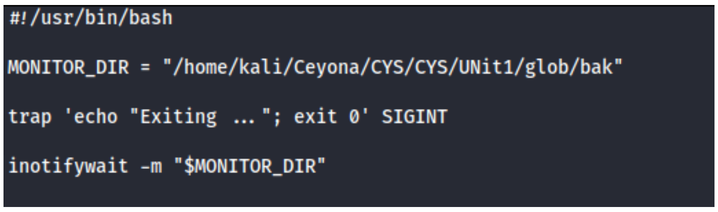
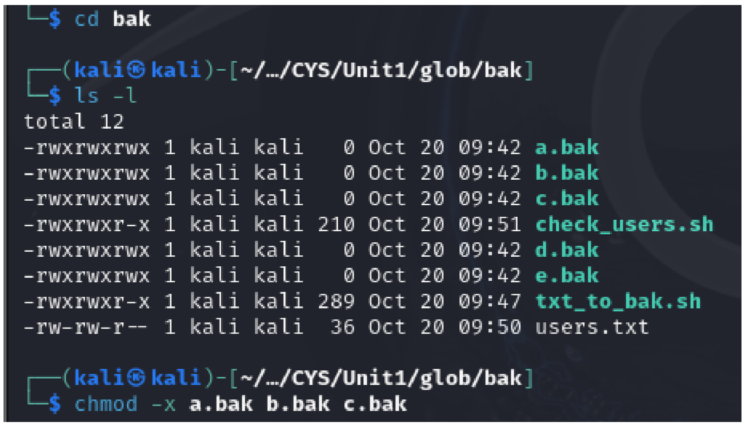
Concepts: Process, trap, kill,

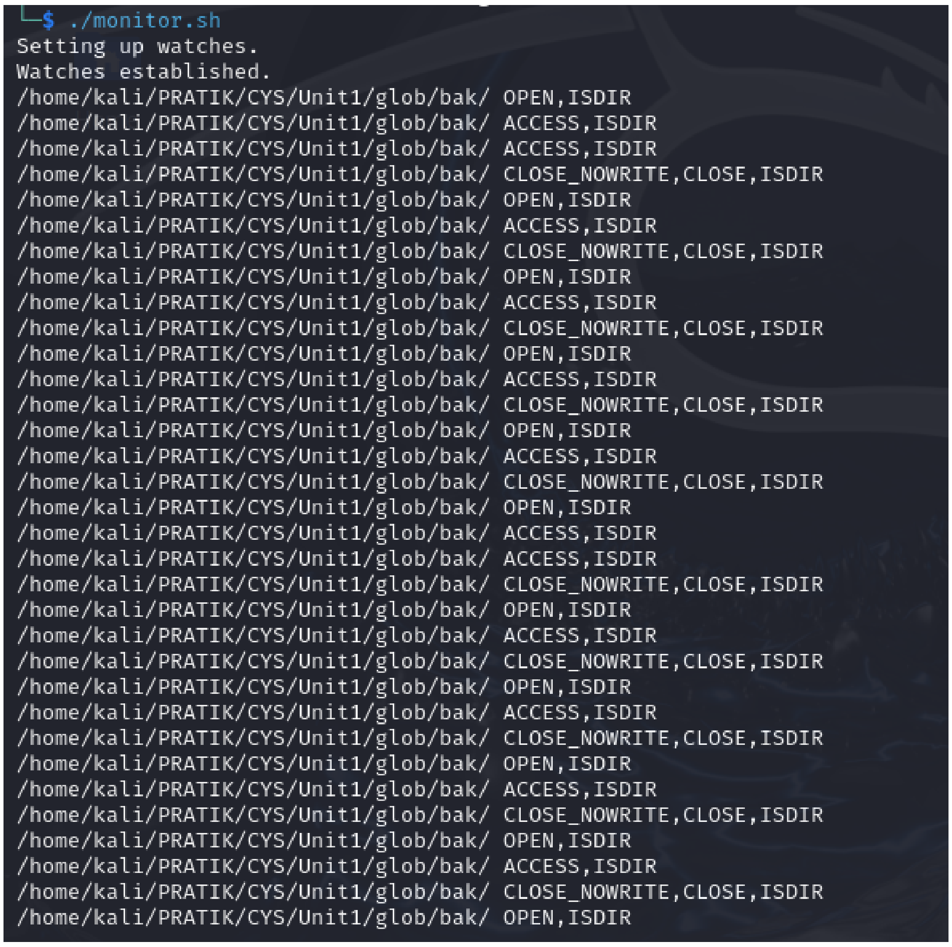
1. Write a bash script that monitors a sensitive directory (of your choice) for changes using inotifywait (Linux command). Use trap to handle SIGINT (Ctrl+C) to safely exit the script without leaving any processes running.



```
#!/usr/bin/bash

MONITOR_DIR = "/home/kali/Ceyona/CYS/CYS/UNit1/glob/bak"

trap 'echo "Exiting ..."; exit 0' SIGINT

inotifywait -m "$MONITOR_DIR"
```

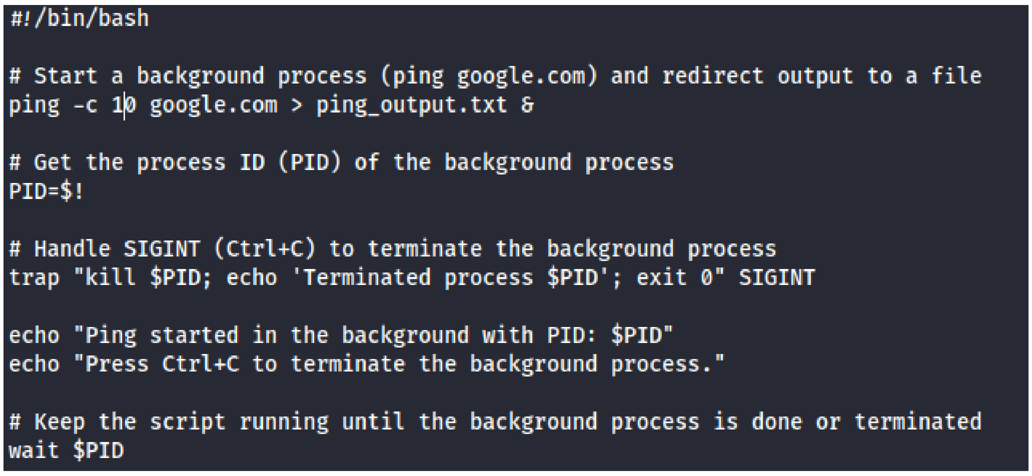


```
└─$ cd bak

┌──(kali㉿kali)-[~/…/CYS/Unit1/glob/bak]
└─$ ls -l
total 12
-rwxrwxrwx 1 kali kali   0 Oct 20 09:42 a.bak
-rwxrwxrwx 1 kali kali   0 Oct 20 09:42 b.bak
-rwxrwxrwx 1 kali kali   0 Oct 20 09:42 c.bak
-rwxrwxr-x 1 kali kali 210 Oct 20 09:51 check_users.sh
-rwxrwxrwx 1 kali kali   0 Oct 20 09:42 d.bak
-rwxrwxrwx 1 kali kali   0 Oct 20 09:42 e.bak
-rwxrwxr-x 1 kali kali 289 Oct 20 09:47 txt_to_bak.sh
-rw-rw-r-- 1 kali kali  36 Oct 20 09:50 users.txt

┌──(kali㉿kali)-[~/…/CYS/Unit1/glob/bak]
└─$ chmod -x a.bak b.bak c.bak
```

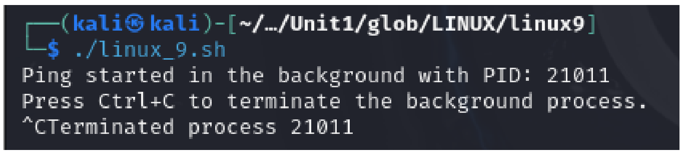


```
└─$ ./monitor.sh
Setting up watches.
Watches established.
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ ACCESS,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ CLOSE_NOWRITE,CLOSE,ISDIR
/home/kali/PRATIK/CYS/Unit1/glob/bak/ OPEN,ISDIR
```

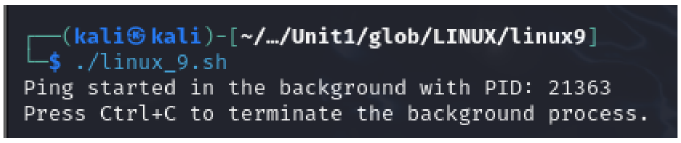
2. Write a bash script that starts a background process (of your choice), and use trap to catch SIGINT and terminate the process cleanly.



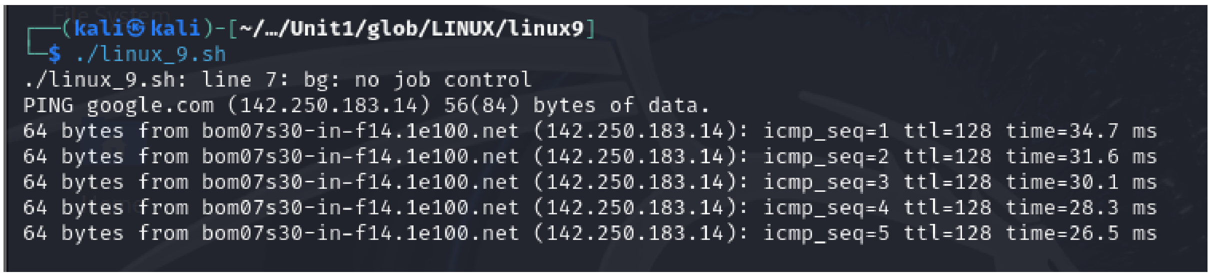
```bash
#!/bin/bash

# Start a background process (ping google.com) and redirect output to a file
ping -c 10 google.com > ping_output.txt &

# Get the process ID (PID) of the background process
PID=$!

# Handle SIGINT (Ctrl+C) to terminate the background process
trap "kill $PID; echo 'Terminated process $PID'; exit 0" SIGINT

echo "Ping started in the background with PID: $PID"
echo "Press Ctrl+C to terminate the background process."

# Keep the script running until the background process is done or terminated
wait $PID
```



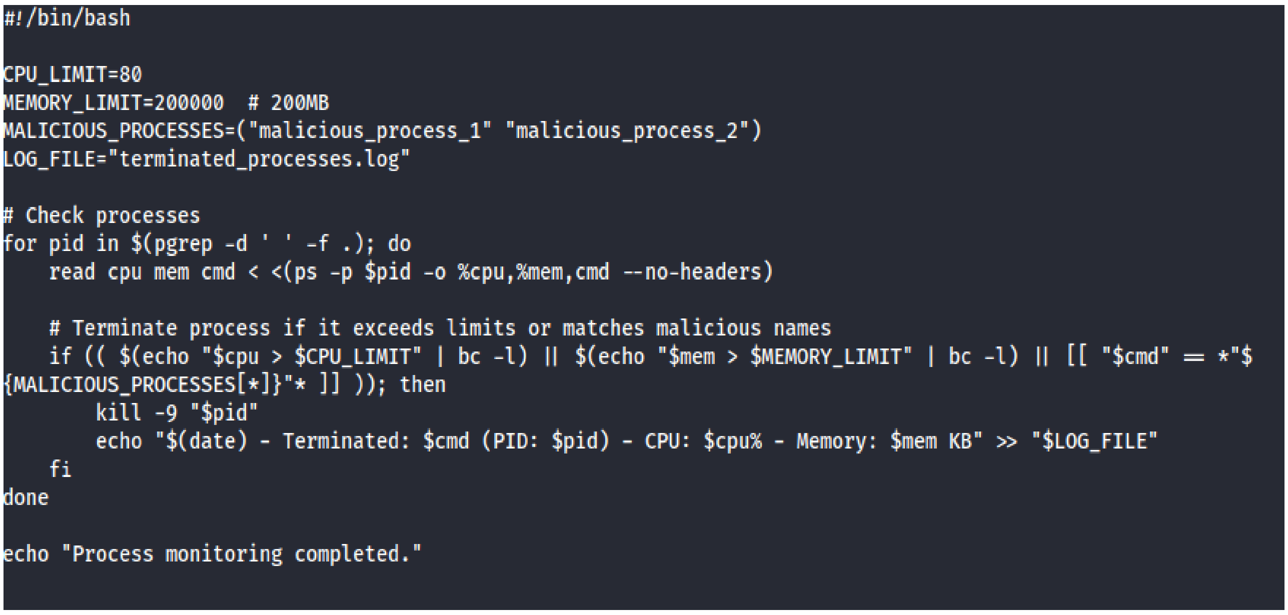
```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9.sh
Ping started in the background with PID: 21011
Press Ctrl+C to terminate the background process.
^CTerminated process 21011
```

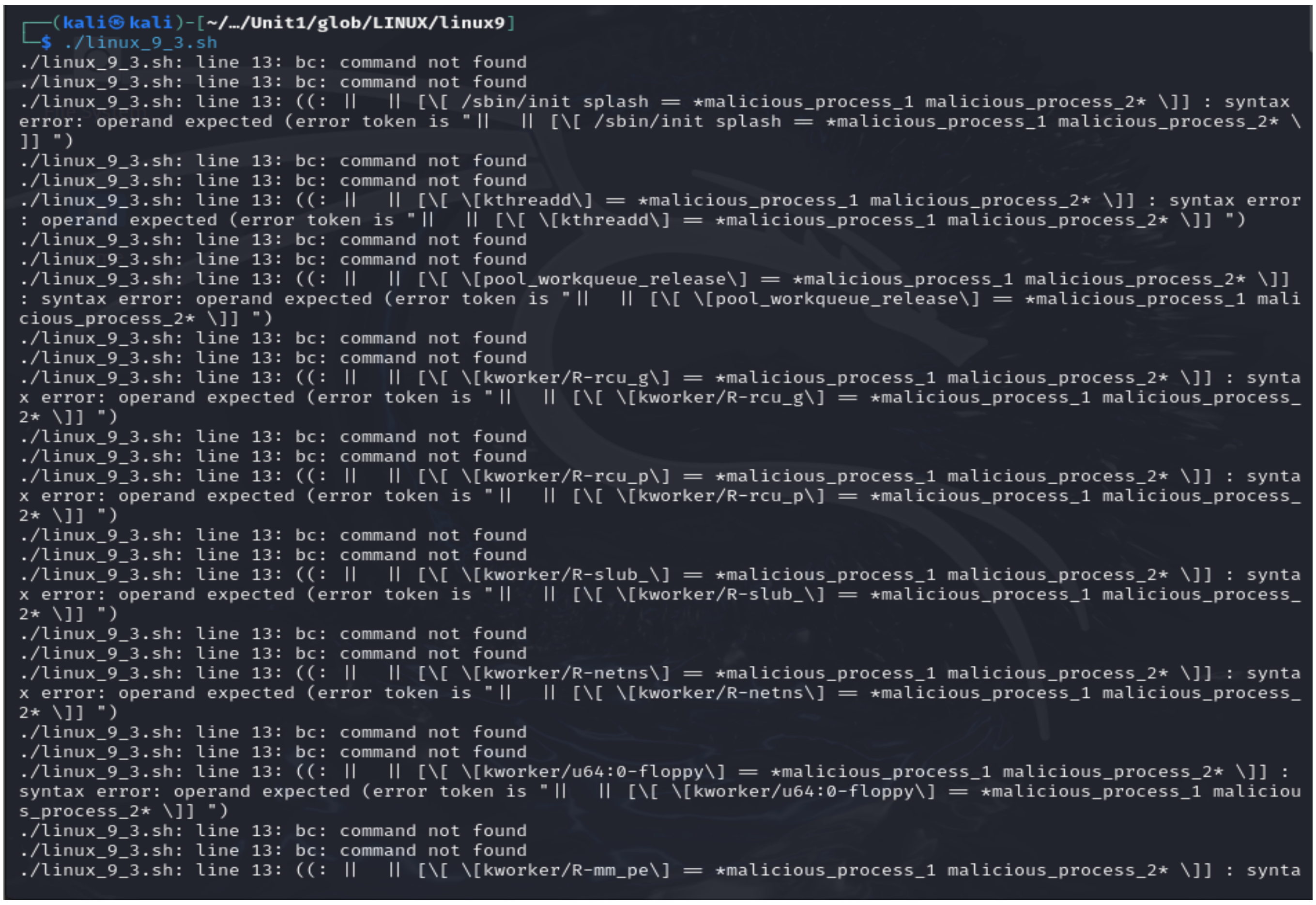

```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9.sh
Ping started in the background with PID: 21363
Press Ctrl+C to terminate the background process.
```

The process which was running in background


```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9.sh
./linux_9.sh: line 7: bg: no job control
PING google.com (142.250.183.14) 56(84) bytes of data.
64 bytes from bom07s30-in-f14.1e100.net (142.250.183.14): icmp_seq=1 ttl=128 time=34.7 ms
64 bytes from bom07s30-in-f14.1e100.net (142.250.183.14): icmp_seq=2 ttl=128 time=31.6 ms
64 bytes from bom07s30-in-f14.1e100.net (142.250.183.14): icmp_seq=3 ttl=128 time=30.1 ms
64 bytes from bom07s30-in-f14.1e100.net (142.250.183.14): icmp_seq=4 ttl=128 time=28.3 ms
64 bytes from bom07s30-in-f14.1e100.net (142.250.183.14): icmp_seq=5 ttl=128 time=26.5 ms
```

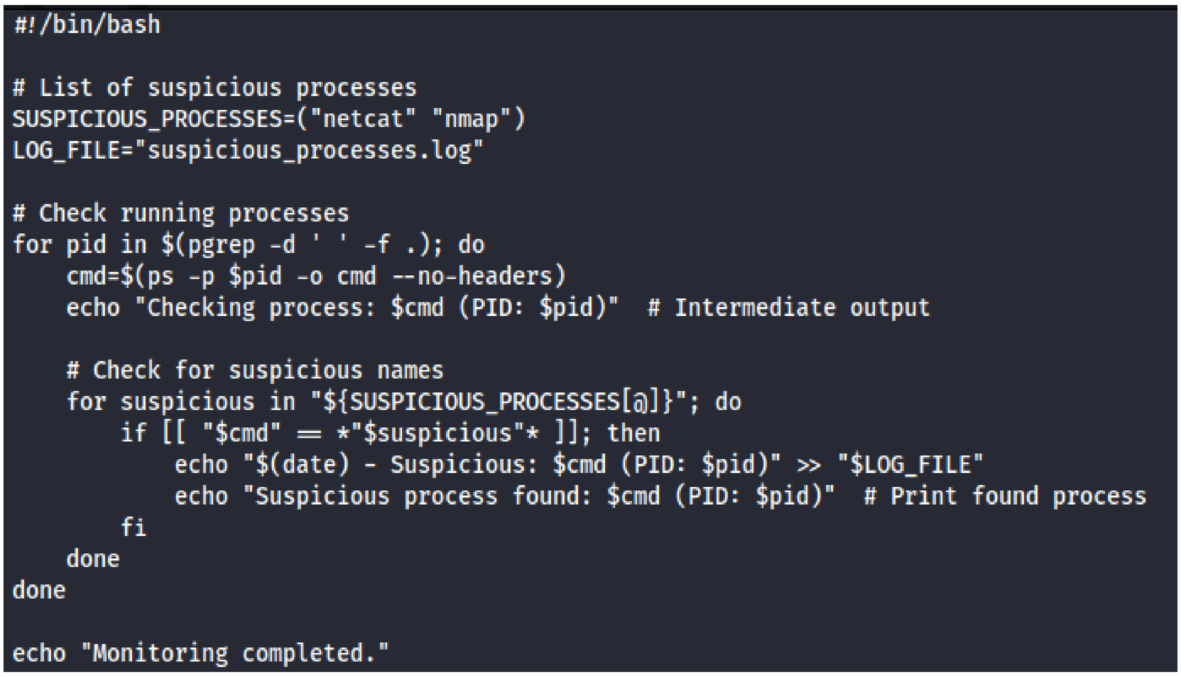
3. Write a script that kills any process exceeding a defined CPU or memory usage limit or matching a list of malicious process names. The script should log the terminated process details for auditing purposes.


```
#!/bin/bash

CPU_LIMIT=80
MEMORY_LIMIT=200000  # 200MB
MALICIOUS_PROCESSES=("malicious_process_1" "malicious_process_2")
LOG_FILE="terminated_processes.log"

# Check processes
for pid in $(pgrep -d ' ' -f .); do
    read cpu mem cmd < <(ps -p $pid -o %cpu,%mem,cmd --no-headers)

    # Terminate process if it exceeds limits or matches malicious names
    if (( $(echo "$cpu > $CPU_LIMIT" | bc -l) || $(echo "$mem > $MEMORY_LIMIT" | bc -l) || [[ "$cmd" == *"$
{MALICIOUS_PROCESSES[*]}"* ]] )); then
        kill -9 "$pid"
        echo "$(date) - Terminated: $cmd (PID: $pid) - CPU: $cpu% - Memory: $mem KB" >> "$LOG_FILE"
    fi
done

echo "Process monitoring completed."
```

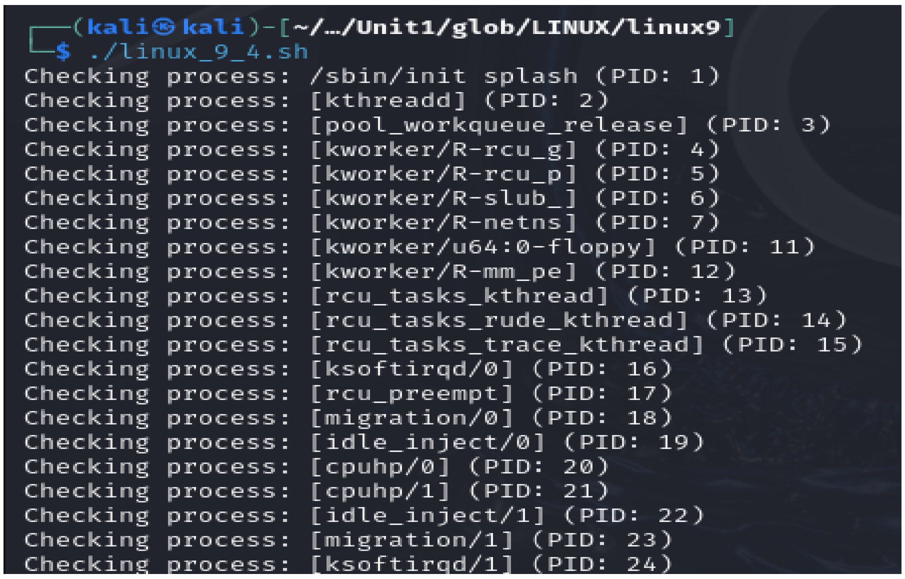


```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9_3.sh
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ /sbin/init splash == *malicious_process_1 malicious_process_2* \]] : syntax
error: operand expected (error token is "||   || [\[ /sbin/init splash == *malicious_process_1 malicious_process_2* \
]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kthreadd\] == *malicious_process_1 malicious_process_2* \]] : syntax error
: operand expected (error token is "||   || [\[ \[kthreadd\] == *malicious_process_1 malicious_process_2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[pool_workqueue_release\] == *malicious_process_1 malicious_process_2* \]]
: syntax error: operand expected (error token is "||   || [\[ \[pool_workqueue_release\] == *malicious_process_1 mali
cious_process_2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/R-rcu_g\] == *malicious_process_1 malicious_process_2* \]] : synta
x error: operand expected (error token is "||   || [\[ \[kworker/R-rcu_g\] == *malicious_process_1 malicious_process_
2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/R-rcu_p\] == *malicious_process_1 malicious_process_2* \]] : synta
x error: operand expected (error token is "||   || [\[ \[kworker/R-rcu_p\] == *malicious_process_1 malicious_process_
2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/R-slub_\] == *malicious_process_1 malicious_process_2* \]] : synta
x error: operand expected (error token is "||   || [\[ \[kworker/R-slub_\] == *malicious_process_1 malicious_process_
2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/R-netns\] == *malicious_process_1 malicious_process_2* \]] : synta
x error: operand expected (error token is "||   || [\[ \[kworker/R-netns\] == *malicious_process_1 malicious_process_
2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/u64:0-floppy\] == *malicious_process_1 malicious_process_2* \]] :
syntax error: operand expected (error token is "||   || [\[ \[kworker/u64:0-floppy\] == *malicious_process_1 maliciou
s_process_2* \]] ")
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: bc: command not found
./linux_9_3.sh: line 13: ((: ||   || [\[ \[kworker/R-mm_pe\] == *malicious_process_1 malicious_process_2* \]] : synta
```

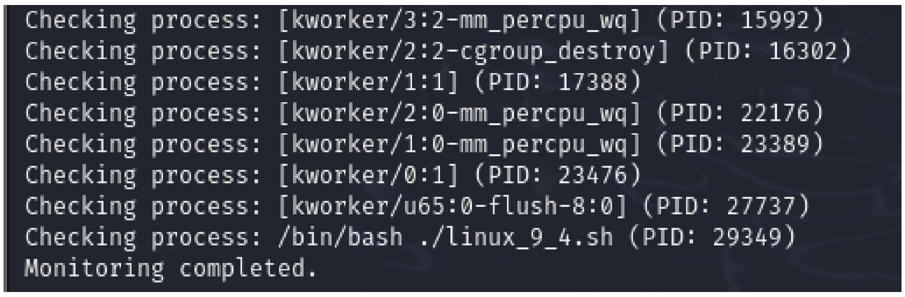
4. Write a script that monitors running processes and identifies any process that matches a list of known suspicious names (like netcat, nmap).



```bash
#!/bin/bash

# List of suspicious processes
SUSPICIOUS_PROCESSES=("netcat" "nmap")
LOG_FILE="suspicious_processes.log"

# Check running processes
for pid in $(pgrep -d ' ' -f .); do
    cmd=$(ps -p $pid -o cmd --no-headers)
    echo "Checking process: $cmd (PID: $pid)"  # Intermediate output

    # Check for suspicious names
    for suspicious in "${SUSPICIOUS_PROCESSES[@]}"; do
        if [[ "$cmd" == *"$suspicious"* ]]; then
            echo "$(date) - Suspicious: $cmd (PID: $pid)" >> "$LOG_FILE"
            echo "Suspicious process found: $cmd (PID: $pid)"  # Print found process
        fi
    done
done

echo "Monitoring completed."
```

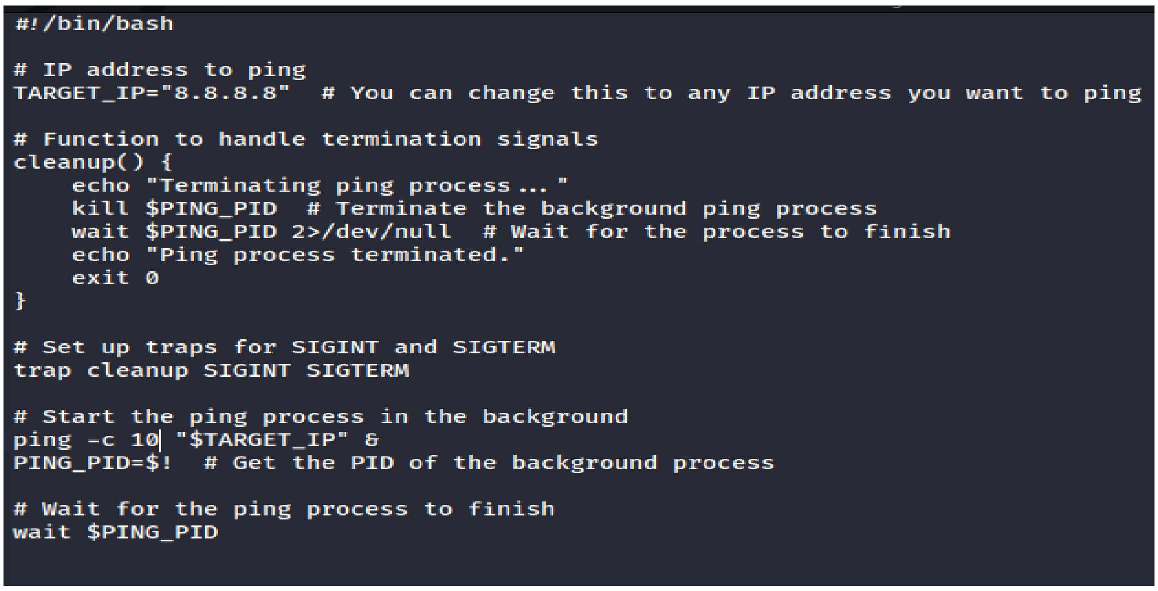


```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9_4.sh
Checking process: /sbin/init splash (PID: 1)
Checking process: [kthreadd] (PID: 2)
Checking process: [pool_workqueue_release] (PID: 3)
Checking process: [kworker/R-rcu_g] (PID: 4)
Checking process: [kworker/R-rcu_p] (PID: 5)
Checking process: [kworker/R-slub_] (PID: 6)
Checking process: [kworker/R-netns] (PID: 7)
Checking process: [kworker/u64:0-floppy] (PID: 11)
Checking process: [kworker/R-mm_pe] (PID: 12)
Checking process: [rcu_tasks_kthread] (PID: 13)
Checking process: [rcu_tasks_rude_kthread] (PID: 14)
Checking process: [rcu_tasks_trace_kthread] (PID: 15)
Checking process: [ksoftirqd/0] (PID: 16)
Checking process: [rcu_preempt] (PID: 17)
Checking process: [migration/0] (PID: 18)
Checking process: [idle_inject/0] (PID: 19)
Checking process: [cpuhp/0] (PID: 20)
Checking process: [cpuhp/1] (PID: 21)
Checking process: [idle_inject/1] (PID: 22)
Checking process: [migration/1] (PID: 23)
Checking process: [ksoftirqd/1] (PID: 24)
```

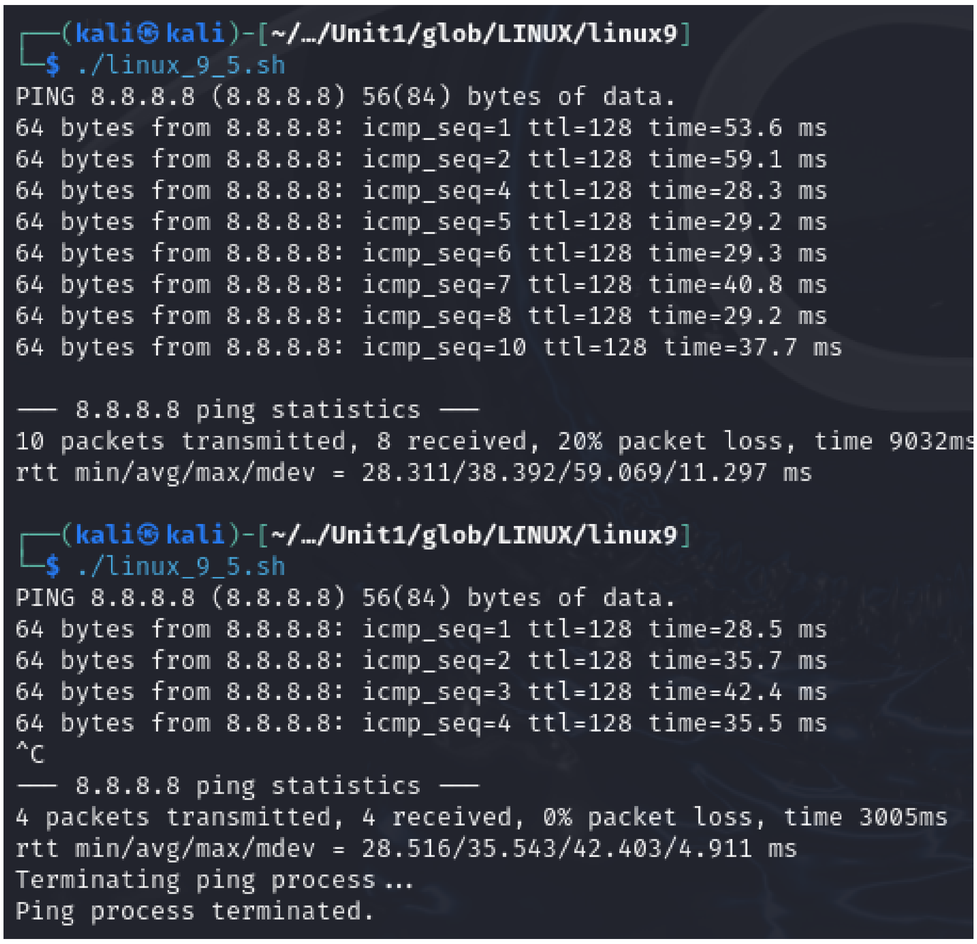


```
Checking process: [kworker/3:2-mm_percpu_wq] (PID: 15992)
Checking process: [kworker/2:2-cgroup_destroy] (PID: 16302)
Checking process: [kworker/1:1] (PID: 17388)
Checking process: [kworker/2:0-mm_percpu_wq] (PID: 22176)
Checking process: [kworker/1:0-mm_percpu_wq] (PID: 23389)
Checking process: [kworker/0:1] (PID: 23476)
Checking process: [kworker/u65:0-flush-8:0] (PID: 27737)
Checking process: /bin/bash ./linux_9_4.sh (PID: 29349)
Monitoring completed.
```

5. Create a script that runs a background process (such as a continuous ping to a specified IP address). Use trap to capture termination signals (SIGINT, SIGTERM) and ensure the background process is terminated safely when the script is interrupted.

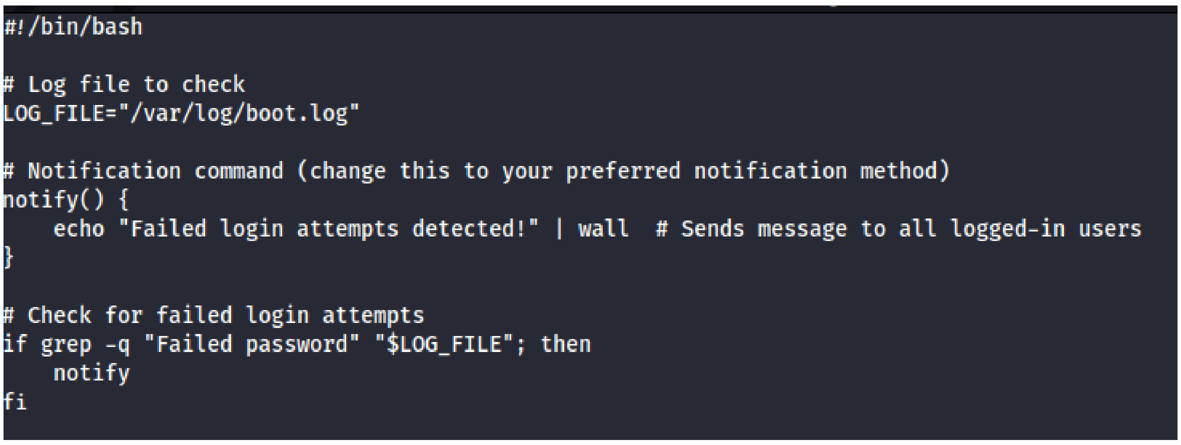


```
#!/bin/bash

# IP address to ping
TARGET_IP="8.8.8.8"  # You can change this to any IP address you want to ping

# Function to handle termination signals
cleanup() {
    echo "Terminating ping process..."
    kill $PING_PID  # Terminate the background ping process
    wait $PING_PID 2>/dev/null  # Wait for the process to finish
    echo "Ping process terminated."
    exit 0
}

# Set up traps for SIGINT and SIGTERM
trap cleanup SIGINT SIGTERM

# Start the ping process in the background
ping -c 10 "$TARGET_IP" &
PING_PID=$!  # Get the PID of the background process

# Wait for the ping process to finish
wait $PING_PID
```

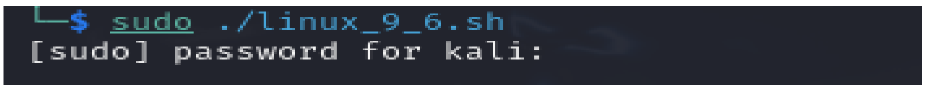


```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9_5.sh
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=53.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=59.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=28.3 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=128 time=29.2 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=128 time=29.3 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=128 time=40.8 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=128 time=29.2 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=128 time=37.7 ms

--- 8.8.8.8 ping statistics ---
10 packets transmitted, 8 received, 20% packet loss, time 9032ms
rtt min/avg/max/mdev = 28.311/38.392/59.069/11.297 ms

┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9_5.sh
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=28.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=35.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=42.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=35.5 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 28.516/35.543/42.403/4.911 ms
Terminating ping process...
Ping process terminated.
```

6. Write a script that checks /var/log/auth.log for failed login attempts and sends notification if any are found. Schedule this script to run every 15 minutes using cron command.



```
#!/bin/bash

# Log file to check
LOG_FILE="/var/log/boot.log"

# Notification command (change this to your preferred notification method)
notify() {
    echo "Failed login attempts detected!" | wall  # Sends message to all logged-in users
}

# Check for failed login attempts
if grep -q "Failed password" "$LOG_FILE"; then
    notify
fi
```



```
└─$ sudo ./linux_9_6.sh
[sudo] password for kali:
```

Scheduling with Cron:

To schedule the script to run every 15 minutes, follow these steps:

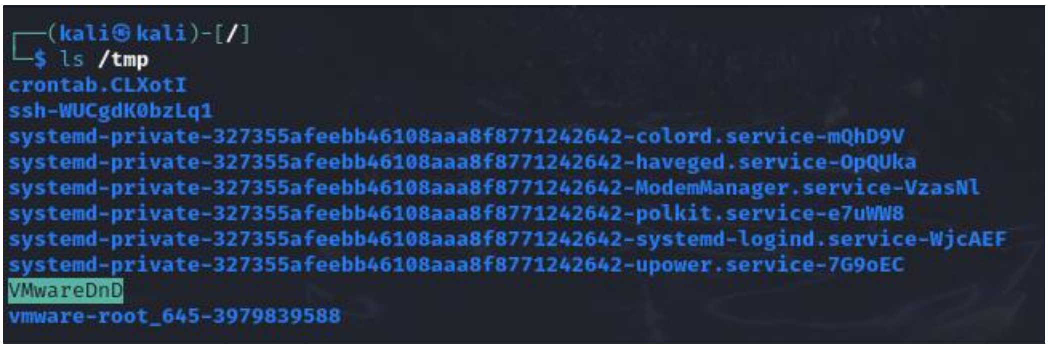
1. Open the crontab editor:

crontab -e

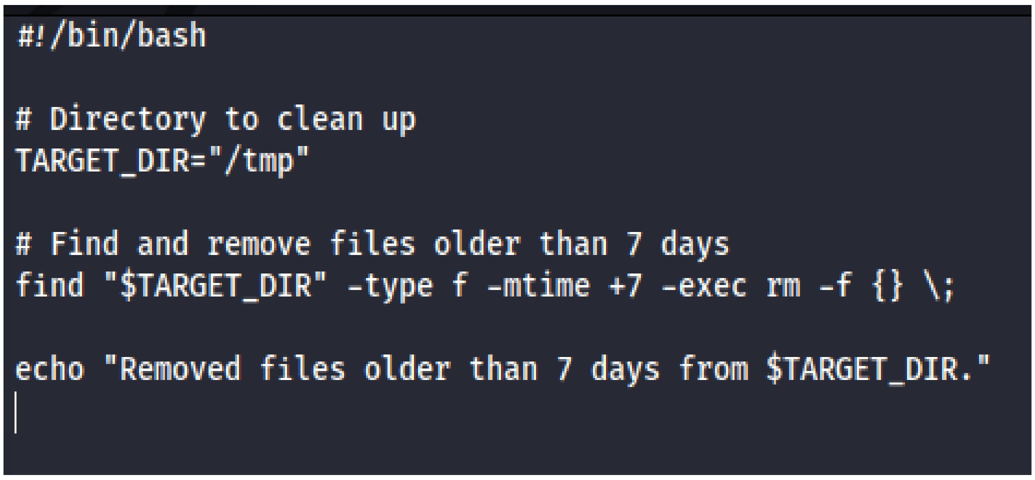
2. Add the following line to the crontab file:

***/15 * * * * /path/to/check_failed_logins.sh**

7. Write a script that removes all files older than 7 days from the /tmp directory, and use at to schedule the script to run at 2:00 AM the next day.

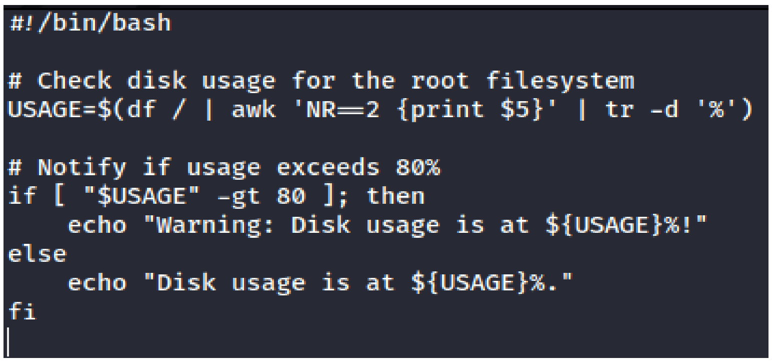


```
┌──(kali㉿kali)-[/]
└─$ ls /tmp
crontab.CLXotI
ssh-WUCgdK0bzLq1
systemd-private-327355afeebb46108aaa8f8771242642-colord.service-mQhD9V
systemd-private-327355afeebb46108aaa8f8771242642-haveged.service-OpQUka
systemd-private-327355afeebb46108aaa8f8771242642-ModemManager.service-VzasNl
systemd-private-327355afeebb46108aaa8f8771242642-polkit.service-e7uWW8
systemd-private-327355afeebb46108aaa8f8771242642-systemd-logind.service-WjcAEF
systemd-private-327355afeebb46108aaa8f8771242642-upower.service-7G9oEC
VMwareDnD
vmware-root_645-3979839588
```

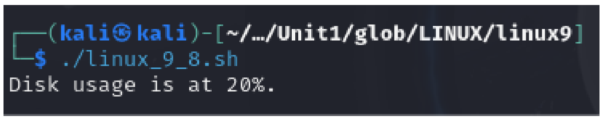


```
#!/bin/bash

# Directory to clean up
TARGET_DIR="/tmp"

# Find and remove files older than 7 days
find "$TARGET_DIR" -type f -mtime +7 -exec rm -f {} \;

echo "Removed files older than 7 days from $TARGET_DIR."
```

8. Write a script to check if disk usage exceeds 80%, and use at to schedule it to run at a specific time.



```
#!/bin/bash

# Check disk usage for the root filesystem
USAGE=$(df / | awk 'NR==2 {print $5}' | tr -d '%')

# Notify if usage exceeds 80%
if [ "$USAGE" -gt 80 ]; then
    echo "Warning: Disk usage is at ${USAGE}%!"
else
    echo "Disk usage is at ${USAGE}%."
fi
```



```
┌──(kali㉿kali)-[~/…/Unit1/glob/LINUX/linux9]
└─$ ./linux_9_8.sh
Disk usage is at 20%.
```