Practical 1

1. Introduction to LINUX environment and related system programming.

- **VirtualBox** is a free and open-source virtualization software developed by Oracle. It allows you to run multiple operating systems (OS) on a single physical machine by creating virtual machines (VMs).

Download VirtualBox For Windows

1. Go to the official VirtualBox website: https://www.virtualbox.org/
2. Click on the "Download VirtualBox" link.
3. Select **Windows hosts** and download the .exe installer for your version of Windows (e.g., VirtualBox x.x.x for Windows hosts).

Install VirtualBox:

- Once the download is complete, **run the installer**.
- Follow the on-screen instructions.
- Click **Next** on the initial setup screen.
- Choose the installation options based on your preference (you can usually leave the default options as is).
- Click **Install** to begin the installation process.
- You might be prompted to allow some network features or install device drivers, so click **Yes** or **Allow** when prompted.
- Wait for the installation to finish, and click **Finish** to complete the process.

- Download Kali Linux

Kali Linux comes with a wide range of pre-installed tools that are essential for tasks such as penetration testing, vulnerability, scanning, cybersecurity, digital forensics, network monitoring, and more.

Download Kali Linux ISO

- Go to the official Kali Linux website: https://www.kali.org/downloads/
- Download the appropriate **Kali Linux ISO** file for your system (e.g., 64-bit or 32-bit version).

- Cisco Packet Tracker

Cisco Packet Tracer is a network simulation software developed by Cisco Systems. It allows users to create, configure, and simulate network topologies without the need for physical hardware.

Step 1: Create a Cisco Networking Academy Account

- Go to the **Cisco Networking Academy** website: https://www.netacad.com/.
- **Sign up** for a free account if you don't already have one, or **log in** if you already have an account.

Step 2: Download Cisco Packet Tracer

- After logging into your **Cisco Networking Academy** account, go to the **Resources** section.
- Navigate to **Packet Tracer** and click on the **Download** option.
- Choose the correct version for your operating system (Windows, macOS, or Linux).
- Download the **installer file** for your platform (e.g., .exe for Windows or .dmg for macOS).

Step 3: Install Cisco Packet Tracer

For Windows:

· **Download and Run Installer**:

· Locate and double-click the .exe file.

· **Select Language**:

· Choose your language (usually English) and click **OK**.

· **Follow Wizard**:

· Click **Next**, accept the License Agreement, and click **Next**.

· **Choose Installation Folder**:

· Select the default folder or choose your desired one, then click **Next**.

· **Start Installation**:

· Click **Install** and wait for it to complete.

· **Finish**:

· Click **Finish** and optionally launch Packet Tracer.

For Online Practice You can follow these link

Platforms to Try:

- **OnlineGDB**: https://www.onlinegdb.com
- **Replit**: https://replit.com
- **JDoodle**: https://www.jdoodle.com

Practical2

Explore directory structure of Linux File System. Understand and represent file system of LINUX with brief details.sss

The Linux file system follows a hierarchical directory structure. All directories are organized into a single root (/) directory, and all files and directories in Linux are contained under this root directory. The structure is tree-like and well-organized, making it easy to navigate and maintain. Let's explore the main directories that are typically found in the Linux file system.

Key Directories in the Linux File System:

- **/ (Root Directory):** The top-most directory from which all other directories and files branch out.
- **/bin:** Contains essential executable binaries required for the system to boot and operate (e.g., ls, cp).
- **/boot:** Holds boot loader files, including the kernel image and boot configuration files.
- **/dev:** Contains device files representing hardware devices (e.g., disk drives, input devices).
- **/etc:** Stores system-wide configuration files, such as user account details (/etc/passwd) and system settings.
- **/home:** Home directories for individual users to store their personal files (e.g., /home/user1).
- **/lib:** Contains shared libraries needed by system binaries and user programs for execution.
- **/media:** Temporary mount points for removable media like USB drives or CDs.
- **/mnt:** A directory typically used for temporarily mounting external file systems (e.g., network drives).
- **/opt:** Optional application software packages not typically part of the core Linux system.
- /proc: A virtual file system that provides information about running processes and system parameters.
- /root: The home directory for the root user (administrator).
- /run: Contains runtime data for processes, including PID files and lock files.
- /sbin: System binaries for system administrators (e.g., fsck, reboot).
- /srv: Data files for services provided by the system (e.g., web server data).

- /sys: Virtual file system providing information about the system and kernel (e.g., /sys/block for block devices).
- /tmp: Temporary storage for files that can be cleared after system reboot.
- /usr: Contains user-related programs and files, including user binaries (/usr/bin), libraries (/usr/lib), and documentation (/usr/share).
- /var: Stores variable data that changes during system operation (e.g., log files, spool files).

Note: For Reference : Linux Directory Structure - GeeksforGeeks

Practical 3

Perform basic commands of Linux. Demonstration of “vi” editor.

1. ls - The most frequently used command in Linux to list directories
2. pwd - Print working directory command in Linux
3. cd - Linux command to navigate through directories
4. mkdir - Command used to create directories in Linux
5. mv - Move or rename files in Linux
6. cp - Similar usage as mv but for copying files in Linux
7. rm - Delete files or directories
8. touch - Create blank/empty files
9. ln - Create symbolic links (shortcuts) to other files
10. cat - Display file contents on the terminal
11. clear - Clear the terminal display
12. echo - Print any text that follows the command
13. less - Linux command to display paged outputs in the terminal
14. man - Access manual pages for all Linux commands
15. diff - Find the difference between two files
16. zip - Zip files in Linux
17. unzip - Unzip files in Linux
18. ssh - Secure Shell command in Linux
19. service - Linux command to start and stop services
20. ps - Display active processes
21. df - Display disk filesystem information
22. mount - Mount file systems in Linux
23. chmod - Command to change file permissions
24. chown - Command for granting ownership of files or folders
25. ifconfig - Display network interfaces and IP addresses
26. wget - Direct download files from the internet
27. sudo - Command to escalate privileges in Linux

28. alias - Create custom shortcuts for your regularly used commands

29. whereis - Locate the binary, source, and manual pages for a command

30. passwd - Create or update passwords for existing users

Practical 4

4. Introduction of various networking equipment.
Various networking equipment serves specific roles in connecting devices, ensuring secure data transmission, and optimizing performance across networks. Some key devices include:

- **Router**: Routes data between networks.
- **Switch**: Connects devices within the same network.
- **Modem**: Converts digital data to analog for transmission.
- **NIC**: Connects devices to a network.
- **Access Point**: Provides wireless access to a network.
- **Firewall**: Protects the network from unauthorized access.
- **Gateway**: Connects networks with different protocols.
- **Repeater**: Extends signal range.

Note: We have already covered this topic in our Unit 1 lecture.
for more reference: Network Devices (Hub, Repeater, Bridge, Switch, Router, Gateways and Brouter) - GeeksforGeeks

Practical 5

Introduction to pipes and related system calls for pipe management. Write a program to create a pipe and send “Hello” message.

A **pipe** is a mechanism used in inter-process communication (IPC) that allows data to be passed from one process to another. Pipes are unidirectional, meaning data flows in one direction only. In Linux, pipes are commonly used to connect the output of one process to the input of another.

Key System Calls for Pipe Management

pipe(): int pipe(int pipefd[2]);//Creates a pipe, which consists of two file descriptors: one for reading and one for writing.

write():ssize_t write(int fd, const void *buf, size_t count);// Writes data to a file or file descriptor (in this case, to the pipe's write end).

read():ssize_t read(int fd, void *buf, size_t count);//Reads data from a file or file descriptor (in this case, from the pipe's read end)

fork():pid_t fork(void); //Creates a new process by duplicating the calling process

exit():void exit(int status);//Terminates the calling process and returns a status code.

C Program to Create a Pipe and Send "Hello" Message

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main() {
    int pipefd[2];  // Array to store the pipe file descriptors
    char message[] = "Hello from pipe!";
    char buffer[100];

    // Create the pipe
    if (pipe(pipefd) == -1) {
        perror("pipe");
        exit(1);
    }

    // Write to the pipe
    write(pipefd[1], message, strlen(message) + 1);  //+1 to include the null terminator

    // Read from the pipe
    read(pipefd[0], buffer, sizeof(buffer));

    // Print the message read from the pipe
    printf("Message from pipe: %s\n", buffer); //placeholder is where the contents of
buffer will be inserted.

    // Close the pipe file descriptors
    close(pipefd[0]);
    close(pipefd[1]);

    return 0;
}
```

Hello print program:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>  // For exit() function
#define MSGSIZE 16
char* msg1 = "hello friends";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";
int main()
{
    char inbuf[MSGSIZE];  // inbuf is a character array
    int p[2], i;  // Array of two elements, p[0] for reading from the pipe and p[1] for writing to the pipe.
    if (pipe(p) < 0)
        exit(1);  // If pipe creation fails, exit the program.
   /* Write to pipe */
    write(p[1], msg1, MSGSIZE);
    write(p[1], msg2, MSGSIZE);
    write(p[1], msg3, MSGSIZE);
    for (i = 0; i < 3; i++) {
        /* Read from pipe */
        read(p[0], inbuf, MSGSIZE);
        printf("%s\n", inbuf);  // Correct format specifier to print the string
    }
    return 0;
}
```