DEPARTMENT OF INFORMATION TECHNOLOGY

LABORATORY MANUAL

SUBJECT: OPERATING SYSTEM LAB

BRANCH: 4TH SEM, IT

FACULTY NAME: DR. ANITARANI BRAHMA

JHARSUGUDA ENGINEERING SCHOOL, JHARSUGUDA

Vision:

To focus on development of skilled and confident personalities of today and tomorrow by using cutting edge technology in the department of Information Technology to accept need based challenges with a sense of social responsibilities.

Mission:

- M1: To impart quality education by implementing state-of- the- art teaching-learning methods to enrich the academic competency, credibility and integrity of the students.
- M2: To implement the educational program in our department from fundamental engineering to recent technology as per emerging trends in the field of Information Technology.
- M3: To facilitate a conducive ambience and infrastructure to develop professional skills and nurture innovation in students.
- M4: To inculcate sensitivity towards society, respect for environment and promote high standards of ethics.

Learning Objectives:

- Explain fundamental operating system concepts, including UNIX/Linux architecture, processes, memory management, and file systems.
- Demonstrate the use of system calls, inter-process communication (IPC) mechanisms, and process scheduling techniques in practical scenarios.
- Analyze memory allocation techniques, virtual memory concepts, and disk scheduling algorithms to assess their impact on system performance.
- Evaluate file system implementation techniques and security measures like authentication, access control, and system logs for maintaining system integrity.
- Design a system simulation that integrates process management, memory allocation, and disk scheduling to demonstrate OS functionality.

Introduction

Experiment: Install Linux in Windows

Hardware & Software Requirement:

Computer, installed Windows O.S., internet connection

Theory: We can install Linux on Windows in different ways, depending on our needs. Here are the best options:

1. Using Windows Subsystem for Linux (WSL)

WSL lets you run a Linux environment directly on Windows without needing a virtual machine or dual boot.

2. Using a Virtual Machine (VM) – Good for GUI Linux

If we want a full Linux desktop, use a virtual machine.

3. Dual Boot – Full Linux Installation

This installs Linux alongside Windows and lets you choose which OS to boot.

Steps to Create an Ubuntu Virtual Machine in VMware Workstation

1 Open VMware Workstation

- Launch **VMware Workstation**
- Click **Create a New Virtual Machine**

---

2 Choose Installation Method

- Select **Typical (recommended)**
- Click **Next**

---

3 Select Ubuntu ISO File

- Choose **Installer disc image file (iso)**
- Click **Browse**
- Select your downloaded **Ubuntu .iso file**
- Click **Next**

 VMware may auto-detect Ubuntu — that’s normal.

4️⃣ Easy Install (If Prompted)

- Enter:
  - **Full name** (anything)
  - **Username**
  - **Password**
- Click **Next**

(If you want manual install, uncheck "Easy Install")

5️⃣ Name the Virtual Machine

- Give a name like: **Ubuntu 22.04**
- Choose a location (default is fine)
- Click **Next**

6️⃣ Allocate Disk Space

- Recommended:
  - **20–25 GB** minimum
- Select **Store virtual disk as a single file**
- Click **Next**

7️⃣ Customize Hardware (Important)

Click **Customize Hardware** and set:

- **Memory (RAM):**
  - Minimum: **2 GB**
  - Recommended: **4 GB** (if your PC allows)
- **Processors:**
  - 1 processor, 2 cores (or default)
- Click **Close**

Then click **Finish**

8️⃣ Start Ubuntu Installation

- Click **Power on this virtual machine**
- Ubuntu will start installing automatically
- Wait for installation to complete (5–15 minutes)

9️⃣ Login to Ubuntu

- After installation, Ubuntu will reboot
- Log in using the username & password you created

How to Open Terminal to run Linux command

Method 1: Open Terminal

Press:

Ctrl + Alt + T

➡ This opens the **Terminal window** instantly.

Method 2: Using Mouse (GUI way)

1. Click **Show Applications** (bottom-left corner, 9 dots)
2. Search for **Terminal**
3. Click **Terminal**

Conclusion: In this experiment, we have successfully learned the procedure to install Linux in Windows platform.

Experiment 1

(Introduction to Linux Commands related to File Operation & Text Processing)

Experiment 1.1

Aim: Experimenting with basic Linux & Terminal Commands

Experiment: Running of basic Linux commands

Hardware & Software Requirement:

Computer, installed Windows O.S., VMware

Description:

Command	Description
pwd	Shows the current directory path.
ls	Lists files and directories.
ls -l	Lists files with details (permissions, owner, size, date).
tree	Used to **display the directory structure in a tree-like (hierarchical) format**, making it easy to understand how files and folders are organized
cd <directory>	Changes directory (e.g., cd Documents).
cd ..	Moves to the previous directory.
mkdir<dir>	Creates a new directory.
rmdir<dir>	Removes an empty directory.
rm <file>	Deletes a file.
rm -r <dir>	Deletes a directory and its contents.

touch <file>	Creates a new empty file
cp <source><destination>	Copies a file.
mv <source><destination>	Moves or renames a file.

File Permission

Command	Description
Whoami	Displays the current user.
Who	Shows all logged-in users.
chmod<permissions><file>	Changes file permissions.
chown<user>:<group><file>	Changes file owner.
Passwd	Changes user password.

Conclusion: In this experiment, we have successfully learned different commands to manage files and

Experiment 1.2

(File Operations & Text Processing)

Experiment: Running of commands related to file operations & text processing

Hardware & Software Requirement:

Computer, installed Windows O.S., VMware

Description:

COMMAND	DESCRIPTION
`cat <file>`	Displays the contents of a file.
`nano <file>`	Opens a file in the **Nano** text editor.
`vim <file>`	Opens a file in the **Vim** text editor.
`head -n <N><file>`	Shows the first N lines of a file.
`tail -n <N><file>`	Shows the last N lines of a file.
`grep`	**searching plain-text data for lines that match a specific pattern** and printing the matching lines to standard output
`Awk` `$ awk '{print}' employee.txt`	It reads files line by line, applies patterns, and performs specified actions on matching lines. By default Awk prints every line of data from the specified file.
`Sed` **`sed [OPTIONS] 'COMMAND' [INPUTFILE...]`**	'OPTIONS': These are optional flags that modify the behavior of the sed command. 'COMMAND': This defines the command or sequence of commands to execute on the input file. 'INPUTFILE': One or more input files to be processed. Options: -i Edit the file in-place (overwrite) -n Suppress automatic printing of lines. -e Allows multiple commands. -f Reads sed commands from a file. -r Use extended regular expressions.
`Cut`	The cut command is used to **extract specific columns, characters, or fields** from each line of a file. cut -c 1-5 file.txt # Extract characters 1 to 5 cut -d ":" -f 1 file.txt # Extract first field using ':' delimiter
`Sort`	The sort command is used to **arrange lines of a file in ascending or descending order**. sort file.txt # Sort alphabetically sort -r file.txt # Reverse order sort -n marks.txt # Numeric sort
`Uniq`	The uniq command is used to **remove or identify duplicate**

	lines in a sorted file. uniq file.txt # Remove duplicate lines uniq -c file.txt # Count occurrences uniq -d file.txt # Show only duplicates
Diff	The diff command is used to **compare two files line by line** and show the differences. diff file1.txt file2.txt diff -y file1 file2 # Side-by-side comparison
Redirection(>,>>,|,<,tee)	Redirection is used to **change the standard input, output, or error stream** of a command. > Redirect output (overwrite) >> Redirect output (append) < Redirect input 2> Redirect error &> Redirect output + error

Conclusion: In this experiment, we have successfully learned different commands related to File Operations & Text Processing

Experiment 2

(Process Management)

Experiment: Running of process management commands

Hardware & Software Requirement:

Computer, installed Windows O.S., VMware

Description:

Command	Description	Syntax	Example
Process management Command			
ps	Displays information about active processes.	ps	
top	Displays real-time system processes and CPU usage.	top	
htop	Interactive process viewer similar to top with improved interface.	htop	Displays colorful and user-friendly process monitoring interface.
kill <PID>	Terminates a process using Process ID (PID).	kill [PID]	kill 1234
nice	Starts a process with a specified priority. Nice values range from -20 to +19. A lower nice value means higher scheduling priority.	nice -n [priority] command	nice -n 10 nano file.txt
renice	Changes priority of running process.	renice [priority] -p [PID]	renice 5 -p 1234
Job Control Command			
jobs	Displays list of background jobs.	jobs	Jobs Displays background job list with job numbers.
df -h	Shows disk space usage.		
fg	Brings background job to foreground.	fg %[job_number]	fg %1 Selected job runs in foreground.
bg	Resumes suspended job in background.	bg %[job_number]	Suspended job resumes in background.
Task Scheduling Commands			
Cron	Schedules repetitive tasks automatically.	crontab –e	crontab -e Add line: 0 9 * * * echo "Hello" >> file.txt Task executes daily at 9:00 AM. **Cron Time Format** Minute Hour Day Month Weekday Command

Command	Description	Syntax	Example
Process management Command			
At	Schedules one-time task execution.	at [time]	
User and Group management			
useradd	Creates new user account.	useradd [username]	useradd student11
usermod	Modifies existing user account.	usermod [options] [username]	usermod -aG staff student1 User added to specified group.
passwd	Sets or changes user password	passwd [username]	passwd student1
groupadd	Creates new group.	groupadd [groupname]	groupadd labgroup
gpassword	Manages group membership.	gpasswd -a [username] [groupname]	gpasswd -a student1 labgroup User added to group.
File Ownership and permissions			
chown	Changes file ownership.	chown [owner] [filename]	chown student1 file.txt
chmod	Changes file permissions.	chmod [permissions] [filename]	chmod 755 file.txt r = Read = 4 w = Write = 2 x = Execute = 1
umask	Sets default file permission mask.	umask [value]	umask 022 Default permission settings changed.

Conclusion: In this experiment, we have successfully learned different commands related to process management

Experiment 3

(Package Management)

Experiment: Running of package management commands

Hardware & Software Requirement:

Computer, installed Windows O.S., VMware

Description:

Command	Description	Syntax	Example
Apt	Used in Debian-based systems to install and manage packages	apt install [package_name] apt remove [package_name] apt update apt upgrade	apt update apt install nano apt remove nano
Yum	Used in Red Hat-based systems to manage packages.	yum install [package_name] yum remove [package_name] yum update	yum install httpd
Dnf	Next-generation package manager for RPM-based distributions.	dnf install vim	
snap	Installs universal Linux packages called snaps.	snap install [package_name] snap remove [package_name]	snap install vlc
Networking Commands			
ifconfig	Displays and configures network interfaces.	ifconfig	
Ip	Displays IP address and routing information.	ip addr	ip addr show
Ping	Tests connectivity between systems.	ping [hostname/IP]	ping google.com
Netstat	Displays network connections and statistics.	netstat -a	netstat -tuln
Ss	Displays socket statistics.	ss	ss -tuln
Traceroute	Shows route taken by packets to destination.	traceroute [hostname]	traceroute google.com
Wget	Downloads files from the internet.	wget [URL]	wget http://example.com/file.txt
curl	Transfers data from or to a server.	curl [URL]	curl http://example.com
Remote Access Commands			
ssh	Used to connect securely to remote system.	ssh username@hostname	ssh student@192.168.1.10
scp	Copies files securely between systems.	scp file.txt username@hostname:/path	scp data.txt student@192.168.1.10:/home/student
rsync	Synchronizes files between systems efficiently.	rsync -av source destination	rsync -av file.txt student@192.168.1.10:/home/student

Conclusion: In this experiment, we have successfully learned different commands related to package management.

Experiment 4

(Disk & Storage Management)

Experiment: Running of Disk & Storage Management commands

Hardware & Software Requirement:

Computer, installed Windows O.S., VMware

Description:

df

Description: Shows disk usage.

Syntax: df -h

Example: df -h

du

Description: Shows directory size.

Syntax: du -sh dir

Example: du -sh /home

mount

Description: Mounts device.

Syntax: mount device dir

Example: mount /dev/sdb1 /mnt

umount

Description: Unmounts device.

Syntax: umount dir

Example: umount /mnt

fdisk

Description: Creates partitions.

Syntax: fdisk device

Example: fdisk /dev/sdb

mkfs

Description: Creates file system.

Syntax: mkfs.ext4 device

Example: mkfs.ext4 /dev/sdb1

fsck

Description: Checks filesystem.

Syntax: fsck device

Example: fsck /dev/sdb1

Conclusion: In this experiment, we have successfully learned different commands related to Disk & Storage Management command

Experiment 5

(Programming with Shell Scripting)

Experiment: Running Shell Scripting Programs

Hardware & Software Requirement:

Computer, installed Windows O.S., Gitbash

Description:

Shell Scripting:

We can run Shell program in Windows in different ways, depending on our needs. Here are the best options:

1. Using Command Prompt (CMD)

- Open **Command Prompt** (`Win + R`, type `cmd`, and press `Enter`).
- Navigate to the directory containing the script:

```
sh
CopyEdit
cd path\to\your\script
```

- Run the script using:

```
sh
CopyEdit
sh script.sh
```

- If you don't have a shell installed, you'll need **Git Bash** or **WSL**.

2. Using PowerShell

- Open **PowerShell** (`Win + R`, type `powershell`, and press `Enter`).
- Run the script with:

```
powershell
CopyEdit
./script.sh
```

- If that doesn't work, use:

```
powershell
CopyEdit
bash script.sh
```

- PowerShell doesn't natively support `.sh` scripts, so you need **WSL** or **Git Bash**.

3. Using Git Bash (Recommended for Shell Scripts)

- Install Git for Windows.
- Open **Git Bash** and navigate to the script folder.
- Run the script:

```
sh
CopyEdit
./script.sh
```

1. Write a Shell script to print a message (file name- 1.sh)

```
#!/bin/bash
echo "Hello from Git Bash!"
```

2. Write a Shell script to print the command line arguments in reverse order. (file name- 2.sh)

Algorithm:

1. Start
2. Read command line arguments
3. Use a loop to print them in reverse order
4. End

Code:

```
#!/bin/bash

# Get number of arguments
n=$#

# Loop in reverse order
for (( i=$n; i>0; i-- ))
do
echo ${!i}
done
```

Then in command line type

./reverse_args.sh one two three four

2. Write a Shell script to check whether the given number is palindrome or not.

Algorithm:

1. Start
2. Read a number
3. Reverse the number
4. Compare with the original number
5. Print if palindrome or not
6. End

Code:

```
#!/bin/bash


echo "Enter a number"
read num
```

```
rev=0
temp=$num

while [ $temp -gt 0 ]
do
    digit=$((temp % 10))
    rev=$((rev * 10 + digit))
    temp=$((temp / 10))
done

if [ $num -eq $rev ]
then
    echo "Palindrome"
else
    echo "Not Palindrome"
fi
```

3. Write a Shell script to sort the given array elements in ascending order using bubble sort.

Code:

```
#!/bin/bash
arr=(34 23 5 67 89 12)
n=${#arr[@]}
for ((i = 0; i<n; i++))
do
   for ((j = 0; j<n-i-1; j++))
   do
      if [ ${arr[j]} -gt ${arr[$((j+1))]} ]; then
         temp=${arr[j]}
arr[$j]=${arr[$((j+1))]}
arr[$((j+1))]=$temp
      fi
   done
done
echo "Sorted Array: ${arr[@]}"
```

4. Write a Shell script to perform sequential search on a given array elements.

Code:

```
#!/bin/bash
echo "Enter the number to search:"
read key
arr=(10 20 30 40 50)
found=0
for num in "${arr[@]}"
do
   if [ "$num" -eq "$key" ]; then
      found=1
      break
   fi
done
if [ "$found" -eq1 ]; then
   echo "Element found"
else
```

```
    echo "Element not found"
fi
```

5. Write a Shell script to perform binary search on a given array elements.

Code:

```
#!/bin/bash
arr=(10 20 30 40 50 60 70 80 90)
echo "Enter element to search: "
read key
low=0
high=$((${#arr[@]} - 1))
found=0
while [ "$low" -le "$high" ]
do
    mid=$(((low + high) / 2))
    if [ "$key" -eq "${arr[mid]}" ]; then
        found=1
        break
elif [ "$key" -lt "${arr[mid]}" ]; then
        high=$((mid - 1))
    else
        low=$((mid + 1))
    fi
done
if [ "$found" -eq1 ]; then
    echo "Element found"
else
    echo "Element not found"
fi
```

Conclusion: In this experiment, we have successfully learned different commands related to process management.

Experiment No.6

(System Monitoring Linux Commands)

Experiment: Running System Monitoring & Linux command

Hardware & Software Requirement:

Computer, installed Windows O.S., Vmware

Description:

vmstat

Description: Shows memory stats.

Syntax: vmstat

Example: vmstat 2

iostat

Description: Shows I/O stats.

Syntax: iostat

Example: iostat -x

sar

Description: Reports activity.

Syntax: sar

Example: sar -u 1 5

uptime

Description: Shows running time.

Syntax: uptime

Example: uptime

dstat

Description: Shows resource stats.

Syntax: dstat

Example: dstat -cd

iotop

Description: Shows disk usage.

Syntax: iotop

Example: iotop

Logs

journalctl

Description: Views logs.

Syntax: journalctl

Example: journalctl -xe

dmesg

Description: Shows kernel logs.

Syntax: dmesg

Example: dmesg | less

logrotate

Description: Rotates logs.

Syntax: logrotate config

Example: logrotate /etc/logrotate.conf

Conclusion: In this experiment, we have successfully learned different commands related system monitoring & performance tuning.

Experiment 7

(Experimenting with Firewall & Security Related Linux Commands)

Experiment: Running of firewall & security related commands

Hardware & Software Requirement:

Computer, installed Windows O.S., Vmware

Description:

iptables

Description: Configures firewall.

Syntax: iptables -L

Example: iptables -L

ufw

Description: Simple firewall.

Syntax: ufw enable

Example: ufw allow 22

Services

systemctl

Description: Controls services.

Syntax: systemctl start service

Example: systemctl start apache2

service

Description: Controls legacy services.

Syntax: service name start

Example: service ssh start

Conclusion: In this experiment, we have successfully learned different commands related firewall & security

Faculty Senior Lecturer