Linux Essentials, Networking, and Automation

Task 1: Linux Essentials & File Permissions

Objective:

To learn basic Linux user management, directory structure creation, file operations, and file permission settings by creating a user, organizing project directories, and securing files.

1.Create a new user as USER

sudo adduser USER or **sudo useradd USER**

```
File  Actions  Edit  View  Help
└─$ sudo useradd -m USER

┌──(kali㉿kali)-[~]
└─$ sudo passwd USER
New password:
Retype new password:
passwd: password updated successfully
```

2.Create project directory structure:

sudo mkdir -p /home/USER/projectX/{logs,scripts}

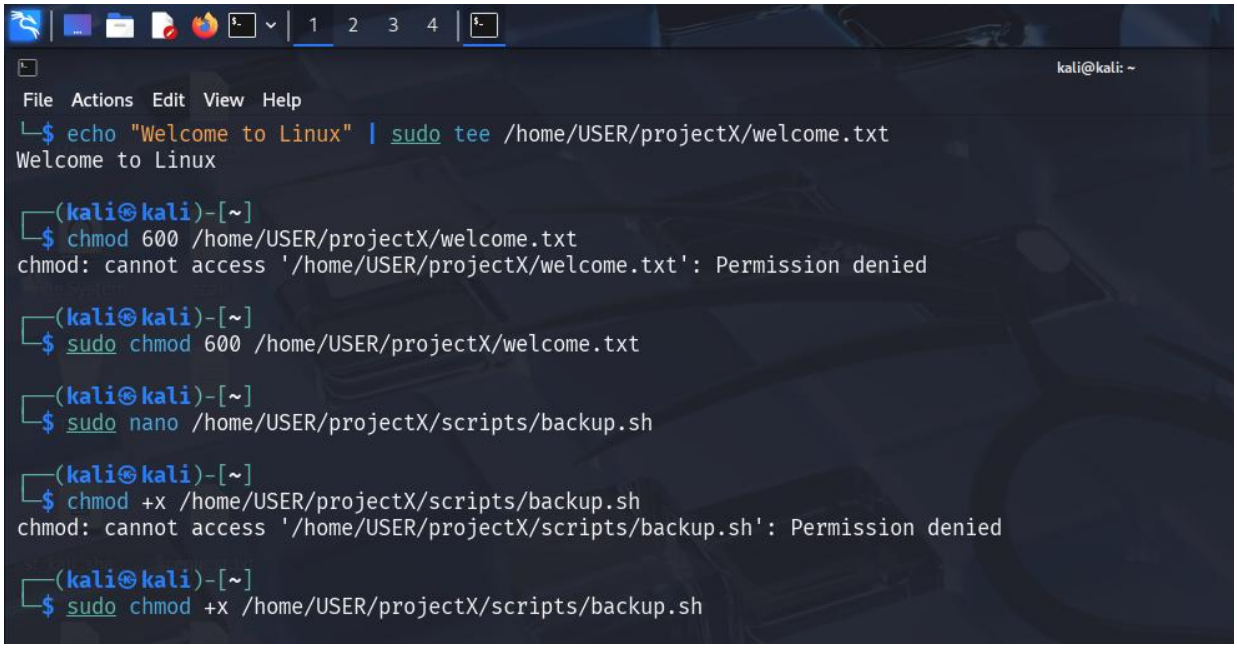
3.Set ownership to user:

sudo chown -R USER:USER /home/user

```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir -p /home/USER/projectX/{logs,scripts}

┌──(kali㉿kali)-[~]
└─$ sudo chown -R USER:USER /home/USER
```

4.Create welcome.txt with content:

echo "Welcome to Linux" | sudo tee /home/USER/projectX/welcome.txt



5.Set file permissions (only user can read/write):

sudo chmod 600 /home/USER/projectX/welcome.txt

file permissions like this:

- **6 (Owner):** Read + Write (4 + 2 = 6)
- **0 (Group):** No permissions
- **0 (Others):** No permissions

6.Create backup.sh script in scripts folder:

nano /home/USER/projectX/scripts/backup.sh

Content of backup.sh:

#!/bin/bash

cp /home/USER/projectX/welcome.txt

/home/USER/projectX/logs/welcome_$(date +%Y%m%d_%H%M%S).txt

```
┌──(kali㉿kali)-[~]
└─$ sudo /home/USER/projectX/scripts/backup.sh

┌──(kali㉿kali)-[~]
└─$ ls /home/USER/projectX/logs/

ls: cannot access '/home/USER/projectX/logs/': Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo ls /home/USER/projectX/logs/

welcome_2025-07-23_23-09-34.txt

┌──(kali㉿kali)-[~]
└─$
```

Make executable and run:

chmod +x /home/USER/projectX/scripts/backup.sh

./home/USER/projectX/scripts/backup.sh

Execte as a root user if the permission has appeared as permission denied.

7.To view type sudo ls /home/USER/projectX/logs/

Conclusion:

We created the USER account, built the projectX directory structure, added a welcome.txt file with custom content, and set permissions so only USER can access it. A backup.sh script was also created to back up the file with a timestamp.

Task 2: Networking Toolkit

Objective:

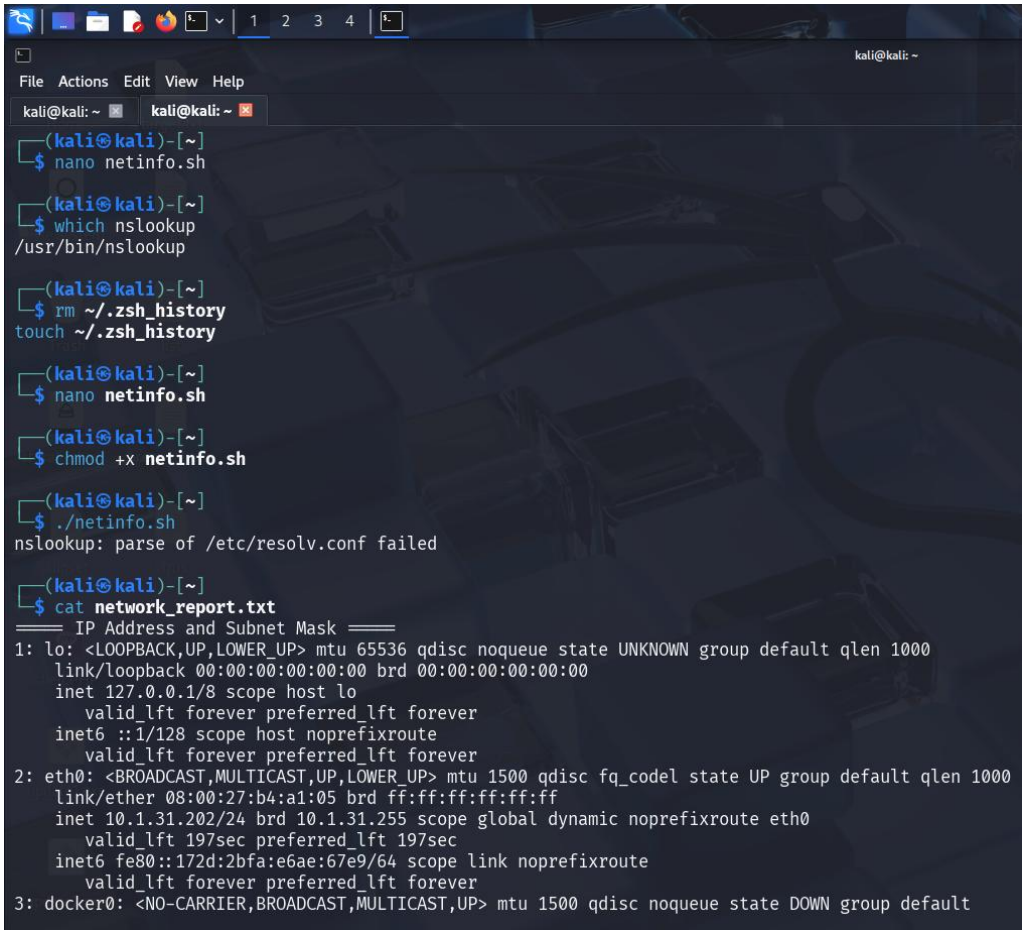
To create a shell script netinfo.sh that gathers and stores key network information including IP details, open ports, connectivity check, and DNS resolution.

Steps Performed:

1. Create the script file:

nano /home/studentuser/projectX/scripts/netinfo.sh

2. Add the following script content:



#!/bin/bash

Network Information Script

```
echo "----- Network Report -----" >
/home/studentuser/projectX/network_report.txt

echo "Date & Time: $(date)" >>
/home/studentuser/projectX/network_report.txt


# Display IP address, subnet mask, and default gateway

echo -e "\nIP Address, Subnet Mask & Gateway:" >>
/home/studentuser/projectX/network_report.txt

ip addr show >> /home/studentuser/projectX/network_report.txt

ip route | grep default >> /home/studentuser/projectX/network_report.txt


# List open ports

echo -e "\nOpen Ports:" >> /home/studentuser/projectX/network_report.txt

ss -tuln >> /home/studentuser/projectX/network_report.txt


# Ping google.com and log response

echo -e "\nPing google.com:" >>
/home/studentuser/projectX/network_report.txt

ping -c 4 google.com >> /home/studentuser/projectX/network_report.txt


# DNS lookup for openai.com

echo -e "\nDNS Lookup for openai.com:" >>
/home/studentuser/projectX/network_report.txt

nslookup openai.com >> /home/studentuser/projectX/network_report.txt
```

3. **Save and exit the file (Ctrl + O, Enter, Ctrl + X).**
4. Make the script executable:

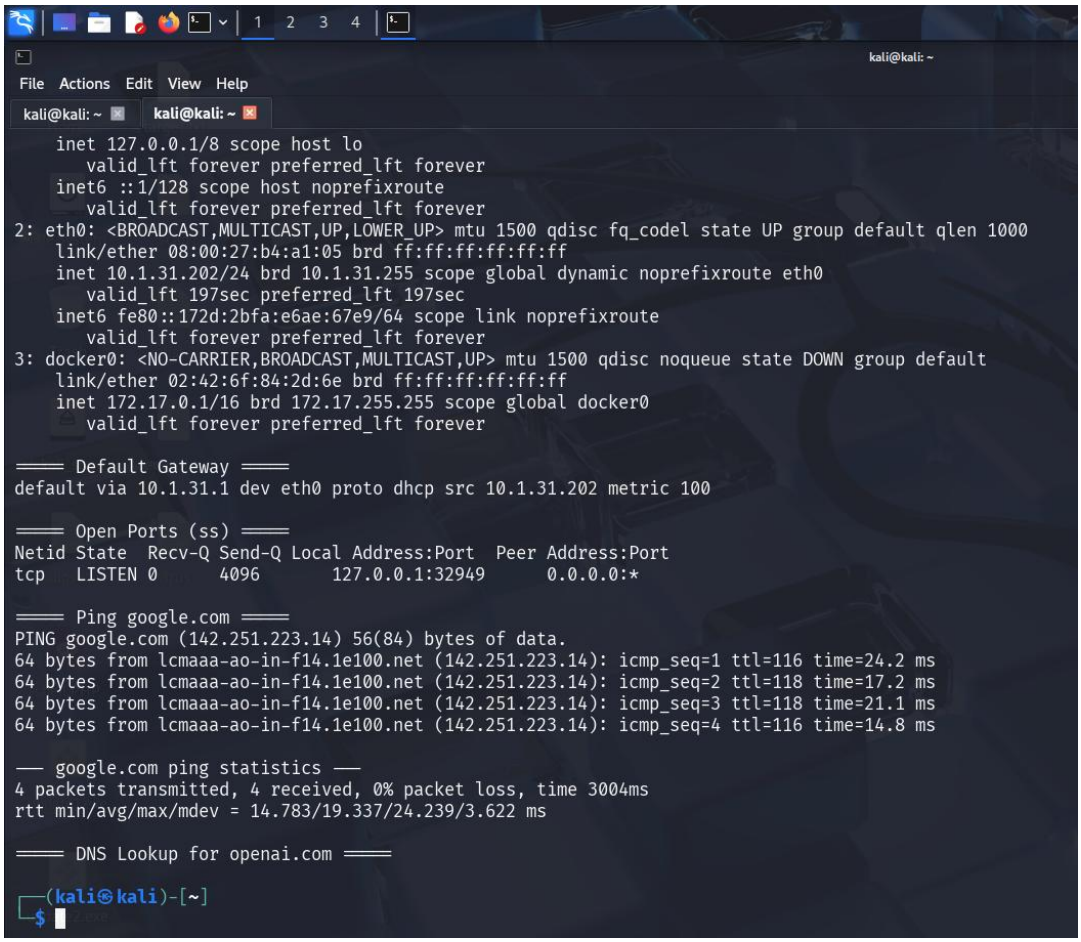
chmod +x /home/studentuser/projectX/scripts/netinfo.sh

5. Run the script:

./home/studentuser/projectX/scripts/netinfo.sh

6. Check the output report:

cat /home/studentuser/projectX/network_report.txt



Conclusion

We created netinfo.sh to gather IP details, open ports, connectivity (ping), and DNS resolution. The output is stored in network_report.txt, providing a quick network status report for troubleshooting.

Task 3: Mini Server Monitor Script

Objective:

To create a monitoring script monitor.sh that checks the nginx service status, displays system resource usage (CPU, memory, disk), and logs the results with timestamps for server health analysis.

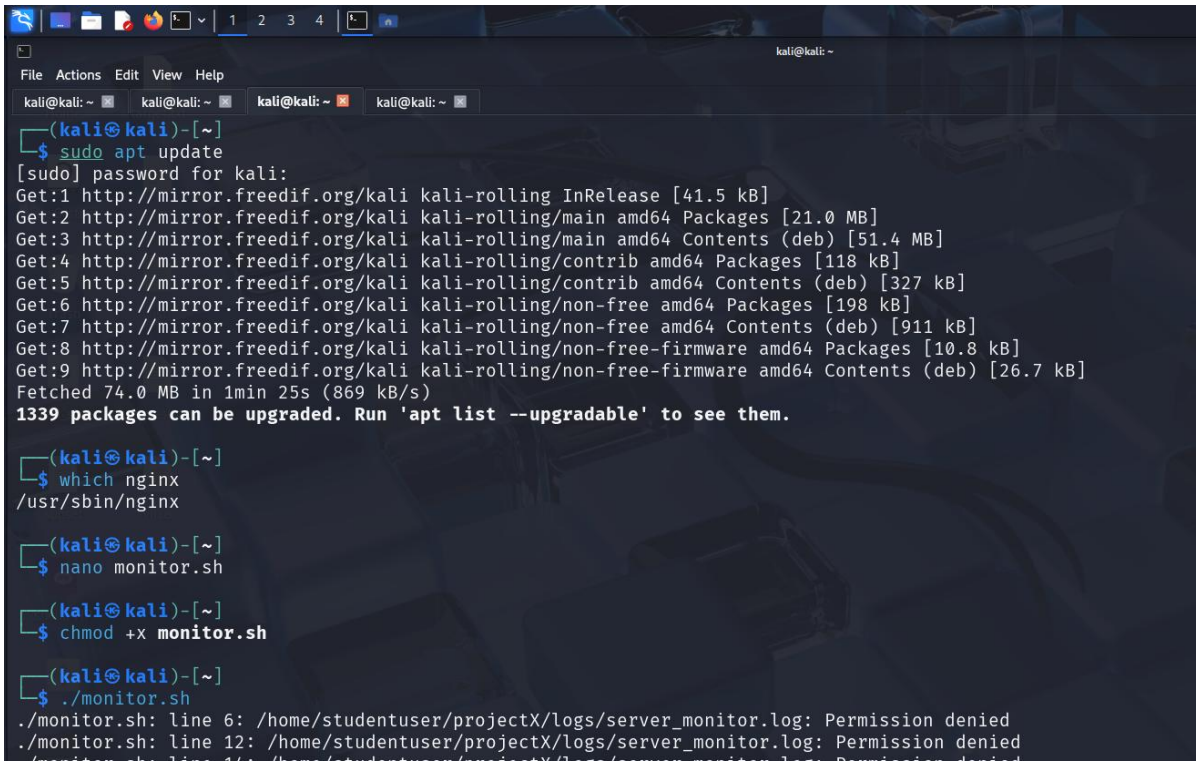
Steps Performed:

1. Navigate to the scripts directory:

cd /home/studentuser/projectX/scripts

2. Create and edit the monitor script:

nano monitor.sh



3. Add the following script content:

#!/bin/bash

LOGFILE="/home/studentuser/projectX/logs/monitor.log"

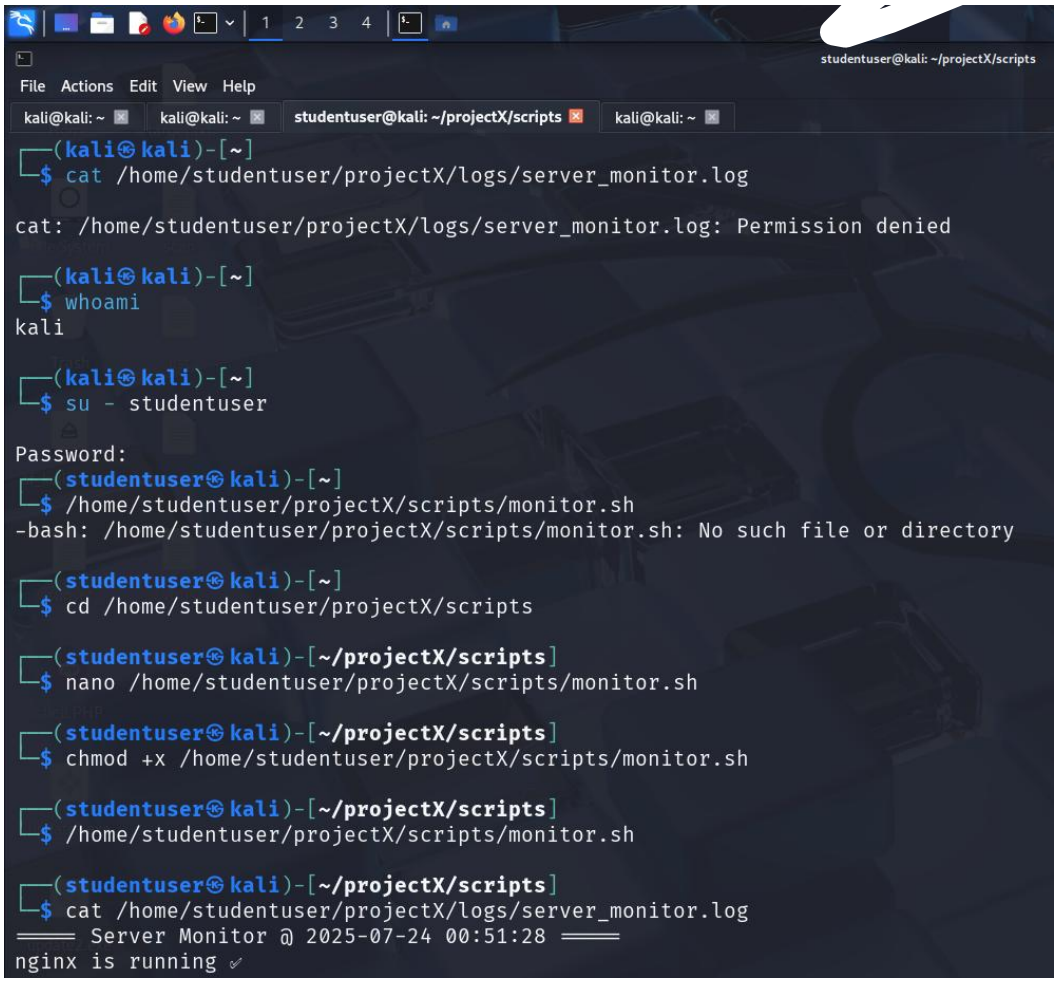
```
echo "----- Server Monitor Log -----" >> $LOGFILE
echo "Date: $(date)" >> $LOGFILE

# Check if nginx is running
if systemctl is-active --quiet nginx; then
    echo "nginx is running." >> $LOGFILE
else
    echo "nginx is not running. Starting nginx..." >> $LOGFILE
    sudo systemctl start nginx
fi

# Display memory usage
echo -e "\nMemory Usage:" >> $LOGFILE
free -h >> $LOGFILE

# Display CPU load
echo -e "\nCPU Load:" >> $LOGFILE
uptime >> $LOGFILE

# Display disk usage
echo -e "\nDisk Usage:" >> $LOGFILE
df -h >> $LOGFILE
```

4. **Save and exit** (Ctrl + O, Enter, Ctrl + X).
5. Give executable permission:

chmod +x monitor.sh

6. Run the script manually to test:

./monitor.sh



7. Check the log file:

cat /home/studentuser/projectX/logs/monitor.log

```
└─$ cd /home/studentuser/projectX/scripts

┌──(studentuser㉿kali)-[~/projectX/scripts]
└─$ nano /home/studentuser/projectX/scripts/monitor.sh

┌──(studentuser㉿kali)-[~/projectX/scripts]
└─$ chmod +x /home/studentuser/projectX/scripts/monitor.sh

┌──(studentuser㉿kali)-[~/projectX/scripts]
└─$ /home/studentuser/projectX/scripts/monitor.sh

┌──(studentuser㉿kali)-[~/projectX/scripts]
└─$ cat /home/studentuser/projectX/logs/server_monitor.log
═══ Server Monitor @ 2025-07-24 00:51:28 ═══
nginx is running ✓

── Memory Usage ──
              total        used        free      shared  buff/cache   available
Mem:           1.9Gi       876Mi       363Mi        26Mi       905Mi       1.1Gi
Swap:          1.0Gi          0B       1.0Gi

── CPU Load ──
 00:51:28 up  1:59,  2 users,  load average: 0.07, 0.07, 0.08

── Disk Usage ──
Filesystem      Size  Used Avail Use% Mounted on
udev            922M     0  922M   0% /dev
tmpfs           198M  984K  197M   1% /run
/dev/sda1        79G   18G   57G  25% /
tmpfs           987M  4.0K  987M   1% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           987M  256K  987M   1% /tmp
kali_shared     150G  106G   45G  71% /media/sf_kali_shared
```

Conclusion:

The monitor.sh script manually checks and logs the status of nginx along with memory, CPU, and disk usage. Running this script as needed helps in basic system monitoring.

Task 4: File Watcher Script

Objective:

To create a script watch_dir.sh that continuously monitors the /home/studentuser/projectX/logs directory for any new .txt files and logs their names along with timestamps into log_monitor.txt.

Steps Performed:

1. Navigate to the scripts directory:

cd /home/studentuser/projectX/scripts

2. Create the watch_dir.sh script:

nano watch_dir.sh

```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir -p /home/studentuser/projectX/logs

[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ nano /home/studentuser/projectX/scripts/watch_dir.sh


┌──(kali㉿kali)-[~]
└─$ nano /home/studentuser/projectX/scripts/watch_dir.sh


┌──(kali㉿kali)-[~]
└─$ sudo nano /home/studentuser/projectX/scripts/watch_dir.sh


┌──(kali㉿kali)-[~]
└─$ chmod +x /home/studentuser/projectX/scripts/watch_dir.sh

chmod: cannot access '/home/studentuser/projectX/scripts/watch_dir.sh': Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo chmod +x /home/studentuser/projectX/scripts/watch_dir.sh
```

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1339 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(kali㉿kali)-[~]
└─$ inotifywait --version

Command 'inotifywait' not found, but can be installed with:
sudo apt install inotify-tools
Do you want to install it? (N/y)y
sudo apt install inotify-tools
Installing:
  inotify-tools

Installing dependencies:
  libinotifytools0

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 1339
  Download size: 56.7 kB
  Space needed: 233 kB / 60.5 GB available

Continue? [Y/n] y
Get:1 http://http.kali.org/kali kali-rolling/main amd64 libinotifytools0 amd64 4.23.9.0-2+b1 [23.1 kB
Get:2 http://http.kali.org/kali kali-rolling/main amd64 inotify-tools amd64 4.23.9.0-2+b1 [33.6 kB]
Fetched 56.7 kB in 1s (39.2 kB/s)
Selecting previously unselected package libinotifytools0:amd64.
```

```
1 2 3 4                                                kali@kali: ~
File Actions Edit View Help
kali@kali: ~   kali@kali: ~
Selecting previously unselected package inotify-tools.
Preparing to unpack .../inotify-tools_4.23.9.0-2+b1_amd64.deb ...
Unpacking inotify-tools (4.23.9.0-2+b1) ...
Setting up libinotifytools0:amd64 (4.23.9.0-2+b1) ...
Setting up inotify-tools (4.23.9.0-2+b1) ...
Processing triggers for libc-bin (2.41-6) ...
Processing triggers for man-db (2.13.0-1) ...
Processing triggers for kali-menu (2025.1.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

┌──(kali㉿kali)-[~]
└─$ /home/studentuser/projectX/scripts/watch_dir.sh

zsh: permission denied: /home/studentuser/projectX/scripts/watch_dir.sh

┌──(kali㉿kali)-[~]
└─$ sudo /home/studentuser/projectX/scripts/watch_dir.sh

Setting up watches.
Watches established.
```

3. Add the following content to the script:

```
#!/bin/bash
WATCH_DIR="/home/studentuser/projectX/logs"
LOG_FILE="/home/studentuser/projectX/logs/log_monitor.txt"

# Monitor for new .txt files
inotifywait -m -e create --format '%f' "$WATCH_DIR" | while read NEWFILE
do
   if [[ $NEWFILE == *.txt ]]; then
      echo "$(date): New file detected - $NEWFILE" >> $LOG_FILE
   fi
done
```

4. **Save and exit** (Ctrl + O, Enter, Ctrl + X).
5. Give executable permissions:

chmod +x watch_dir.sh

6. Run the script:

./watch_dir.sh

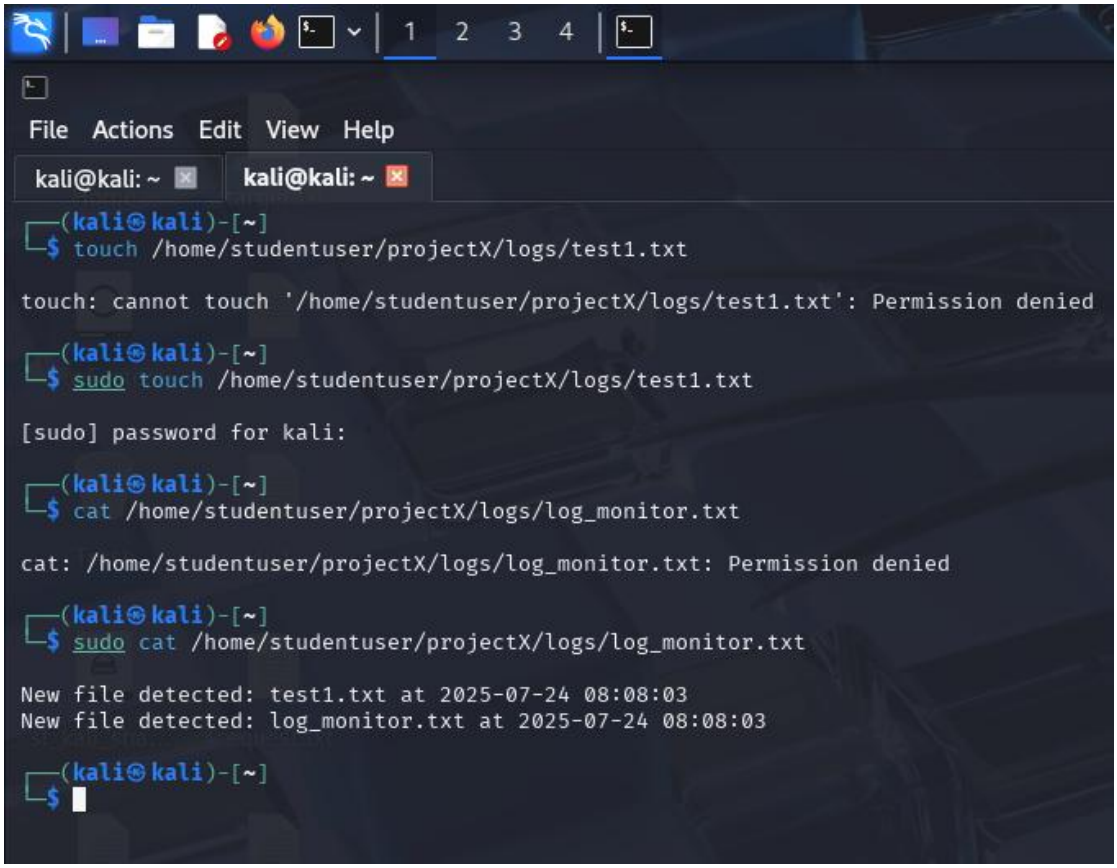
7. Test the watcher:

Open another terminal and create a .txt file in the logs folder:

echo "Test log file" > /home/studentuser/projectX/logs/test1.txt

8. **Check log_monitor.txt for entry:**

cat /home/studentuser/projectX/logs/log_monitor.txt



Conclusion:

we successfully created and tested a **directory watch script** using the inotifywait tool. The script monitors a specified directory for file changes (create, modify, delete) and logs these events with timestamps into a log file located at /home/studentuser/projectX/logs/watch_log.txt

Task 5 – SSH Login Audit

Objective

The objective of this task is to analyze SSH login activity on the system by identifying the last 5 successful and last 5 failed login attempts. This helps in monitoring unauthorized access and auditing user activity.

Steps Followed:

1. **Navigate to projectX logs directory (optional):**

cd /home/studentuser/projectX/logs

2. **Create the ssh_audit.sh script:**

nano /home/studentuser/projectX/scripts/ssh_audit.sh

3. **Add the following script content:**

```
#!/bin/bash

echo "Last 5 Successful Logins:" >
/home/studentuser/projectX/logs/ssh_audit.txt

last -n 5 >> /home/studentuser/projectX/logs/ssh_audit.txt


echo -e "\nLast 5 Failed Login Attempts:" >>
/home/studentuser/projectX/logs/ssh_audit.txt

sudo journalctl _COMM=sshd | grep "Failed password" | tail -n 5 >>
/home/studentuser/projectX/logs/ssh_audit.txt
```

4. Make the script executable:

chmod +x /home/studentuser/projectX/scripts/ssh_audit.sh

```
┌──(kali㉿kali)-[~]
└─$ sudo nano /home/studentuser/projectX/scripts/ssh_audit.sh
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo chmod +x /home/studentuser/projectX/scripts/ssh_audit.sh

┌──(kali㉿kali)-[~]
└─$ sudo /home/studentuser/projectX/scripts/ssh_audit.sh
grep: /var/log/auth.log: No such file or directory
grep: /var/log/auth.log: No such file or directory
SSH audit report saved to /home/studentuser/projectX/logs/ssh_audit.txt

┌──(kali㉿kali)-[~]
└─$ /home/studentuser/projectX/scripts/ssh_audit.sh
zsh: permission denied: /home/studentuser/projectX/scripts/ssh_audit.sh

┌──(kali㉿kali)-[~]
└─$ cat /home/studentuser/projectX/logs/ssh_audit.txt
cat: /home/studentuser/projectX/logs/ssh_audit.txt: Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/ssh_audit.txt
─── Last 5 Successful Logins ───

─── Last 5 Failed Login Attempts ───

┌──(kali㉿kali)-[~]
└─$ sudo ls /home/USER/projectX/logs/
```

5. Run the script:

./home/studentuser/projectX/scripts/ssh_audit.sh

6. Check the output file:

cat /home/studentuser/projectX/logs/ssh_audit.txt

If there are no failed or successful entries (e.g., on a fresh system), the file may only show the headers.

```
File  Actions  Edit  View  Help
kali@kali: ~   kali@kali: ~   kali@kali: ~   kali@kali: ~

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/ssh_audit.txt
─── Last 5 Successful Logins ───

─── Last 5 Failed Login Attempts ───

┌──(kali㉿kali)-[~]
└─$ sudo journalctl -u ssh | tail
-- No entries --

┌──(kali㉿kali)-[~]
└─$ sudo nano /home/studentuser/projectX/scripts/ssh_audit.sh


┌──(kali㉿kali)-[~]
└─$ sudo nano /home/studentuser/projectX/scripts/ssh_audit.sh


┌──(kali㉿kali)-[~]
└─$ sudo chmod +x /home/studentuser/projectX/scripts/ssh_audit.sh

┌──(kali㉿kali)-[~]
└─$ sudo /home/studentuser/projectX/scripts/ssh_audit.sh
SSH audit report saved to /home/studentuser/projectX/logs/ssh_audit.txt

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/ssh_audit.txt
─── Last 5 Successful Logins ───
kali     tty7         :0               Thu Jul 24 07:53 - still logged in
lightdm  tty7         :0               Thu Jul 24 07:53 - 07:53  (00:00)
studentu pts/2                         Thu Jul 24 00:49 - 02:01  (01:12)
kali     tty7         :0               Wed Jul 23 22:52 - still logged in
lightdm  tty7         :0               Wed Jul 23 22:51 - 22:52  (00:00)

─── Last 5 Failed Login Attempts ───
Failed password for invalid user hacker from 192.168.1.6 port 22 ssh2

┌──(kali㉿kali)-[~]
└─$
```

Conclusion:

By performing this task, we successfully extracted information about the last 5 successful and failed SSH login attempts from system logs. This helps in auditing login activity and improving system security.

Task 6: Crontab Practice

Objective:

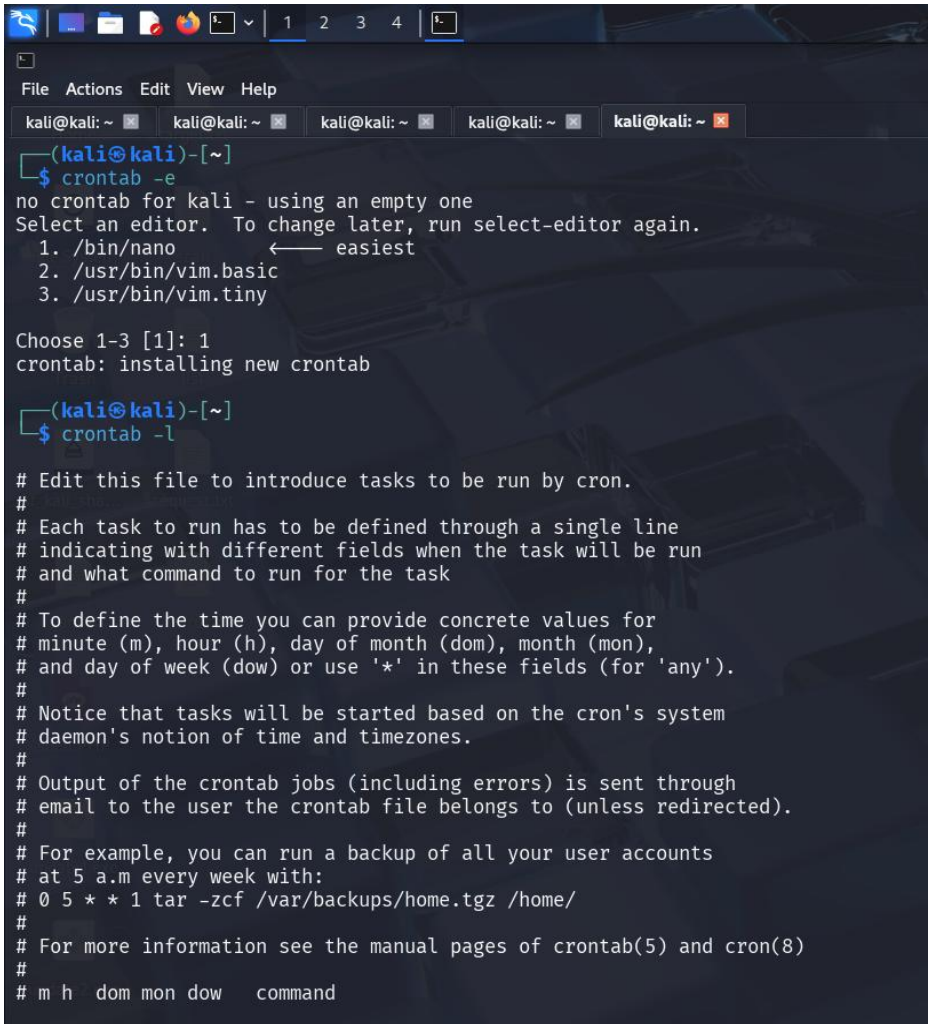
The objective of this task is to schedule automated jobs using cron. This helps in automating repetitive tasks such as daily messages, backups, and cleanup operations.

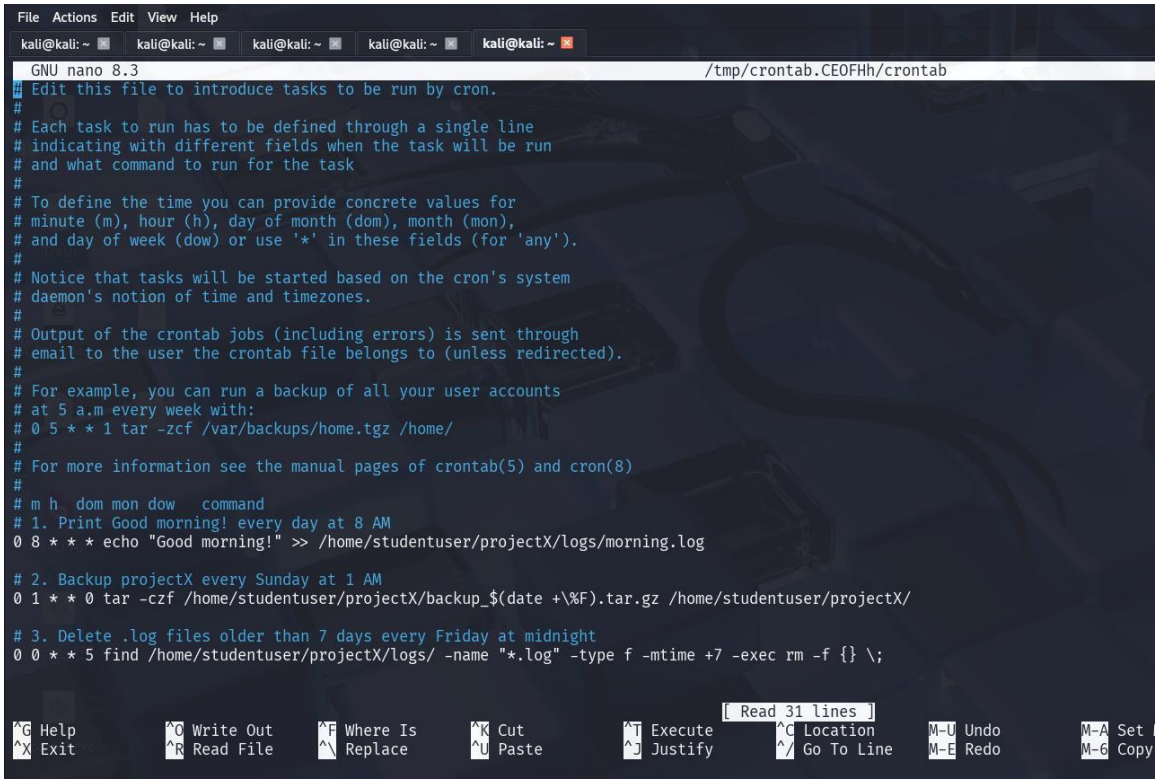
Steps Performed:

1. Open the crontab editor for the current user:

crontab -e

This opens the cron table in the default editor.





3. Save and Exit Nano

- Press **CTRL + O** (Write Out), then hit **Enter**.
- Press **CTRL + X** to exit.

4. Verify the Cron Jobs

- Run:
- crontab -l
- This will list all scheduled tasks you added.

```
┌──(kali㉿kali)-[~]
└─$ crontab -l

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
# 1. Print Good morning! every day at 8 AM
0 8 * * * echo "Good morning!" >> /home/studentuser/projectX/logs/morning.log

# 2. Backup projectX every Sunday at 1 AM
0 1 * * 0 tar -czf /home/studentuser/projectX/backup_$(date +\%F).tar.gz /home/studentuser/projectX/

# 3. Delete .log files older than 7 days every Friday at midnight
0 0 * * 5 find /home/studentuser/projectX/logs/ -name "*.log" -type f -mtime +7 -exec rm -f {} \;

┌──(kali㉿kali)-[~]
```

5. Check if Automation Works

Wait until 8 AM or manually run the command:

echo "Good morning" | sudo tee -a /home/studentuser/projectX/logs/morning.log

- tee -a appends the text with sudo privileges.

cat /home/studentuser/projectX/logs/morning.log

- echo appends text.
- cat displays the contents of the log.

This simulates what the cron job would do.

```
┌──(kali㉿kali)-[~]
└─$ echo "Good morning" | sudo tee -a /home/studentuser/projectX/logs/morning.log

[sudo] password for kali:
Good morning

┌──(kali㉿kali)-[~]
└─$ echo "Good morning" >> /home/studentuser/projectX/logs/morning.log

zsh: permission denied: /home/studentuser/projectX/logs/morning.log

┌──(kali㉿kali)-[~]
└─$ cat /home/studentuser/projectX/logs/morning.log
cat: /home/studentuser/projectX/logs/morning.log: Permission denied

┌──(kali㉿kali)-[~]
└─$ echo "Good morning" | sudo tee -a /home/studentuser/projectX/logs/morning.log

Good morning

┌──(kali㉿kali)-[~]
└─$ cat /home/studentuser/projectX/logs/morning.log
cat: /home/studentuser/projectX/logs/morning.log: Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/morning.log
Good morning
Good morning
```

Conclusion:

With nano and crontab -e, we created scheduled tasks for daily greetings, weekly backups, and automatic log cleanup. This ensures automation without manual intervention.

Task 7 – Port Scanner Script

Objective:

Write a simple shell script to scan ports 20–25 on a user-supplied IP using nc (netcat) or timeout**.**

Steps:

1. Open Nano to Create the Script

nano port_scanner.sh

2. Add the Following Script Content

Paste this code inside nano:

```
#!/bin/bash

# Port Scanner Script for ports 20-25


# Ask the user for the IP address

read -p "Enter IP address to scan: " ip


echo "Scanning ports 20 to 25 on $ip..."


# Loop through ports 20 to 25

for port in {20..25}

do

   timeout 1 bash -c "echo > /dev/tcp/$ip/$port" 2>/dev/null && \

   echo "Port $port is OPEN" || \

   echo "Port $port is CLOSED"

done
```

3. Save and Exit Nano

- Press Ctrl + O, then Enter to save.
- Press Ctrl + X to exit.

4. Make the Script Executable

chmod +x port_scanner.sh

5. Run the Script

./port_scanner.sh

- Enter the target IP when prompted.
- It will check ports 20, 21, 22, 23, 24, and 25.

```
┌──(kali㉿kali)-[~]
└─$ sudo cd /home/studentuser/projectX/scripts
[sudo] password for kali:
sudo: cd: command not found
sudo: "cd" is a shell built-in command, it cannot be run directly.
sudo: the -s option may be used to run a privileged shell.
sudo: the -D option may be used to run a command in a specific directory.

┌──(kali㉿kali)-[~]
└─$ nano portscanner.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x portscanner.sh

┌──(kali㉿kali)-[~]
└─$ ./portscanner.sh 127.0.0.1
Scanning ports 20-25 on 127.0.0.1 ...
Port 20 is CLOSED
Port 21 is CLOSED
Port 22 is CLOSED
Port 23 is CLOSED
Port 24 is CLOSED
Port 25 is CLOSED

┌──(kali㉿kali)-[~]
└─$
```

Conclusion:

This script scans ports 20–25 on any given IP using a for loop and timeout to detect open or closed ports.

Task 8 – Website Availability Checker

Objective:

Create a script to check the availability of websites listed in sites.txt using curl or ping, and log the results into site_status.log.

Steps:

1. Create a File with Website URLs

nano sites.txt

Add some websites, one per line:

https://google.com

https://openai.com

https://github.com

Save with Ctrl + O → Enter, then exit with Ctrl + X.

```
┌──(kali㉿kali)-[~]
└─$ sudo cd /home/studentuser/projectX/scripts
[sudo] password for kali:
sudo: cd: command not found
sudo: "cd" is a shell built-in command, it cannot be run directly.
sudo: the -s option may be used to run a privileged shell.
sudo: the -D option may be used to run a command in a specific directory.

┌──(kali㉿kali)-[~]
└─$ nano sites.txt

┌──(kali㉿kali)-[~]
└─$ nano site_checker.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x site_checker.sh
```

2. Create the Script

nano site_checker.sh

Paste the following script:

#!/bin/bash

Website Availability Checker

```
input="sites.txt"
output="site_status.log"

# Clear previous log
> $output

while IFS= read -r site
do
    echo "Checking $site..."
    if curl -Is --max-time 5 "$site" | grep "200 OK" > /dev/null
    then
        echo "$site is UP" | tee -a $output
    else
        echo "$site is DOWN" | tee -a $output
    fi
done < "$input"
```

3. Save and Exit
   - Ctrl + O → Enter
   - Ctrl + X

```
GNU nano 8.3                                                    site_che
#!/bin/bash
# Website Availability Checker

INPUT_FILE="sites.txt"
LOG_FILE="/home/studentuser/projectX/logs/site_status.log"

echo "Website check started at $(date)" >> $LOG_FILE

while read -r site; do
    if curl -Is --connect-timeout 5 "$site" >/dev/null 2>&1; then
        echo "$site is UP" >> $LOG_FILE
    else
        echo "$site is DOWN" >> $LOG_FILE
    fi
done < $INPUT_FILE

echo "Check completed at $(date)" >> $LOG_FILE
echo "————————————————————————————————————" >> $LOG_FILE

^G Help      ^O Write Out   ^F Where Is   ^K Cut      ^T Exe
^X Exit      ^R Read File   ^\ Replace    ^U Paste    ^J Jus
```

4. Make the Script Executable

chmod +x site_checker.sh

5. Run the Script

./site_checker.sh

It will check each site and log results into site_status.log.

6. View the Log

cat /home/studentuser/project/logs/site_status.log

```
┌──(kali㉿kali)-[~]
└─$ sudo ./site_checker.sh

┌──(kali㉿kali)-[~]
└─$ cat /home/studentuser/projectX/logs/site_status.log

cat: /home/studentuser/projectX/logs/site_status.log: Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/site_status.log

Website check started at Fri Jul 25 09:41:47 AM EDT 2025
https://google.com is UP
https://openai.com is UP
https://github.com is UP
Check completed at Fri Jul 25 09:41:48 AM EDT 2025
────────────────────────────────────────

┌──(kali㉿kali)-[~]
└─$
```

Conclusion:

This script automates website monitoring by checking each URL's HTTP status and logging availability results.

Task 9 – Environment and Disk Report

Objective:

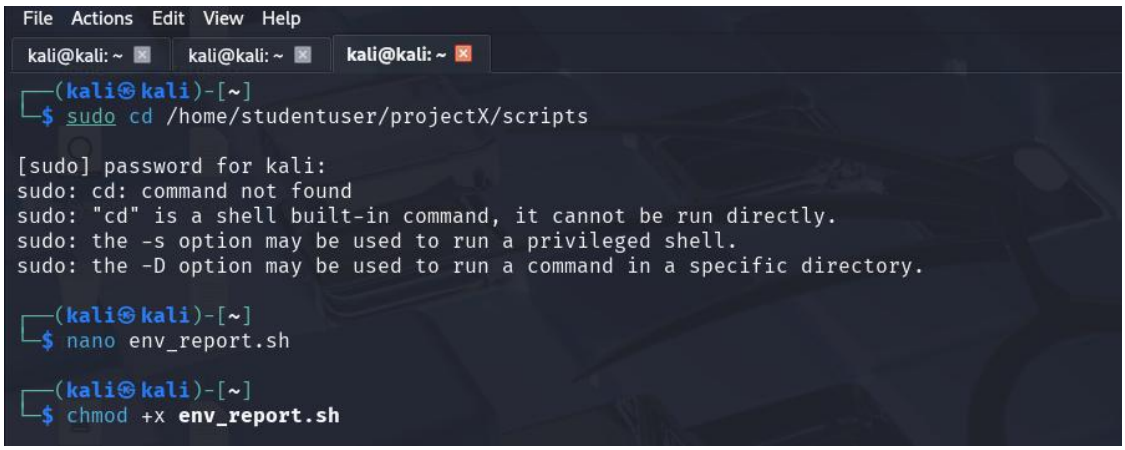
Create a script that generates a system report including:

- Current user, hostname, uptime.
- Mounted filesystems and their usage.
- Path and shell-related environment variables.

Steps:

1. Create the Script

nano env_disk_report.sh



Paste the following script:

```
#!/bin/bash
# Environment and Disk Report Script
output="environment_disk_report.txt"
echo "===== Environment and Disk Report =====" > $output
echo "" >> $output
```

```
# Current user
echo "Current User: $(whoami)" >> $output
# Hostname
echo "Hostname: $(hostname)" >> $output
# Uptime
echo "System Uptime:" >> $output
uptime >> $output
echo "" >> $output
# Mounted Filesystems and Usage
echo "Mounted Filesystems and Usage:" >> $output
df -h >> $output
echo "" >> $output
# Path and Shell Variables
echo "PATH Variable: $PATH" >> $output
echo "Current Shell: $SHELL" >> $output
echo "" >> $output
echo "Report generated on: $(date)" >> $output
```

2. Save and Exit

- Ctrl + O → Enter
- Ctrl + X

3. Make the Script Executable

```
chmod +x env_disk_report.sh
```

4. Run the Script

```
./env_disk_report.sh
```

5. View the Report

cat environment_disk_report.txt

```
┌──(kali㉿kali)-[~]
└─$ sudo ./env_report.sh

┌──(kali㉿kali)-[~]
└─$ cat /home/studentuser/projectX/logs/env_disk_report.txt
cat: /home/studentuser/projectX/logs/env_disk_report.txt: Permission denied

┌──(kali㉿kali)-[~]
└─$ sudo cat /home/studentuser/projectX/logs/env_disk_report.txt
═══ Environment and Disk Report ═══
Generated on: Fri Jul 25 11:05:19 AM EDT 2025

Current User: root
Hostname: kali
System Uptime: up 2 hours, 25 minutes

Mounted Filesystems and Usage:
Filesystem      Size  Used Avail Use% Mounted on
udev            922M     0  922M   0% /dev
tmpfs           198M 1000K  197M   1% /run
/dev/sda1        79G   18G   57G  25% /
tmpfs           987M  4.0K  987M   1% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           987M   36K  987M   1% /tmp
kali_shared     150G  106G   45G  71% /media/sf_kali_shared
tmpfs           1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           198M  120K  198M   1% /run/user/1000
tmpfs           1.0M     0  1.0M   0% /run/credentials/systemd-journald.service

PATH Variable: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
Shell: /usr/bin/zsh

═══ End of Report ═══

┌──(kali㉿kali)-[~]
└─$
```

Conclusion:

This script provides a quick overview of the system environment, disk usage, and user-related details, which is useful for system monitoring and audits.

Task 10 – Compress & Archive Automation

Objective:

Automate the process of finding all .log files larger than 10MB in /home/studentuser/projectX/logs, compressing them into a single archive, and moving the archive to /home/studentuser/projectX/backup/.

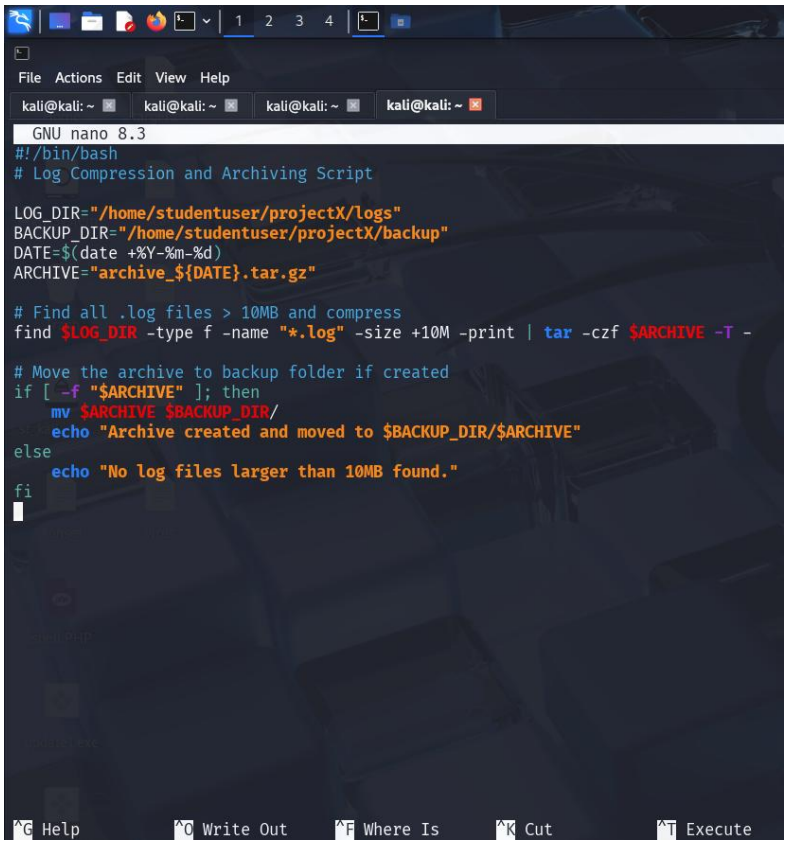
Steps:

1. Create the Backup Directory

sudo mkdir -p /home/studentuser/projectX/backup

2. Create the Script

nano archive_logs.sh



Paste the following script:

```
#!/bin/bash
# Compress & Archive Automation Script

# Variables
LOG_DIR="/home/studentuser/projectX/logs"
BACKUP_DIR="/home/studentuser/projectX/backup"
DATE=$(date +%F)
ARCHIVE_NAME="archive_${DATE}.tar.gz"

# Find .log files > 10MB and compress them
find "$LOG_DIR" -type f -name "*.log" -size +10M -print | tar -czf "$ARCHIVE_NAME" -T -
# Move the archive to the backup directory
mv "$ARCHIVE_NAME" "$BACKUP_DIR/"
echo "Logs larger than 10MB have been archived to $BACKUP_DIR/$ARCHIVE_NAME"
```

3. Save and Exit

- Ctrl + O → Enter
- Ctrl + X

4. Make the Script Executable

```
chmod +x archive_logs.sh
```

5. Run the Script

```
./archive_logs.sh
```

```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir -p /home/studentuser/projectX/backup
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ nano archive_logs.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x archive_logs.sh

┌──(kali㉿kali)-[~]
└─$ ./archive_logs.sh
find: '/home/studentuser/projectX/logs': Permission denied
mv: cannot stat '/home/studentuser/projectX/backup/': Permission denied
Archive created and moved to /home/studentuser/projectX/backup/archive_2025-07-26.tar.gz

┌──(kali㉿kali)-[~]
└─$ sudo ./archive_logs.sh
Archive created and moved to /home/studentuser/projectX/backup/archive_2025-07-26.tar.gz
```

6. Verify the Archive

ls -lh /home/studentuser/projectX/backup/

```
┌──(kali㉿kali)-[~]
└─$ sudo ls -lh /home/studentuser/projectX/backup/

total 8.0K
-rw-r--r-- 1 root root 45 Jul 25 11:14 archive_2025-07-25.tar.gz
-rw-rw-r-- 1 kali kali 45 Jul 26 06:23 archive_2025-07-26.tar.gz

┌──(kali㉿kali)-[~]
└─$
```

Conclusion:

This script automatically finds large .log files, compresses them into a single .tar.gz archive, and securely moves them to the backup directory. It helps in saving disk space and managing old log files efficiently.