
> chmod 600 random.txt
> ls -l random.txt
-rw-----@ 1 miraj  staff  12 10 Dec 22:55 random.txt
>
```