Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Lab Report
on
"Operating System Lab-I"

[Code No.: COMP 307]

Submitted by

Manogya Dahal
Roll No.: 14

Submitted to

Ms. Rabina Shrestha
Department of Computer Science and Engineering

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

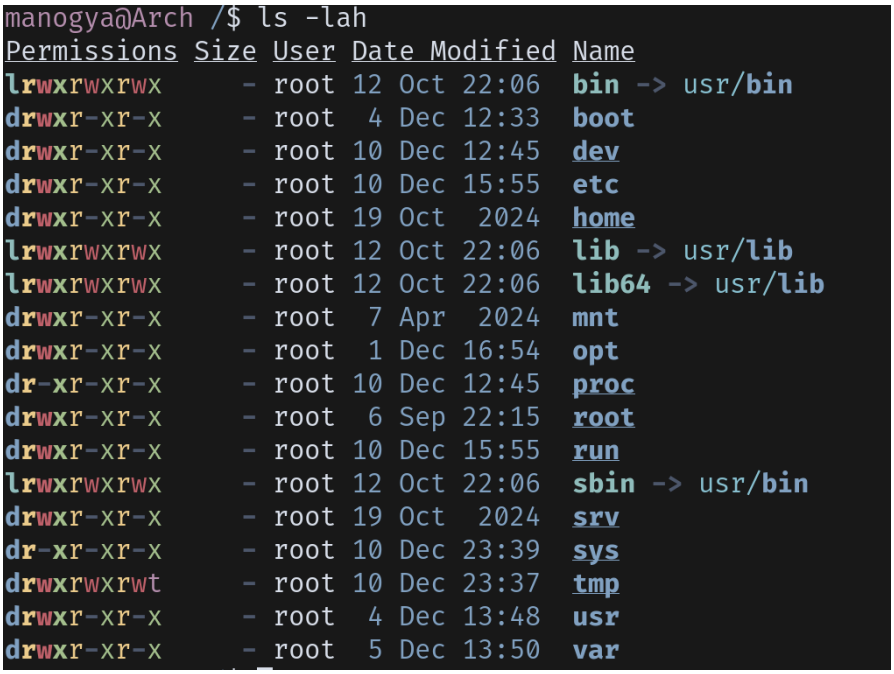


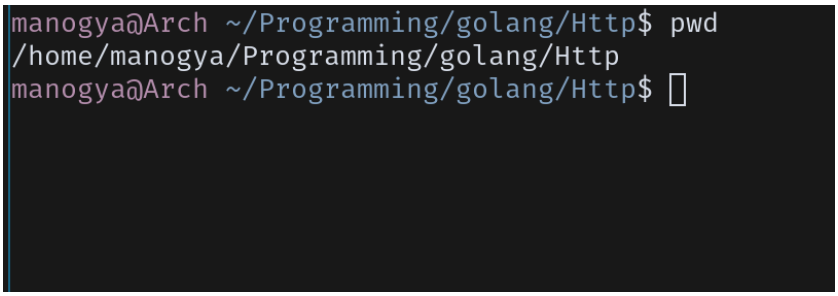
Figure 1: Root Of My Linux System

Q3: Importance of Linux commands in Operating Systems

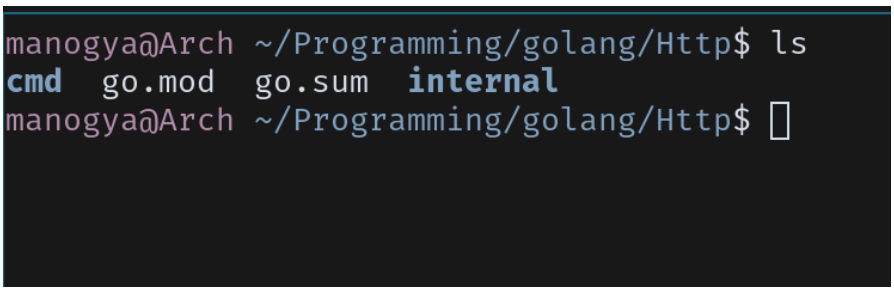
Linux commands are critical because they provide a direct and powerful interface to the operating system. They allow users and administrators to navigate the file system, manage files and directories, monitor system performance, and automate tasks through scripting. Unlike graphical interfaces, command-line 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 Commands

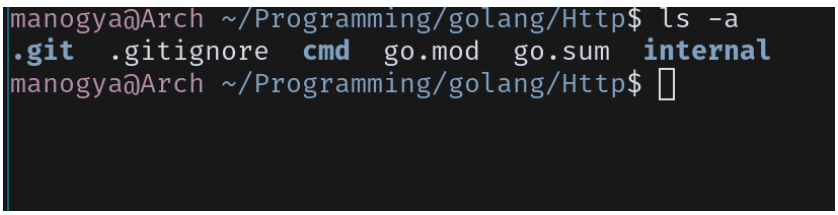
1. **`pwd`**
   The pwd command prints your current working directory. It tells you exactly where you are located inside the Linux file system. This is extremely useful when navigating through multiple folders, working in scripts, or verifying paths before executing commands that affect files. Since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



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).



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.



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.

```
manogya@Arch ~/Programming/golang/Http$ ls -l
drwxr-xr-x   - manogya 12 Oct 11:31 cmd
.rw-r--r-- 243 manogya 27 Sep 13:11 go.mod
.rw-r--r-- 782 manogya 27 Sep 13:11 go.sum
drwxr-xr-x   - manogya 12 Oct 11:32 internal
manogya@Arch ~/Programming/golang/Http$
```

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.

```
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$ pwd
/home/manogya/Programming/golang/Http
manogya@Arch ~/Programming/golang/Http$ cd cmd
manogya@Arch ~/Programming/golang/Http/cmd$ pwd
/home/manogya/Programming/golang/Http/cmd
manogya@Arch ~/Programming/golang/Http/cmd$
```

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`.

```
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$ mkdir hello
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  hello  internal
manogya@Arch ~/Programming/golang/Http$
```

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.

```
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  hello  internal
manogya@Arch ~/Programming/golang/Http$ rmdir hello
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$
```

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.

```
manogya@Arch ~/Programming/golang/Http$ mkdir new
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal  new
manogya@Arch ~/Programming/golang/Http$ rm new
rm: cannot remove 'new': Is a directory
manogya@Arch ~/Programming/golang/Http$ rm -r new
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$
```

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.

```
manogya@Arch ~/Programming/golang/Http$ mkdir new
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal  new
manogya@Arch ~/Programming/golang/Http$ rm new
rm: cannot remove 'new': Is a directory
manogya@Arch ~/Programming/golang/Http$ rm -r new
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$
```

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.

```
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  internal
manogya@Arch ~/Programming/golang/Http$ touch hello
manogya@Arch ~/Programming/golang/Http$ ls
cmd  go.mod  go.sum  hello  internal
manogya@Arch ~/Programming/golang/Http$
```

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.

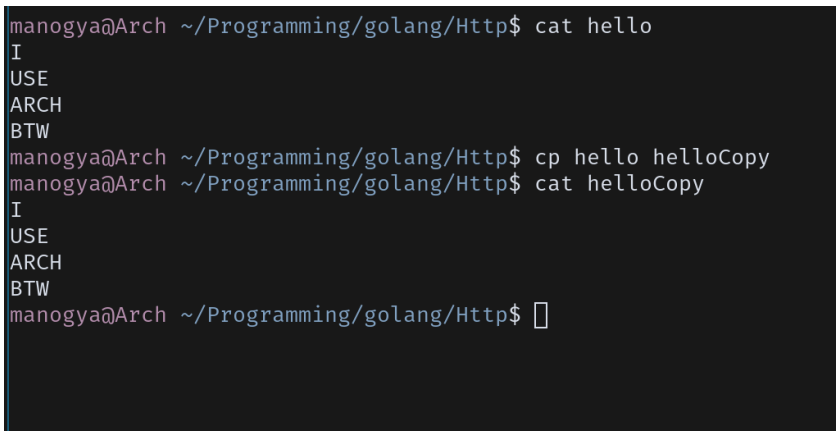
```
manogya@Arch ~/Programming/golang/Http$ cat hello
I
USE
ARCH
BTW
manogya@Arch ~/Programming/golang/Http$
```

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.

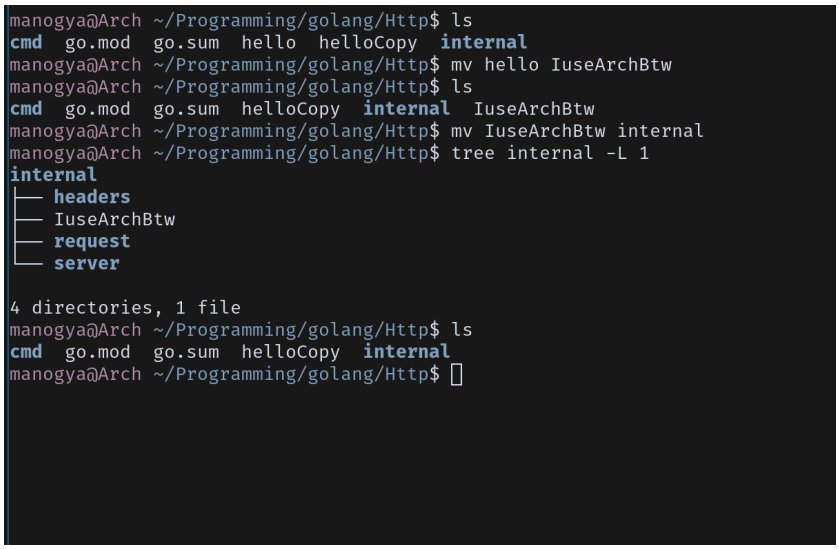
```
  1
~
~
~
~                 NVIM v0.12.0-dev-974+g2c3929624a
~
~           Nvim is open source and freely distributable
~                      https://neovim.io/#chat
~
~          type  :help nvim<Enter>       if you are new!
~          type  :checkhealth<Enter>     to optimize Nvim
~          type  :q<Enter>               to exit
~          type  :help<Enter>            for help
~
~          type  :help news<Enter> to see changes in v0.12
~
~                  Help poor children in Uganda!
~          type  :help iccf<Enter>       for information
~
~
[No Name]                                          0,0-1         All
```

13. **`cp`**
    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.



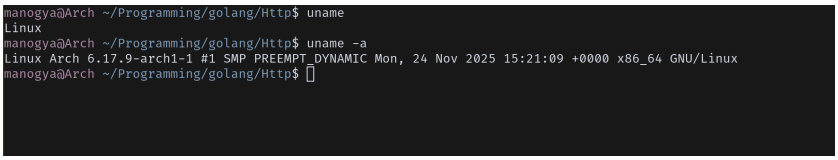
14. **mv**
    mv allows you to move or rename files and directories. Renaming is simply a move within the same folder. It's used for reorganizing or updating file names.



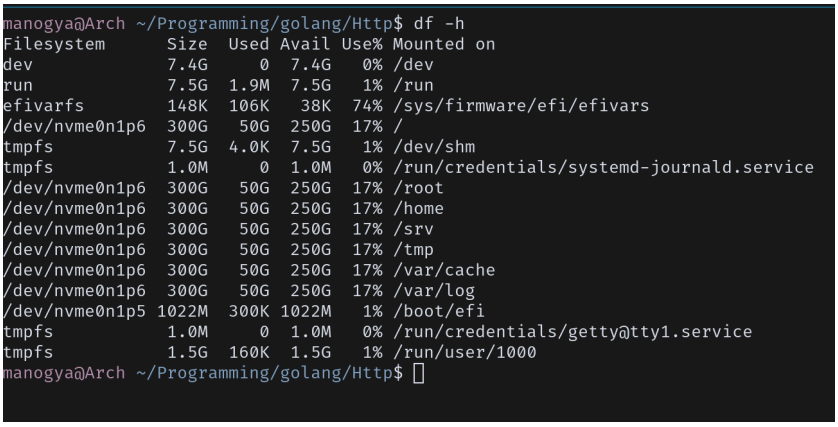
15. **echo**
    Prints text or variable values to the terminal. It is commonly used in scripts to produce messages, debug values, or write text into files using redirection.

16. **uname -a**
    Shows complete system information, including kernel version, machine architec-

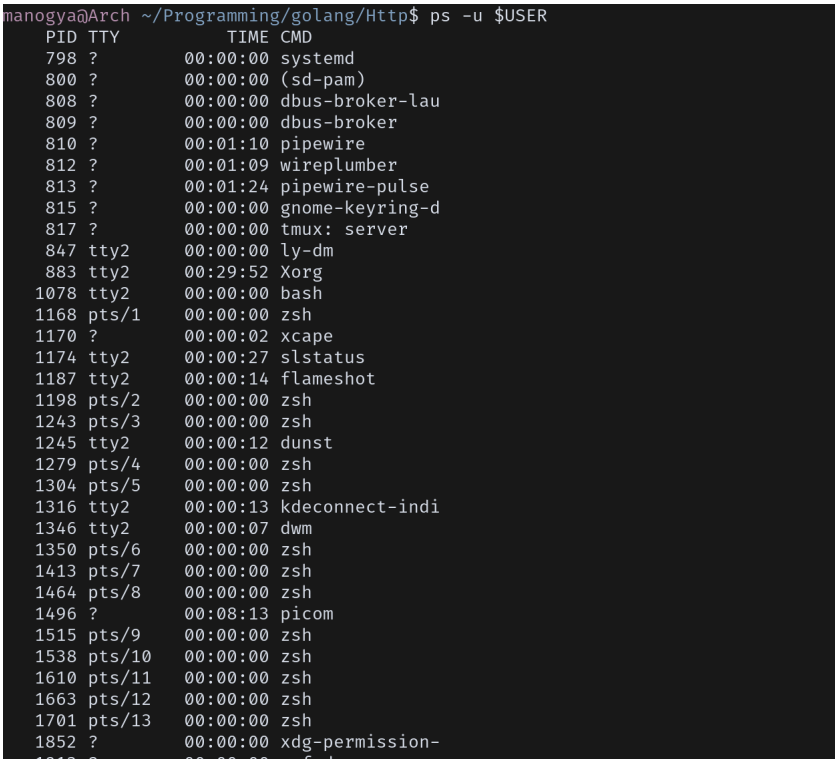
ture, hostname, and operating system. Useful for debugging or checking system specs.



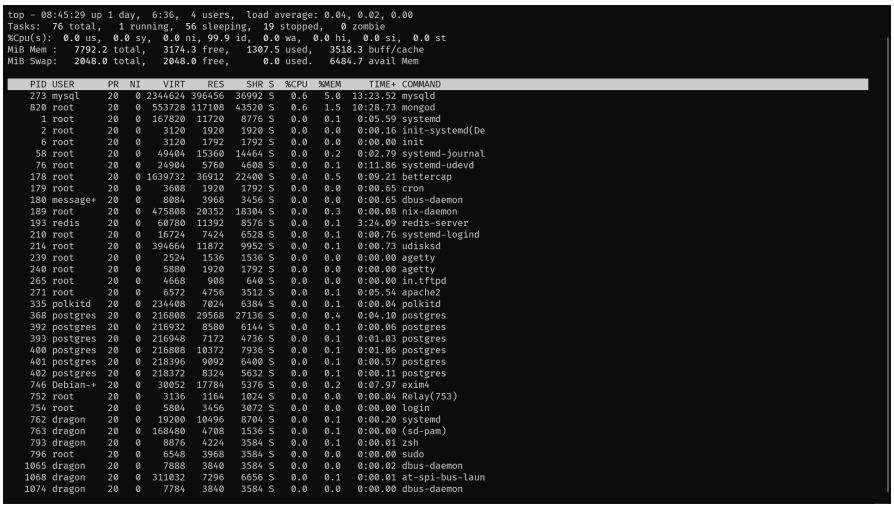
17. **`df -h`**
Displays disk usage in human-readable format (MB/GB). It shows total size, used space, available space, and mounted filesystems. Handy for monitoring storage.



18. **`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.



19. **`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.



20. **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.