What is markdown?

Markdown is a way to write text that can easily become a web page or document. It uses symbols like # to make text headers or * to format text, like making words bold or italic, or other formats with other symbols.

What is Linux?

Linux is a type of computer system that helps control how the computer works. It's like Windows or MacOS but is free and open to everyone. Many servers and phones use Linux, like Android, or web servers.

What is Ubuntu?

Ubuntu is a version of Linux. It is easy to use and free. Many people use Ubuntu to work, learn, and play on their computers. It comes in different flavors, for education, server, Studios, desktop, etc.

Lecture 2 Introduction to Linux Notes

1. What is an Operating System?

An operating system provides all fundamental software features of a computer. An OS enables you to use the computer's hardware providing the basic tools that make the computer useful.

2. What is a kernel?

An OS kernel is a software component that's responsible for managing low level features of the computer.

3. Which other parts aside from the kernel identify an OS?

Three key components of an operating system (OS) include the hardware, kernel and shell.

4. What is linux and linux distribution?

Linux is a Unix-like Operating System. Linux distribution is a complete Linux system package.

5. List at least 4 linux characteristics:

- Linux is open source software
- Linux is free of charge
- Linux is highly scalable and customizable
- You can install Linux on almost any system

6. What is Ubuntu?.

Ubuntu is a Linux distribution, freely available with both community and professional support.

7. What is Debian?

Debian is an all-volunteer organization dedicated to developing free software.

8. List and define the different types of licensing agreements

- ***Open Source:*** The software may be distributed for a fee or free. The source code is distributed with the software.
- ***Closed Source:*** The software is not distributed with the source code. The user is restricted from modifying the code.
- ***Freeware:*** The software is free but the source code is not available.
- ***Shareware:*** The software is free on a trial basis.
- ***Free software:*** The software is distributed with the source code. The software can be free of charge or obtained by a free.

9. ## What is Free Software? Define the 4 freedoms.

Free software is a critical force in the open source world.

- ***Freedom 0:*** Use the software for any purpose
- ***Freedom 1:*** Examine the source code and modify as you see fit
- ***Freedom 2:*** Redistribute the software
- ***Freedom 3:*** Redistribute your modified software

10. ## What is virtualization?

Virtualization is a technology that creates virtual versions of physical hardware, operating systems, and storage devices.

Linux Commands Overview

1. *echo*

Definition: Prints text to the terminal.

Usage:

```
echo "Hello, World!"
```

Example Output:

```
Hello, World!
```

---

2. *date*

Definition: Displays the current date and time.

Usage:

```
date
```

Example Output:

```
Sun Mar 2 12:00:00 UTC 2025
```

---

3. *free*

Definition: Shows memory usage details.

Usage:

```
free -h
```

Example Output:

```
```

```
               total        used        free      shared  buff/cache
available
Mem:           6.0G        2.0G        2.5G        0.5G        1.5G
3.0G
Swap:          2.0G        0.5G        1.5G
```

---

4. *uname*

Definition: Displays system information.

Usage:

```
uname -a
```

Example Output:

```
Linux myserver 5.15.0-25-generic #26-Ubuntu SMP x86_64 GNU/Linux
```

---

5. *history*

Definition: Shows the list of previously executed commands.

Usage:

```
history
```

Example Output:

```
1  ls
2  cd /home
3  echo "Test"
4  history
```

---

6. *man*

Definition: Displays the manual for a command.

Usage:

```
man ls
```

Example Output:

```
LS(1) User Commands LS(1)
NAME
     ls - list directory contents
```

---

7. *apt*

Definition: Package manager for Debian-based systems.

Usage:

```
sudo apt update && sudo apt upgrade -y
```

Example Output:

```
Reading package lists... Done
Building dependency tree... Done
```

---

8. *snap*

Definition: Package management system for Snap applications.

Usage:

```
snap list
```

Example Output:

```
Name           Version        Rev   Tracking       Publisher      Notes
core           16-2.51.1      11167 latest/stable  Canonical✓     core
```

---

9. *flatpak*

Definition: Universal package management system for Linux.

Usage:

```
flatpak list
```

Example Output:

```
Name     Application ID               Version Branch
Firefox  org.mozilla.firefox          96.0.2  stable
```

File path commands

pwd

Used for displaying the current working directory.

Examples:

- ***pwd***: – Shows the current working directory.
- ***pwd && ls8***: – Displays the current directory and lists its contents.
- ***pwd > current_dir.txt***: – Saves the current directory path to a file.

cd

Used for changing the current working directory.

Examples:

- ***cd Downloads***: – Changes directory to the "Downloads" directory.
- ***cd ../***: – Goes back one directory (*/usr/share/themes → /usr/share*).
- ***cd***: – Goes to your home directory.

ls

Used for displaying the files inside a given directory.

Examples:

- ***ls -a***: – Shows all files in the current directory, including hidden ones.
- ***ls***: – Shows all content in the present working directory.
- ***ls -lR ~/Pictures***: – Long list of all files inside a given directory recursively.

tree

Used for displaying the directory structure in a tree format.

Examples:

- ***tree*** – Displays the current directory structure in a tree format.
- ***tree ~/Documents*** – Shows the tree structure of the "Documents" directory.
- ***tree -L 2*** – Limits the depth of the tree display to two levels.

Definitions

File system

The way files are stored and organized.

pathname

Indicates the location of the file in a file system.

Absolute path

Location of a file starting at the root of the file system.

Relative path

Location of a file starting from the current working directory or a directory that is located inside the current working directory.

The difference between your home directory and the home directory

Your home directory holds all of your personal files, while The home directory is the parent of all the home directories.

parent directory

A directory that contains one or more directories and files

child directory or subdirectory

A directory inside another directory.

Bash special characters

They function like commands that tell the shell to perform a specific action without having to type the complete command.

environment variables

They store values of a user’s environment and can be used in commands in the shell.

user defined variables

Variables defined by the user and exist only in the script and subshell that runs the script.

Why do we need use $ with variables in bash shell scripting?

They are used to store user variables.

Notes 5

`mkdir`

Usage:
Creates one or more directories.

Formula:
`mkdir [option] [directory_name]`

Examples:

- `mkdir projects`
  Creates a directory named `projects`.
- `mkdir -p projects/java`
  Creates a parent directory and a subdirectory.
- `mkdir -v mydir1 mydir2 mydir3`
  Creates multiple directories and displays each creation.

`touch`

Usage:
Used to create empty files or update file timestamps.

Formula:
`touch [file_name]`

Examples:

- `touch notes.txt`
  Creates an empty file named `notes.txt`.
- `touch file1.txt file2.txt file3.txt`
  Creates multiple files at once.
- `touch ~/Desktop/logs/output.log`
  Creates a file at a specific path.

`rm`

Usage:
Deletes files and directories.

Formula:
`rm [option] [file_or_directory]`

Examples:

- `rm file1.txt`
  Deletes a single file.
- `rm -i important.txt`
  Prompts before deleting the file.
- `rm -r old_projects`
  Recursively removes a directory and its contents.

---

rmdir

Usage:
Deletes empty directories only.

Formula:
`rmdir [option] [directory_name]`

Examples:

- `rmdir temp`
  Removes an empty directory named `temp`.
- `rmdir -p work/project/final`
  Removes nested directories if they're all empty.
- `rmdir -v folder1 folder2`
  Removes multiple directories and shows a message for each.

---

mv

Usage:
Moves or renames files and directories.

Formula:
`mv [source] [destination]`

Examples:

- `mv report.docx archive/`
  Moves `report.docx` into the `archive` folder.
- `mv photo1.jpg photo_backup.jpg`
  Renames the file.
- `mv file1 file2 folder/`
  Moves multiple files into a folder.

---

cp

Usage:
Copies files or directories from one location to another.

Formula:
`cp [option] [source] [destination]`

Examples:

- `cp todo.txt backup/`
  Copies a file to another directory.
- `cp -r projects/ projects_backup/`
  Copies a directory and its contents recursively.
- `cp *.jpg images/`
  Copies all `.jpg` files to the `images` folder.

---

file

Usage:
Displays the file type of a specified file.

Formula:
`file [option] [file_name]`

Examples:

- `file script.sh`
  Shows the type of the file `script.sh`.
- `file -b image.png`
  Displays the file type only (no filename).
- `file /etc/*`
  Lists the type of each file in the `/etc` directory.

Notes 6: Wildcards and Brace Expansion

What Are Wildcards?

Wildcards are special characters used to match filenames or directory names. They help perform actions on multiple files without having to type each name individually.

Wildcard Table

Wildcard	Definition	Example
`*`	Matches 0 or more characters	`ls *.txt` lists all `.txt` files
`?`	Matches exactly 1 character	`ls f?le.txt` matches `file.txt`, `f1le.txt`, etc.
`[]`	Matches 1 character from a set	`ls [ab]*` matches files starting with `a` or `b`

Usage Examples

`*` Wildcard (zero or more characters)

1. `ls *.sh`
   Lists all files ending in `.sh`.

2. `ls My_*`
   Lists files starting with `My_`.

3. `ls *program*`
   Lists files containing the word `program`.

`?` Wildcard (exactly one character)

1. `ls file?.txt`
   Matches `file1.txt`, `file2.txt`, but not `file10.txt`.

2. `ls ?.sh`
   Matches files like `a.sh`, `b.sh`, but not `ab.sh`.

3. `ls a?.txt`
   Matches `a1.txt`, `ab.txt`, etc.

`[]` Wildcard (character from a set or class)

1. `ls [A-Z]*`
   Lists files that start with an uppercase letter.

2. `ls *[0-9]*`
   Lists files containing numbers.

3. `ls *[[:punct:]]*`
   Lists files containing punctuation characters.

POSIX Character Classes (inside `[]`)

POSIX Class	Equivalent	Matches	Example
`[:upper:]`	`[A-Z]`	Uppercase letters	`ls *[[:upper:]]*`
`[:lower:]`	`[a-z]`	Lowercase letters	`ls *[[:lower:]]*`
`[:digit:]`	`[0-9]`	Digits	`ls *[[:digit:]]*`
`[:alpha:]`	`[A-Za-z]`	Alphabetic characters	`ls *[[:alpha:]]*`
`[:alnum:]`	`[A-Za-z0-9]`	Alphanumeric characters	`ls *[[:alnum:]]*`
`[:punct:]`	punctuation	Punctuation marks	`ls *[[:punct:]]*`

Brace Expansion `{}`

Brace expansion is used to generate arbitrary strings.

Examples:

1. `echo file{1,2,3}.txt`
   Expands to: `file1.txt file2.txt file3.txt`

2. `mkdir project_{A,B,C}`
   Creates: `project_A`, `project_B`, `project_C`

3. `cp file{A,B}.txt`
   Copies `fileA.txt` and `fileB.txt`

When Should I Use Wildcards?

I can use wildcards when I want to work with many files at once:

- Copy all `.jpg` images to a flash drive:
  `cp *.jpg /media/usb`

- Delete all temporary files:
  `rm *.tmp`

- List all files that include a date:
  `ls *2024*`