Placement Empowerment Program

Cloud Computing and DevOps Centre

60 Days of DevOps Challenge

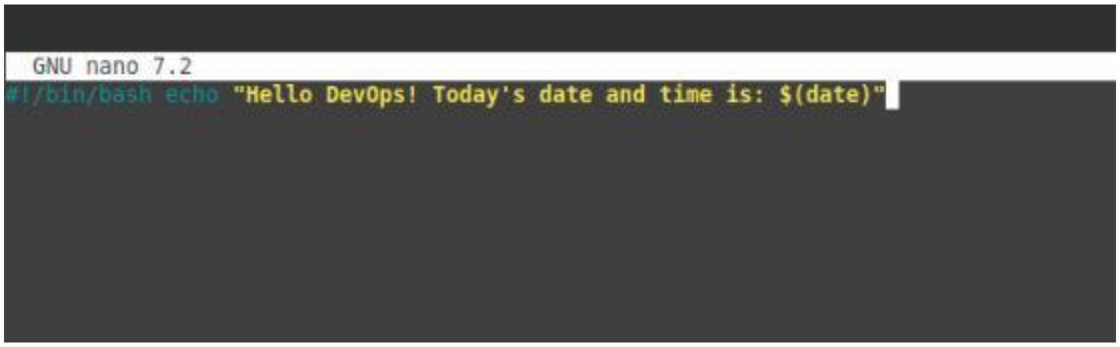
Day 2: Linux Shell Scripting & Automation

Name: VIJAYA NANDANA M *Department: CSE*

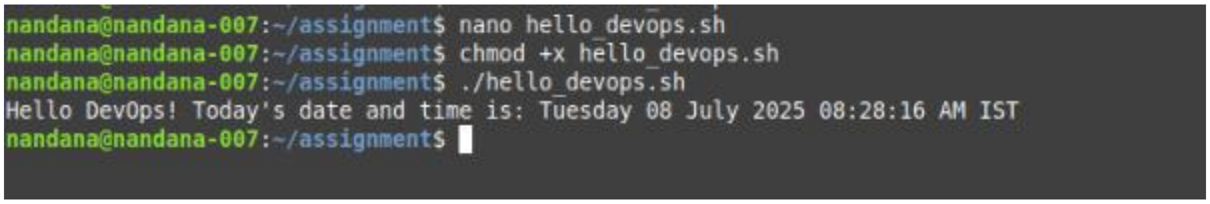
Introduction

This Proof of Concept (PoC) demonstrates Linux Shell Scripting & Automation

Challenge 1: Write a simple Bash script that prints "Hello DevOps" along with the current date and time.

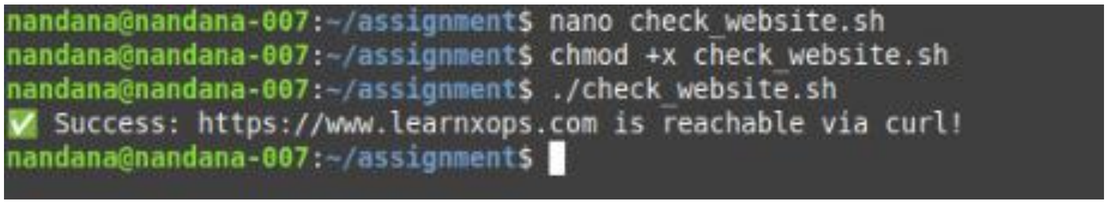


```
  GNU nano 7.2
#!/bin/bash echo "Hello DevOps! Today's date and time is: $(date)"
```

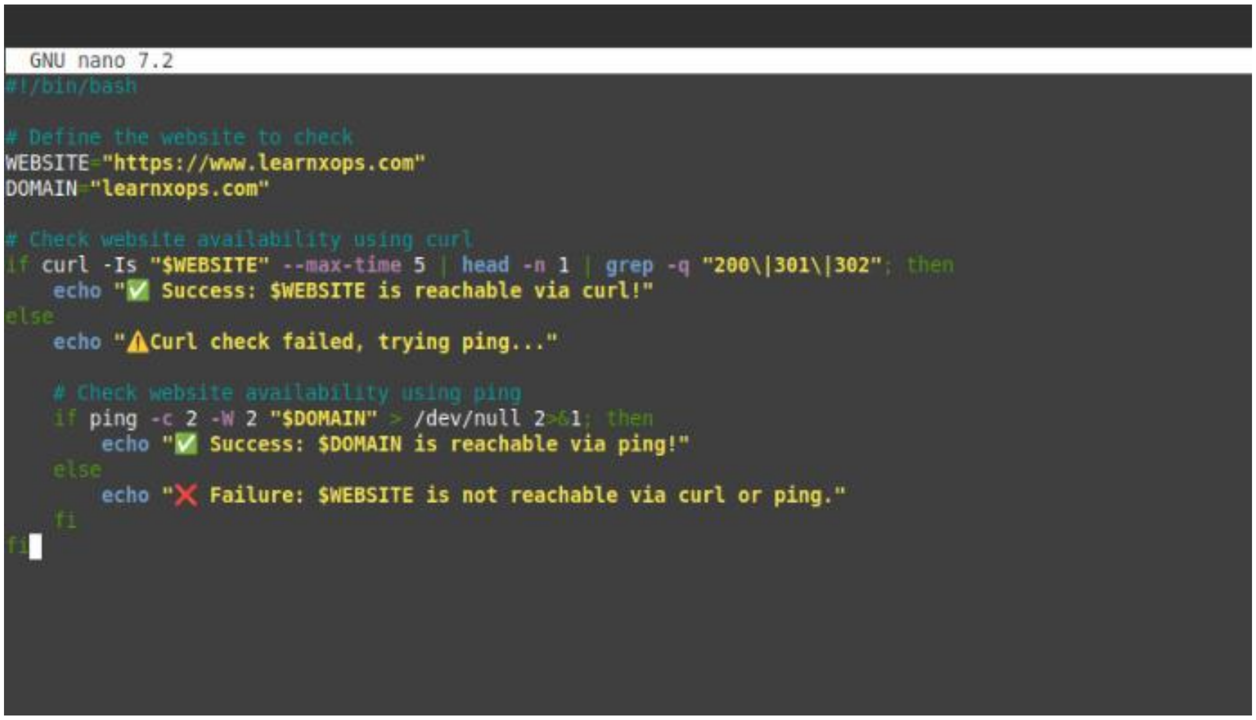


```
nandana@nandana-007:~/assignment$ nano hello_devops.sh
nandana@nandana-007:~/assignment$ chmod +x hello_devops.sh
nandana@nandana-007:~/assignment$ ./hello_devops.sh
Hello DevOps! Today's date and time is: Tuesday 08 July 2025 08:28:16 AM IST
nandana@nandana-007:~/assignment$ 
```

Challenge 2: Create a script that checks if a website is reachable using curl or ping. Print a success or failure message.

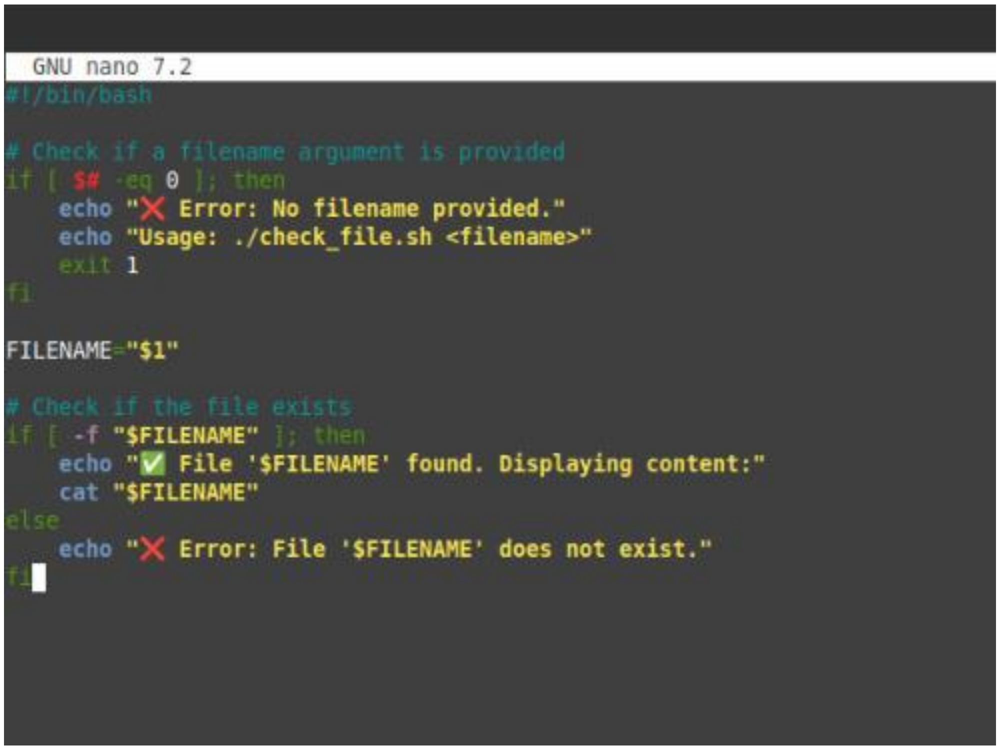


```
nandana@nandana-007:~/assignment$ nano check_website.sh
nandana@nandana-007:~/assignment$ chmod +x check_website.sh
nandana@nandana-007:~/assignment$ ./check_website.sh
✅ Success: https://www.learnxops.com is reachable via curl!
nandana@nandana-007:~/assignment$ 
```

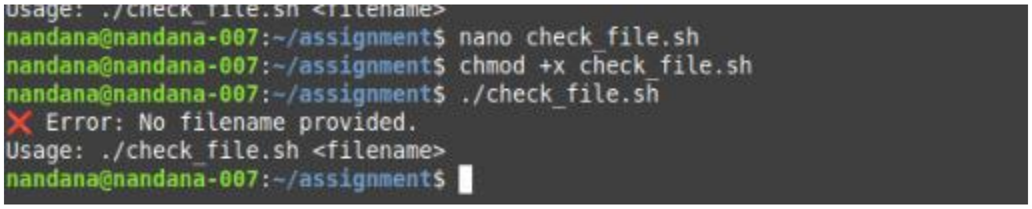


```
GNU nano 7.2
#!/bin/bash

# Define the website to check
WEBSITE="https://www.learnxops.com"
DOMAIN="learnxops.com"

# Check website availability using curl
if curl -Is "$WEBSITE" --max-time 5 | head -n 1 | grep -q "200\|301\|302"; then
    echo "✅ Success: $WEBSITE is reachable via curl!"
else
    echo "⚠️Curl check failed, trying ping..."

    # Check website availability using ping
    if ping -c 2 -W 2 "$DOMAIN" > /dev/null 2>&1; then
        echo "✅ Success: $DOMAIN is reachable via ping!"
    else
        echo "❌ Failure: $WEBSITE is not reachable via curl or ping."
    fi
fi
```

Challenge 3: Write a script that takes a filename as an argument, checks if it exists, and prints the content of the file accordingly.

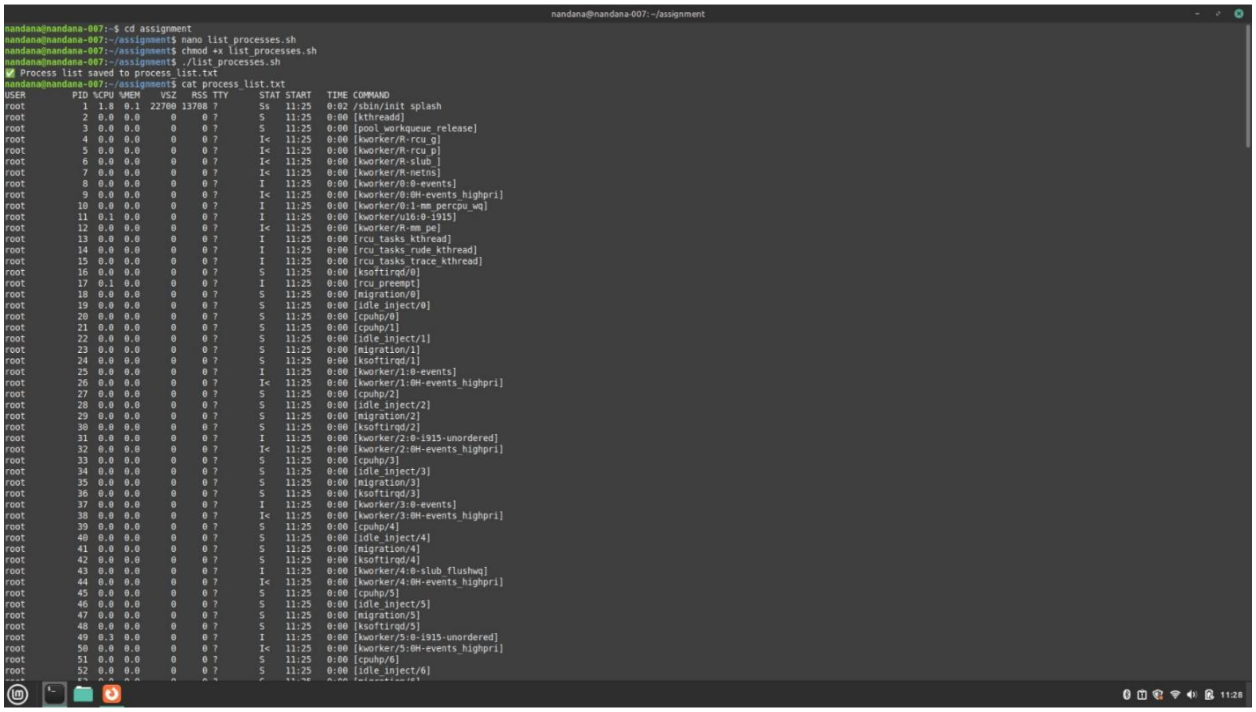


```
GNU nano 7.2
#!/bin/bash

# Check if a filename argument is provided
if [ $# -eq 0 ]; then
    echo "❌ Error: No filename provided."
    echo "Usage: ./check_file.sh <filename>"
    exit 1
fi

FILENAME="$1"

# Check if the file exists
if [ -f "$FILENAME" ]; then
    echo "✅ File '$FILENAME' found. Displaying content:"
    cat "$FILENAME"
else
    echo "❌ Error: File '$FILENAME' does not exist."
fi
```



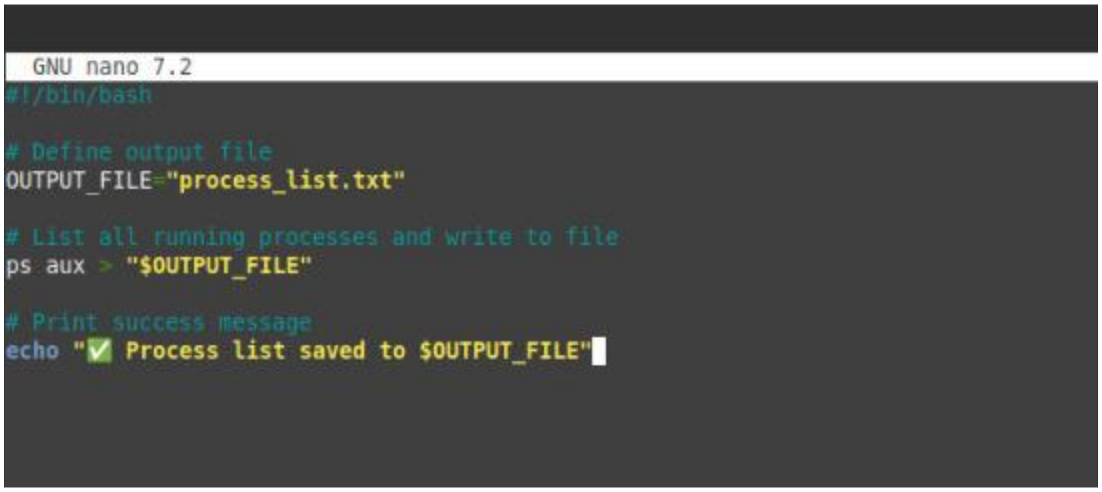
```
Usage: ./check_file.sh <filename>
nandana@nandana-007:~/assignment$ nano check_file.sh
nandana@nandana-007:~/assignment$ chmod +x check_file.sh
nandana@nandana-007:~/assignment$ ./check_file.sh
❌ Error: No filename provided.
Usage: ./check_file.sh <filename>
nandana@nandana-007:~/assignment$ 
```

Challenge 4: Create a script that lists all running processes and writes the output to a file named process_list.txt.



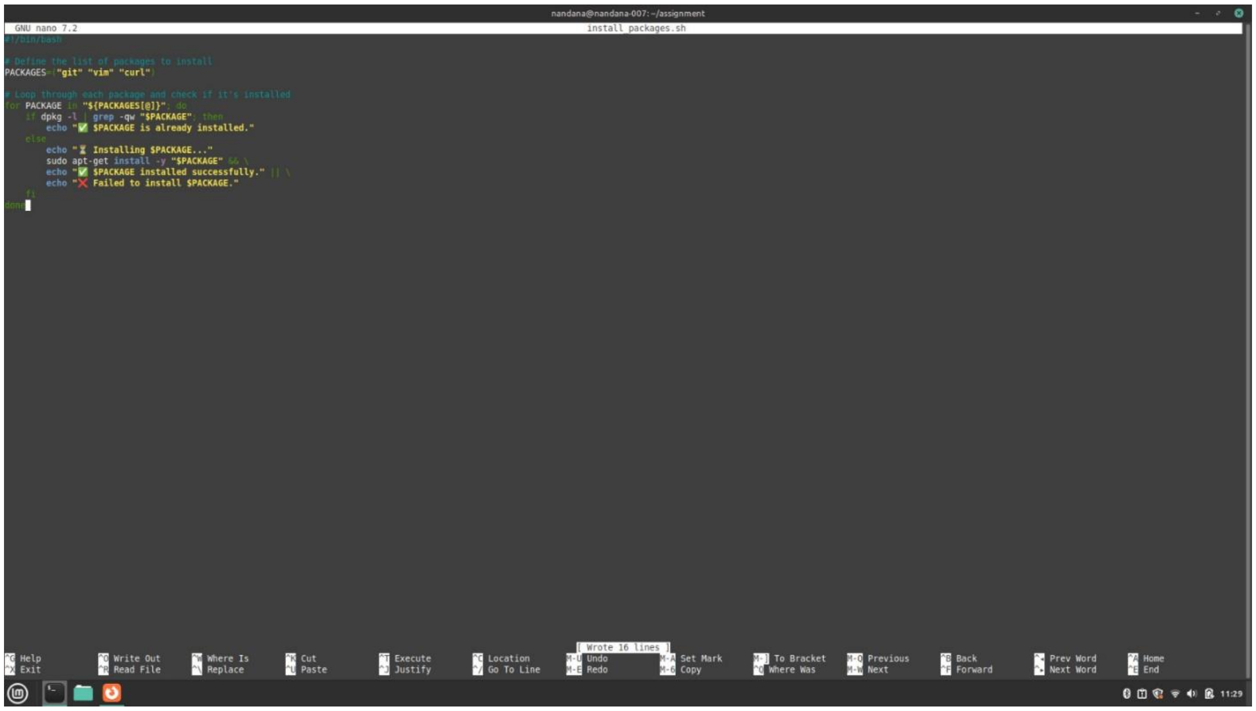
nandana@nandana-007: ~/assignment

```
nandana@nandana-007:~$ cd assignment
nandana@nandana-007:~/assignment$ nano list_processes.sh
nandana@nandana-007:~/assignment$ chmod +x list_processes.sh
nandana@nandana-007:~/assignment$ ./list_processes.sh
✅ Process list saved to process_list.txt
nandana@nandana-007:~/assignment$ cat process_list.txt
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  1.8  0.1  22700 13708 ?        Ss   11:25   0:02 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    11:25   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    11:25   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/R-netns]
root           8  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/0:0-events]
root           9  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/0:0H-events_highpri]
root          10  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/0:1-mm_percpu_wq]
root          11  0.1  0.0      0     0 ?        I    11:25   0:00 [kworker/u16:0-i915]
root          12  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/R-mm_pe]
root          13  0.0  0.0      0     0 ?        I    11:25   0:00 [rcu_tasks_kthread]
root          14  0.0  0.0      0     0 ?        I    11:25   0:00 [rcu_tasks_rude_kthread]
root          15  0.0  0.0      0     0 ?        I    11:25   0:00 [rcu_tasks_trace_kthread]
root          16  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/0]
root          17  0.1  0.0      0     0 ?        I    11:25   0:00 [rcu_preempt]
root          18  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/0]
root          19  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/0]
root          20  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/0]
root          21  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/1]
root          22  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/1]
root          23  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/1]
root          24  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/1]
root          25  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/1:0-events]
root          26  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/1:0H-events_highpri]
root          27  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/2]
root          28  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/2]
root          29  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/2]
root          30  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/2]
root          31  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/2:0-i915-unordered]
root          32  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/2:0H-events_highpri]
root          33  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/3]
root          34  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/3]
root          35  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/3]
root          36  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/3]
root          37  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/3:0-events]
root          38  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/3:0H-events_highpri]
root          39  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/4]
root          40  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/4]
root          41  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/4]
root          42  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/4]
root          43  0.0  0.0      0     0 ?        I    11:25   0:00 [kworker/4:0-slub_flushwq]
root          44  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/4:0H-events_highpri]
root          45  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/5]
root          46  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/5]
root          47  0.0  0.0      0     0 ?        S    11:25   0:00 [migration/5]
root          48  0.0  0.0      0     0 ?        S    11:25   0:00 [ksoftirqd/5]
root          49  0.3  0.0      0     0 ?        I    11:25   0:00 [kworker/5:0-i915-unordered]
root          50  0.0  0.0      0     0 ?        I<   11:25   0:00 [kworker/5:0H-events_highpri]
root          51  0.0  0.0      0     0 ?        S    11:25   0:00 [cpuhp/6]
root          52  0.0  0.0      0     0 ?        S    11:25   0:00 [idle_inject/6]
```

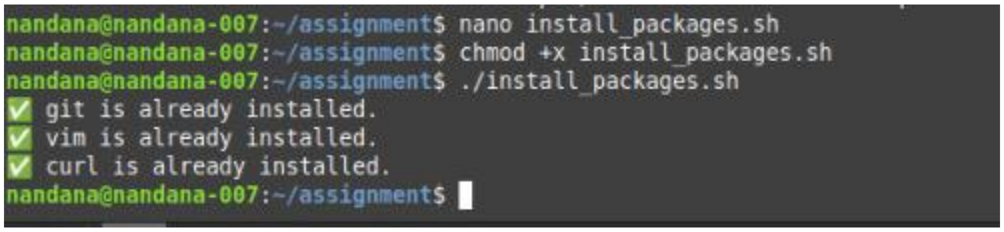


```
  GNU nano 7.2
#!/bin/bash

# Define output file
OUTPUT_FILE="process_list.txt"

# List all running processes and write to file
ps aux > "$OUTPUT_FILE"

# Print success message
echo "✅ Process list saved to $OUTPUT_FILE"
```

Challenge 5: Write a script that installs multiple packages at once (e.g., git, vim, curl). The script should check if each package is already installed before attempting installation.

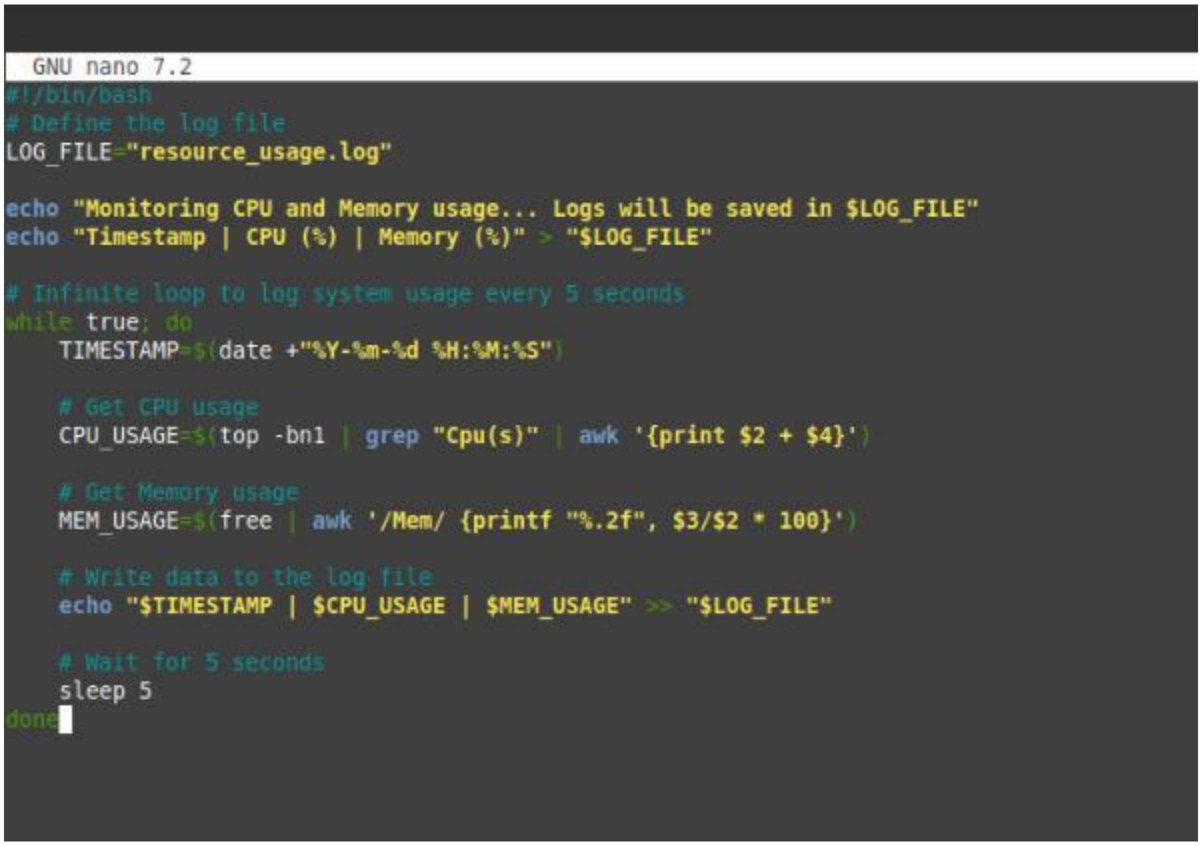


```
nandana@nandana-007: ~/assignment
GNU nano 7.2                         install_packages.sh
#!/bin/bash

# Define the list of packages to install
PACKAGES=("git" "vim" "curl")

# Loop through each package and check if it's installed
for PACKAGE in "${PACKAGES[@]}"; do
    if dpkg -l | grep -qw "$PACKAGE"; then
        echo "✅ $PACKAGE is already installed."
    else
        echo "⏳ Installing $PACKAGE..."
        sudo apt-get install -y "$PACKAGE" && \
        echo "✅ $PACKAGE installed successfully." || \
        echo "❌ Failed to install $PACKAGE."
    fi
done

                              [ Wrote 16 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket  M-Q Previous  ^B Back       ^◂ Prev Word  ^A Home
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was    M-W Next      ^F Forward    ^▸ Next Word  ^E End
```

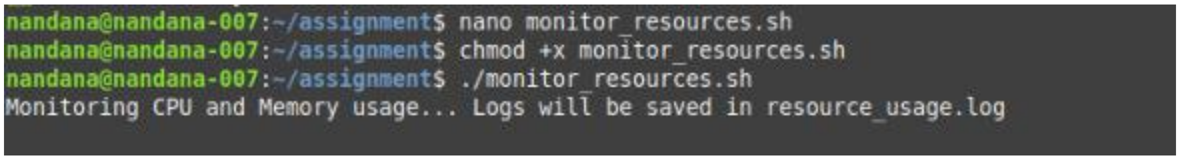


```
nandana@nandana-007:~/assignment$ nano install_packages.sh
nandana@nandana-007:~/assignment$ chmod +x install_packages.sh
nandana@nandana-007:~/assignment$ ./install_packages.sh
✅ git is already installed.
✅ vim is already installed.
✅ curl is already installed.
nandana@nandana-007:~/assignment$
```

Challenge 6: Create a script that monitors CPU and memory usage every 5 seconds and logs the results to a file.



```
 GNU nano 7.2
#!/bin/bash
# Define the log file
LOG_FILE="resource_usage.log"

echo "Monitoring CPU and Memory usage... Logs will be saved in $LOG_FILE"
echo "Timestamp | CPU (%) | Memory (%)" > "$LOG_FILE"

# Infinite loop to log system usage every 5 seconds
while true; do
    TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")

    # Get CPU usage
    CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')

    # Get Memory usage
    MEM_USAGE=$(free | awk '/Mem/ {printf "%.2f", $3/$2 * 100}')

    # Write data to the log file
    echo "$TIMESTAMP | $CPU_USAGE | $MEM_USAGE" >> "$LOG_FILE"

    # Wait for 5 seconds
    sleep 5
done
```



```
nandana@nandana-007:~/assignment$ nano monitor_resources.sh
nandana@nandana-007:~/assignment$ chmod +x monitor_resources.sh
nandana@nandana-007:~/assignment$ ./monitor_resources.sh
Monitoring CPU and Memory usage... Logs will be saved in resource_usage.log
```