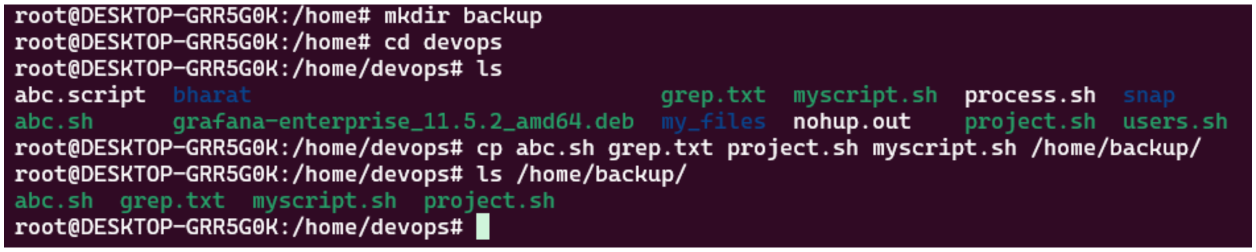
Task 1: File Management Script

Write a Bash script that

- Creates a directory named "backup" in the user's home directory

- Copies files from the current directory into the "backup" directory

As shown below we have created a directory in user's home directory and copied files into it.

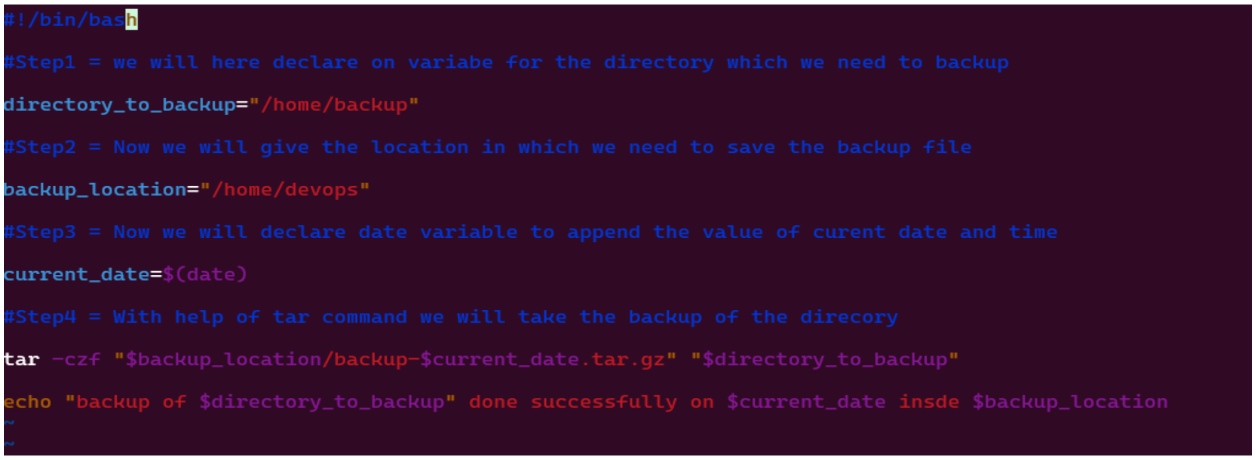


```
root@DESKTOP-GRR5G0K:/home# mkdir backup
root@DESKTOP-GRR5G0K:/home# cd devops
root@DESKTOP-GRR5G0K:/home/devops# ls
abc.script  bharat                             grep.txt  myscript.sh  process.sh  snap
abc.sh      grafana-enterprise_11.5.2_amd64.deb  my_files  nohup.out    project.sh  users.sh
root@DESKTOP-GRR5G0K:/home/devops# cp abc.sh grep.txt project.sh myscript.sh /home/backup/
root@DESKTOP-GRR5G0K:/home/devops# ls /home/backup/
abc.sh  grep.txt  myscript.sh  project.sh
root@DESKTOP-GRR5G0K:/home/devops#
```

Now we will write a script to take the backup of backup folder in other location in /home/devops directory.

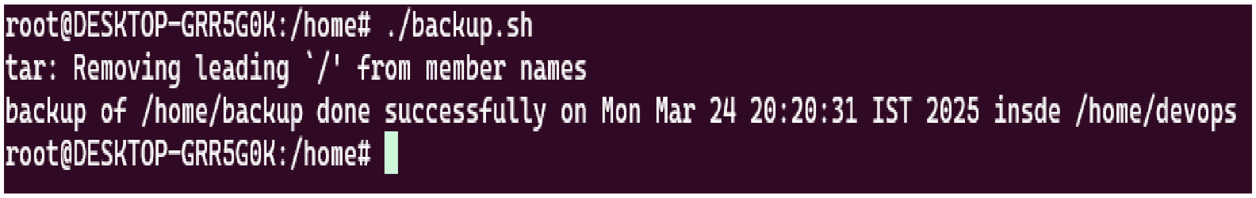
-Appends the current date and time to the filenames of the copied files

For this we will write the script as shown below



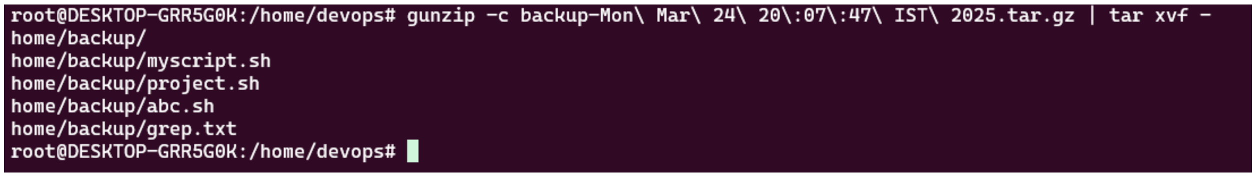
```
#!/bin/bash
#Step1 = we will here declare on variabe for the directory which we need to backup
directory_to_backup="/home/backup"
#Step2 = Now we will give the location in which we need to save the backup file
backup_location="/home/devops"
#Step3 = Now we will declare date variable to append the value of curent date and time
current_date=$(date)
#Step4 = With help of tar command we will take the backup of the direcory
tar -czf "$backup_location/backup-$current_date.tar.gz" "$directory_to_backup"
echo "backup of $directory_to_backup" done successfully on $current_date insde $backup_location
~
~
```

After then we will run the script **backup.sh**



```
root@DESKTOP-GRR5G0K:/home# ./backup.sh
tar: Removing leading `/' from member names
backup of /home/backup done successfully on Mon Mar 24 20:20:31 IST 2025 insde /home/devops
root@DESKTOP-GRR5G0K:/home#
```

We can check the files in the backed up file as by following command



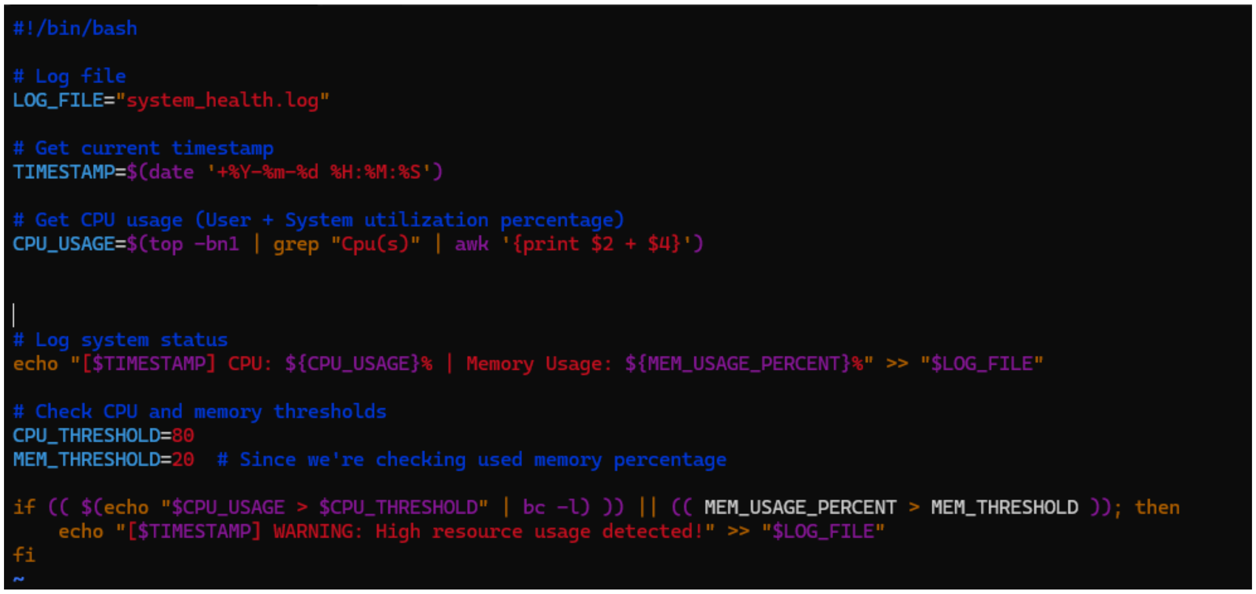
```
root@DESKTOP-GRR5G0K:/home/devops# gunzip -c backup-Mon\ Mar\ 24\ 20\:07\:47\ IST\ 2025.tar.gz | tar xvf -
home/backup/
home/backup/myscript.sh
home/backup/project.sh
home/backup/abc.sh
home/backup/grep.txt
root@DESKTOP-GRR5G0K:/home/devops#
```

Task 2: System Health Check

Create a script that

- Checks the system's CPU and memory usage
- Reports if the CPU usage is above 80% or if the available memory is below 20%
- Logs the results to a file named system_health.log.

For this we will create below script to monitor cpu and memory

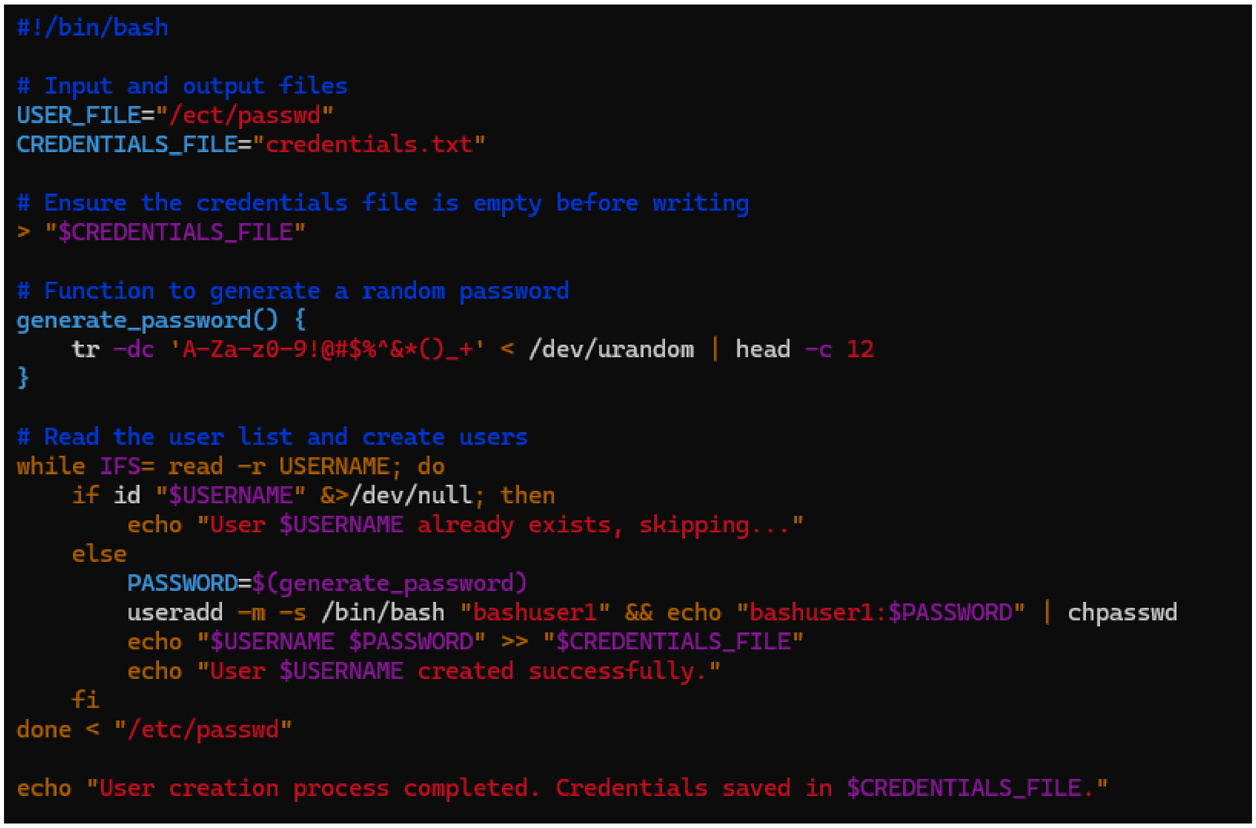


```
#!/bin/bash

# Log file
LOG_FILE="system_health.log"

# Get current timestamp
TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

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


# Log system status
echo "[$TIMESTAMP] CPU: ${CPU_USAGE}% | Memory Usage: ${MEM_USAGE_PERCENT}%" >> "$LOG_FILE"

# Check CPU and memory thresholds
CPU_THRESHOLD=80
MEM_THRESHOLD=20  # Since we're checking used memory percentage

if (( $(echo "$CPU_USAGE > $CPU_THRESHOLD" | bc -l) )) || (( MEM_USAGE_PERCENT > MEM_THRESHOLD )); then
    echo "[$TIMESTAMP] WARNING: High resource usage detected!" >> "$LOG_FILE"
fi
~
```

3. Write a script that

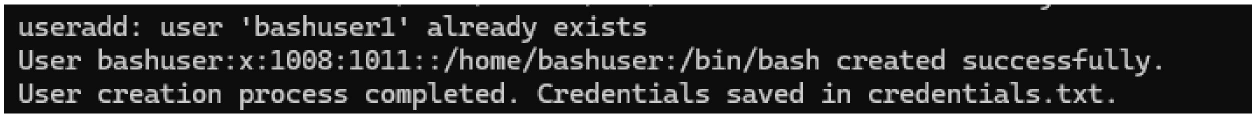
- Reads a list of usernames from a file (e.g., user_list.txt)
- Creates a new user for each username
- Generates a random password for each user and saves the username and password to a file named credentials.txt.

For this we will use below mentioned script

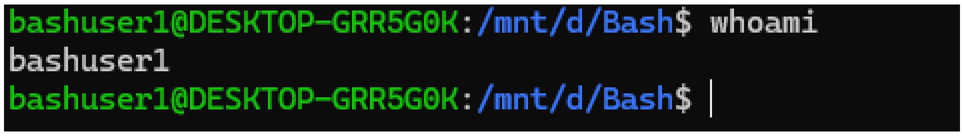


```bash
#!/bin/bash

# Input and output files
USER_FILE="/ect/passwd"
CREDENTIALS_FILE="credentials.txt"

# Ensure the credentials file is empty before writing
> "$CREDENTIALS_FILE"

# Function to generate a random password
generate_password() {
    tr -dc 'A-Za-z0-9!@#$%^&*()_+' < /dev/urandom | head -c 12
}

# Read the user list and create users
while IFS= read -r USERNAME; do
    if id "$USERNAME" &>/dev/null; then
        echo "User $USERNAME already exists, skipping..."
    else
        PASSWORD=$(generate_password)
        useradd -m -s /bin/bash "bashuser1" && echo "bashuser1:$PASSWORD" | chpasswd
        echo "$USERNAME $PASSWORD" >> "$CREDENTIALS_FILE"
        echo "User $USERNAME created successfully."
    fi
done < "/etc/passwd"

echo "User creation process completed. Credentials saved in $CREDENTIALS_FILE."
```

After then we can see bashuser1 created successfully.



```
useradd: user 'bashuser1' already exists
User bashuser:x:1008:1011::/home/bashuser:/bin/bash created successfully.
User creation process completed. Credentials saved in credentials.txt.
```

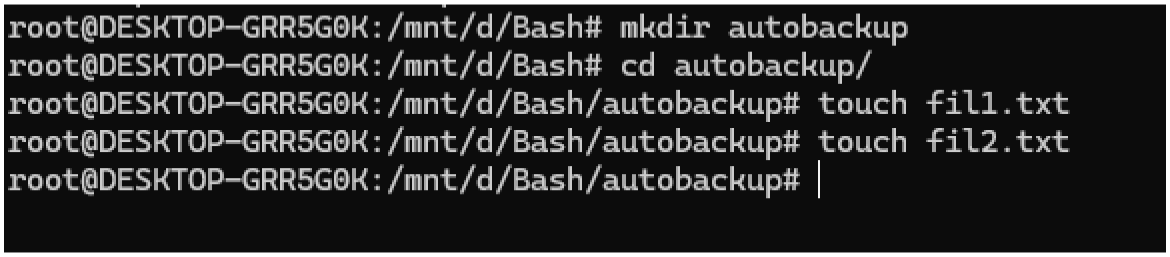


```
bashuser1@DESKTOP-GRR5G0K:/mnt/d/Bash$ whoami
bashuser1
bashuser1@DESKTOP-GRR5G0K:/mnt/d/Bash$
```

Task 4: Automated Backup Create a script that

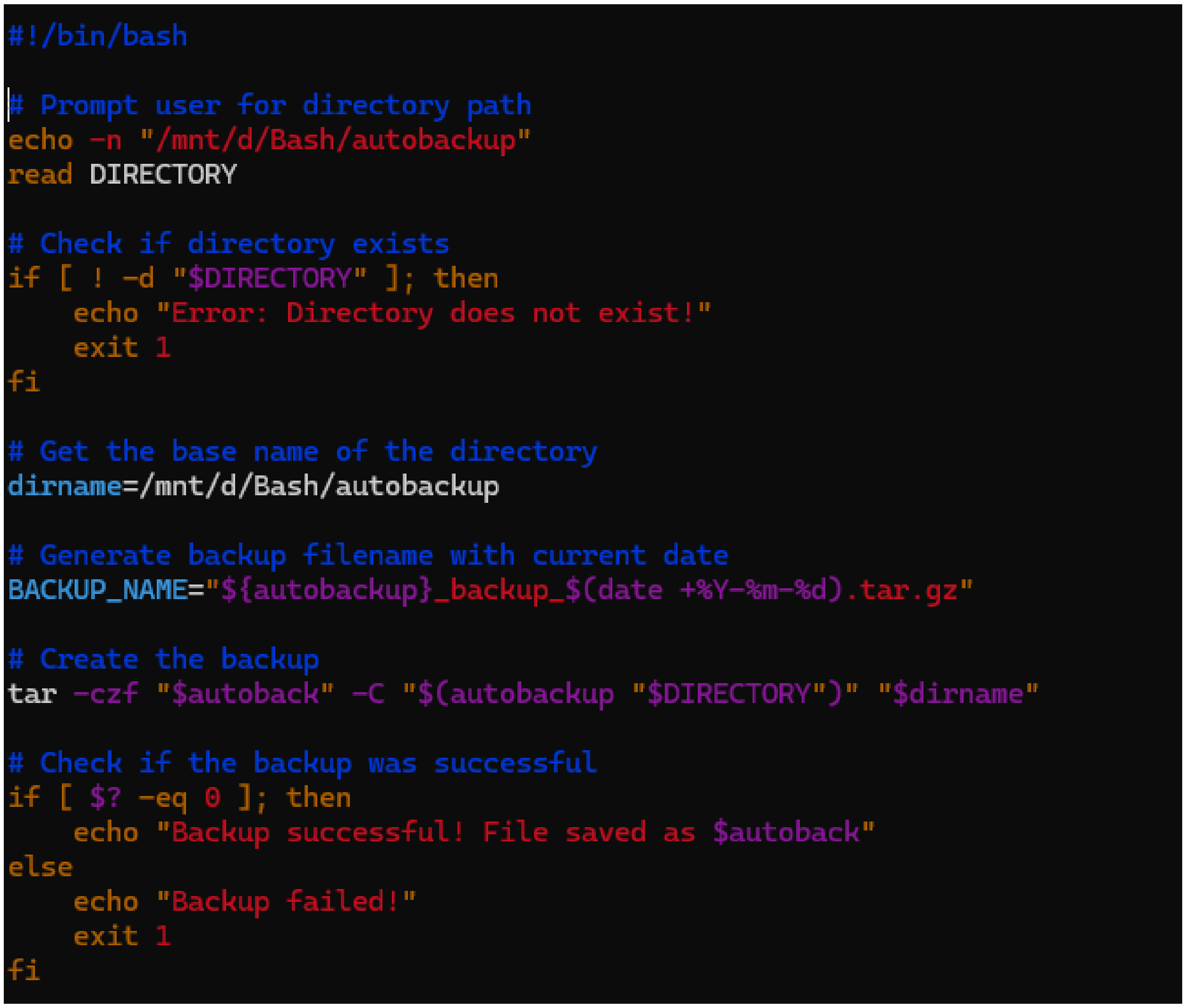
- Takes a directory path as input from the user
- Compresses the directory into a .tar.gz file
- Saves the compressed file with a name that includes the current date (e.g., backup_2023-08-20.tar.gz).

First we will create a folder and shall put some files in it



```
root@DESKTOP-GRR5G0K:/mnt/d/Bash# mkdir autobackup
root@DESKTOP-GRR5G0K:/mnt/d/Bash# cd autobackup/
root@DESKTOP-GRR5G0K:/mnt/d/Bash/autobackup# touch fil1.txt
root@DESKTOP-GRR5G0K:/mnt/d/Bash/autobackup# touch fil2.txt
root@DESKTOP-GRR5G0K:/mnt/d/Bash/autobackup#
```

Now with below script we can create backup file



```
#!/bin/bash

# Prompt user for directory path
echo -n "/mnt/d/Bash/autobackup"
read DIRECTORY

# Check if directory exists
if [ ! -d "$DIRECTORY" ]; then
    echo "Error: Directory does not exist!"
    exit 1
fi

# Get the base name of the directory
dirname=/mnt/d/Bash/autobackup

# Generate backup filename with current date
BACKUP_NAME="${autobackup}_backup_$(date +%Y-%m-%d).tar.gz"

# Create the backup
tar -czf "$autoback" -C "$(autobackup "$DIRECTORY")" "$dirname"

# Check if the backup was successful
if [ $? -eq 0 ]; then
    echo "Backup successful! File saved as $autoback"
else
    echo "Backup failed!"
    exit 1
fi
```

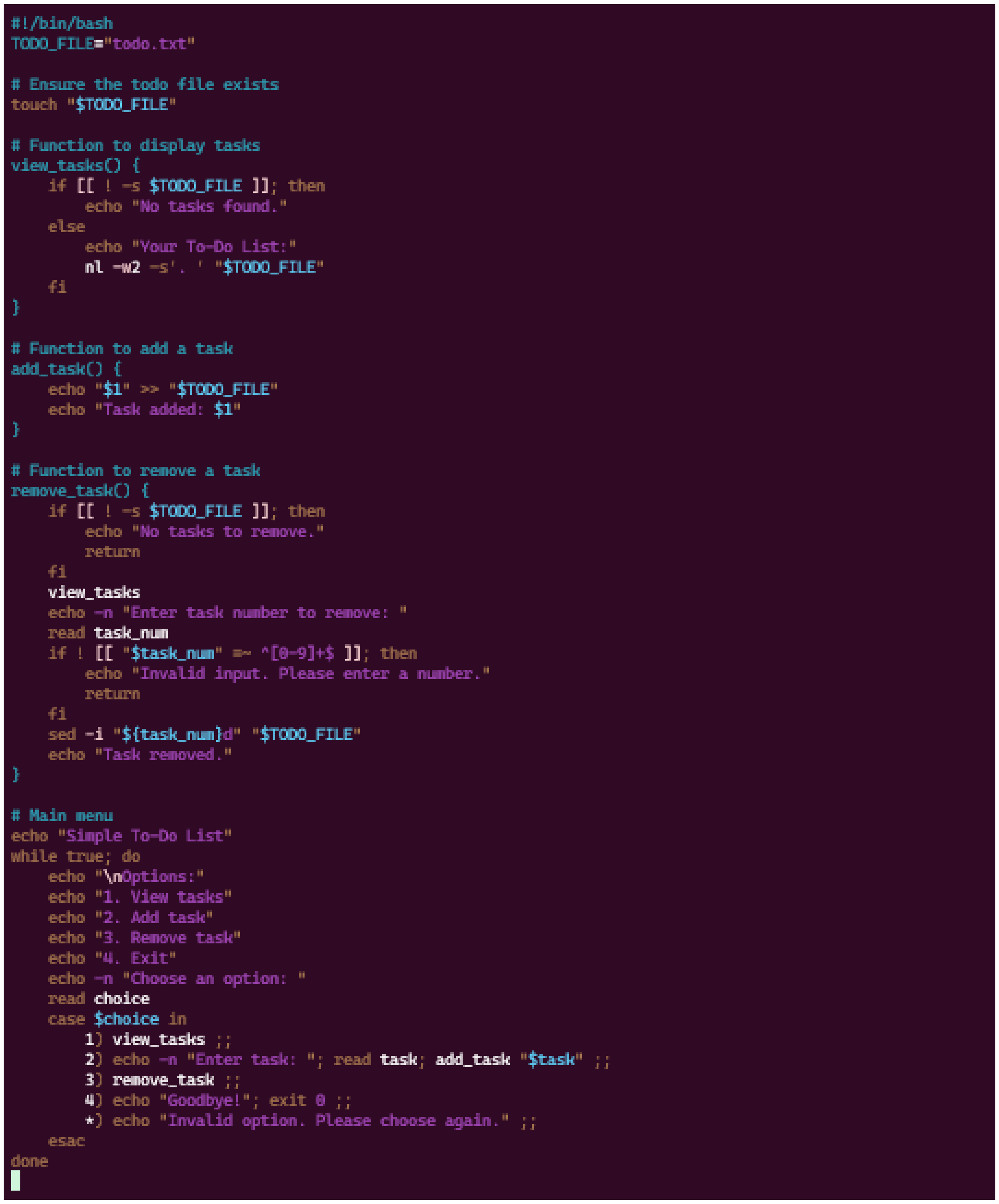
Task 5: Simple To-Do List Create a Bash script that

- Implements a simple command-line to-do list

- Allows the user to add tasks, view tasks, and remove tasks^

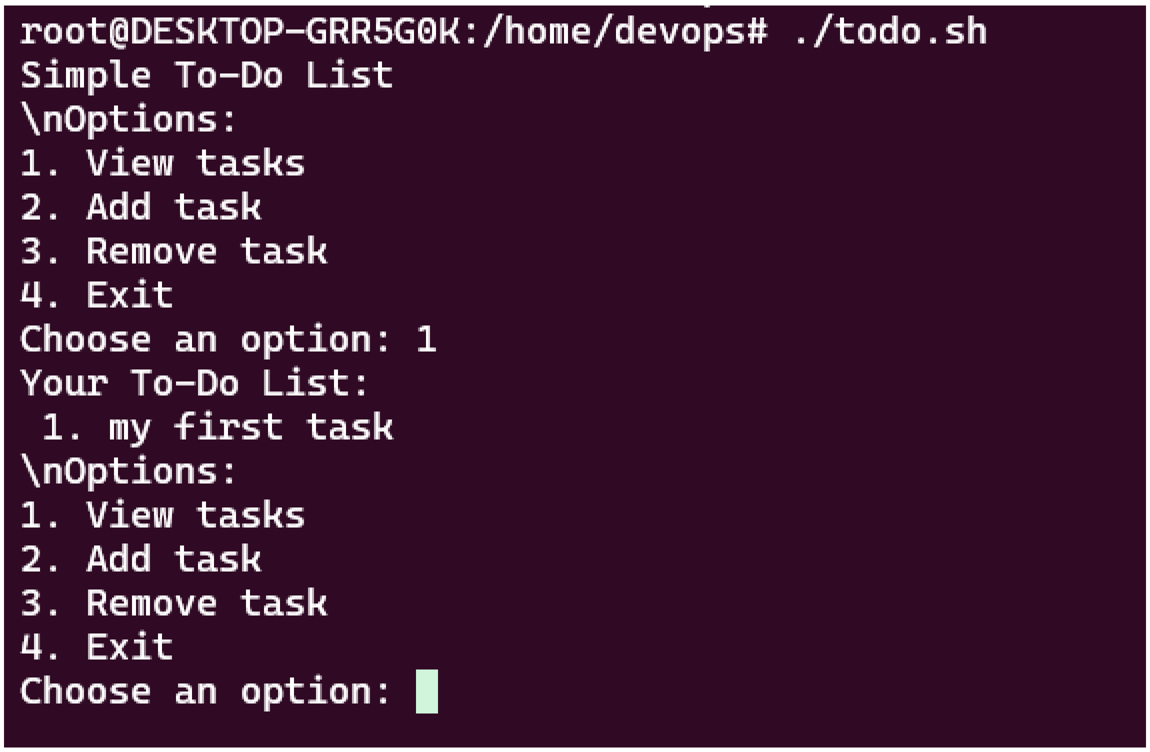
- Saves the tasks to a file (e.g., todo.txt).

For this we well use below script as shown



```
#!/bin/bash
TODO_FILE="todo.txt"

# Ensure the todo file exists
touch "$TODO_FILE"

# Function to display tasks
view_tasks() {
    if [[ ! -s $TODO_FILE ]]; then
        echo "No tasks found."
    else
        echo "Your To-Do List:"
        nl -w2 -s'. ' "$TODO_FILE"
    fi
}

# Function to add a task
add_task() {
    echo "$1" >> "$TODO_FILE"
    echo "Task added: $1"
}

# Function to remove a task
remove_task() {
    if [[ ! -s $TODO_FILE ]]; then
        echo "No tasks to remove."
        return
    fi
    view_tasks
    echo -n "Enter task number to remove: "
    read task_num
    if ! [[ "$task_num" =~ ^[0-9]+$ ]]; then
        echo "Invalid input. Please enter a number."
        return
    fi
    sed -i "${task_num}d" "$TODO_FILE"
    echo "Task removed."
}

# Main menu
echo "Simple To-Do List"
while true; do
    echo "\nOptions:"
    echo "1. View tasks"
    echo "2. Add task"
    echo "3. Remove task"
    echo "4. Exit"
    echo -n "Choose an option: "
    read choice
    case $choice in
        1) view_tasks ;;
        2) echo -n "Enter task: "; read task; add_task "$task" ;;
        3) remove_task ;;
        4) echo "Goodbye!"; exit 0 ;;
        *) echo "Invalid option. Please choose again." ;;
    esac
done
```

After then we will run the script for various functions

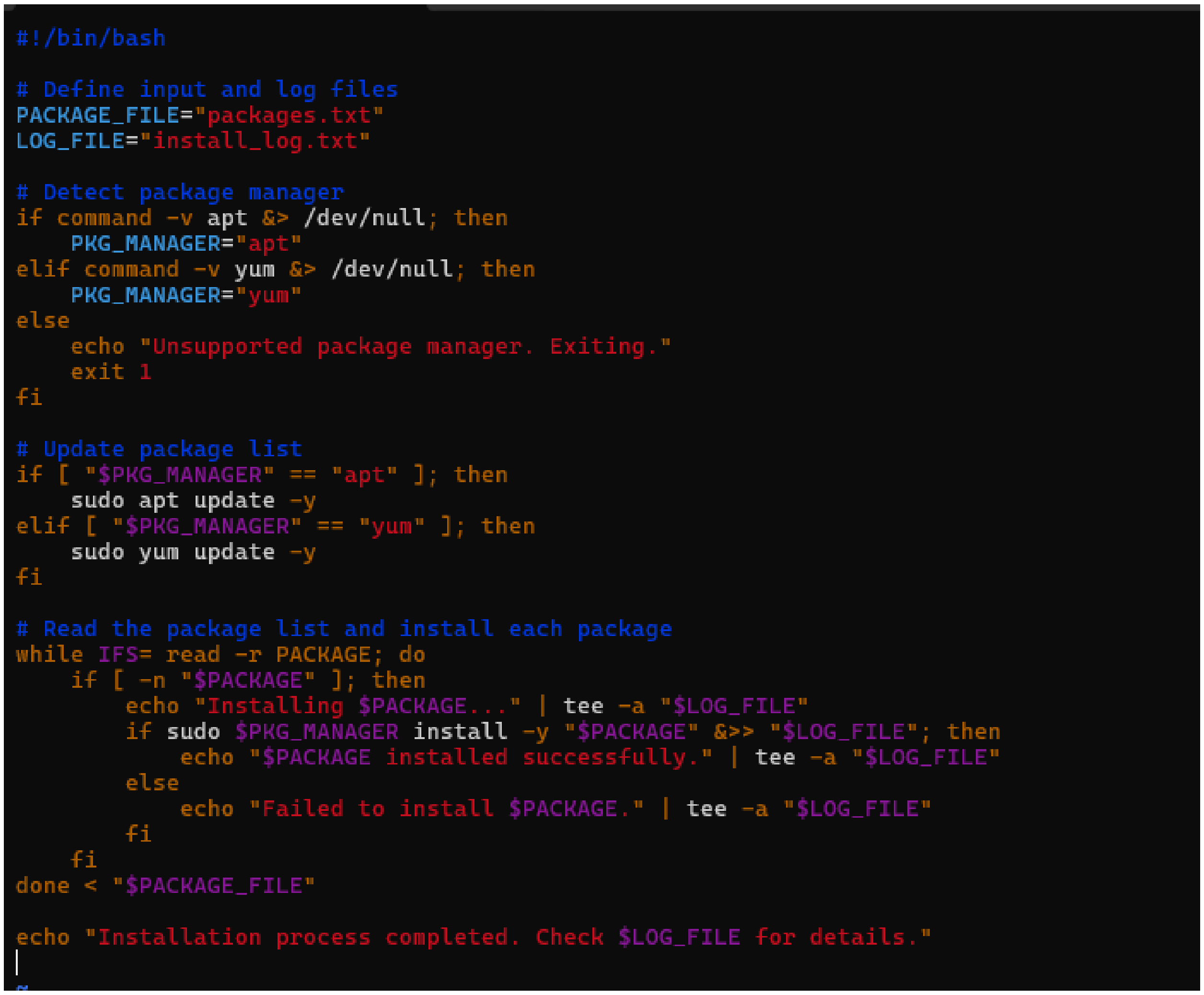


```
root@DESKTOP-GRR5G0K:/home/devops# ./todo.sh
Simple To-Do List
\nOptions:
1. View tasks
2. Add task
3. Remove task
4. Exit
Choose an option: 1
Your To-Do List:
 1. my first task
\nOptions:
1. View tasks
2. Add task
3. Remove task
4. Exit
Choose an option: 
```

Task 6: Automated Software Installation Write a script that

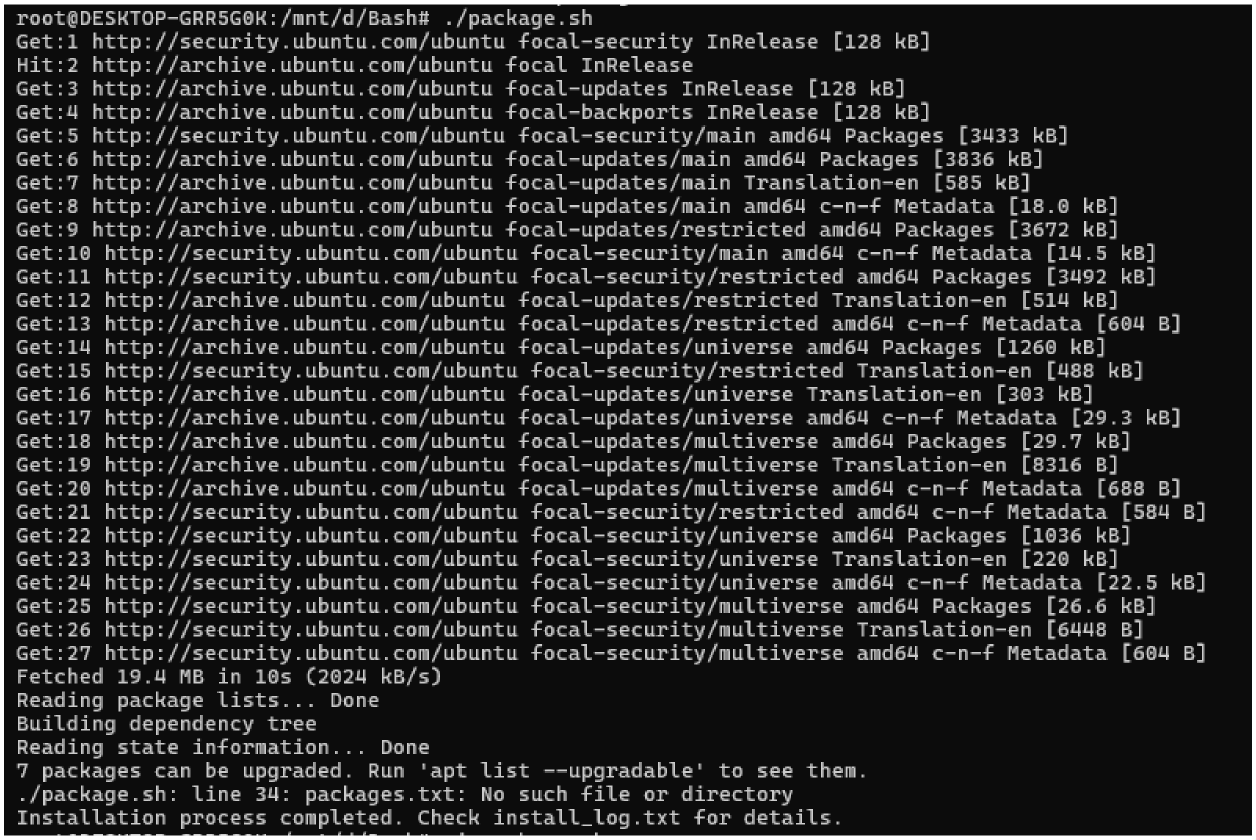
- Reads a list of software package names from a file (e.g., packages.txt)
- Installs each package using the appropriate package manager (apt, yum, etc.)
- Logs the installation status of each package

For this we will create below script for package update



```bash
#!/bin/bash

# Define input and log files
PACKAGE_FILE="packages.txt"
LOG_FILE="install_log.txt"

# Detect package manager
if command -v apt &> /dev/null; then
    PKG_MANAGER="apt"
elif command -v yum &> /dev/null; then
    PKG_MANAGER="yum"
else
    echo "Unsupported package manager. Exiting."
    exit 1
fi

# Update package list
if [ "$PKG_MANAGER" == "apt" ]; then
    sudo apt update -y
elif [ "$PKG_MANAGER" == "yum" ]; then
    sudo yum update -y
fi

# Read the package list and install each package
while IFS= read -r PACKAGE; do
    if [ -n "$PACKAGE" ]; then
        echo "Installing $PACKAGE..." | tee -a "$LOG_FILE"
        if sudo $PKG_MANAGER install -y "$PACKAGE" &>> "$LOG_FILE"; then
            echo "$PACKAGE installed successfully." | tee -a "$LOG_FILE"
        else
            echo "Failed to install $PACKAGE." | tee -a "$LOG_FILE"
        fi
    fi
done < "$PACKAGE_FILE"

echo "Installation process completed. Check $LOG_FILE for details."
```

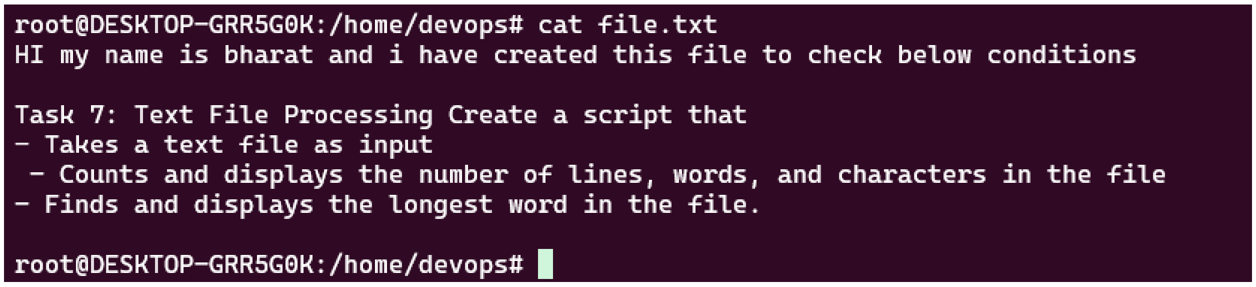
After then we can see in below image that package has been installed


```
root@DESKTOP-GRR5G0K:/mnt/d/Bash# ./package.sh
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3433 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3836 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [585 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [18.0 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [3672 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [14.5 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [3492 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [514 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [604 B]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1260 kB]
Get:15 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [488 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [303 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [29.3 kB]
Get:18 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 Packages [29.7 kB]
Get:19 http://archive.ubuntu.com/ubuntu focal-updates/multiverse Translation-en [8316 B]
Get:20 http://archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 c-n-f Metadata [688 B]
Get:21 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 c-n-f Metadata [584 B]
Get:22 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1036 kB]
Get:23 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [220 kB]
Get:24 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [22.5 kB]
Get:25 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [26.6 kB]
Get:26 http://security.ubuntu.com/ubuntu focal-security/multiverse Translation-en [6448 B]
Get:27 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 c-n-f Metadata [604 B]
Fetched 19.4 MB in 10s (2024 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
7 packages can be upgraded. Run 'apt list --upgradable' to see them.
./package.sh: line 34: packages.txt: No such file or directory
Installation process completed. Check install_log.txt for details.
```

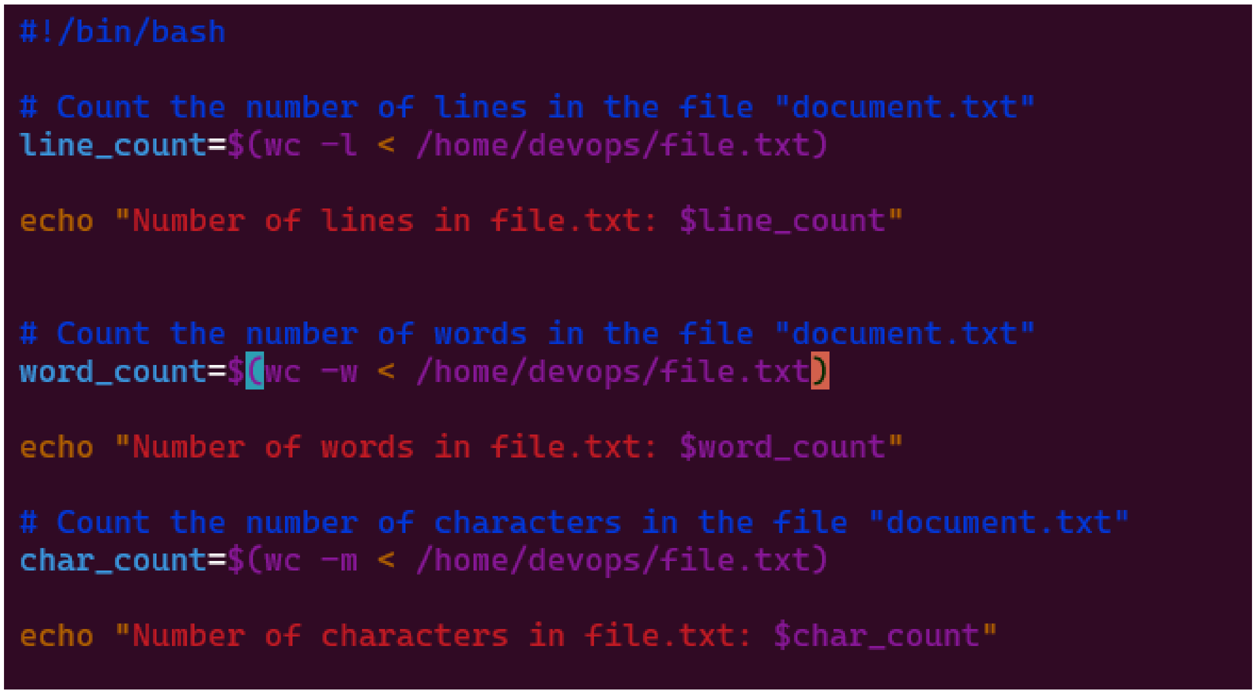
Task 7: Text File Processing Create a script that

- Takes a text file as input
- Counts and displays the number of lines, words, and characters in the file
- Finds and displays the longest word in the file.

For this we will create a file.txt in /home/devops directory

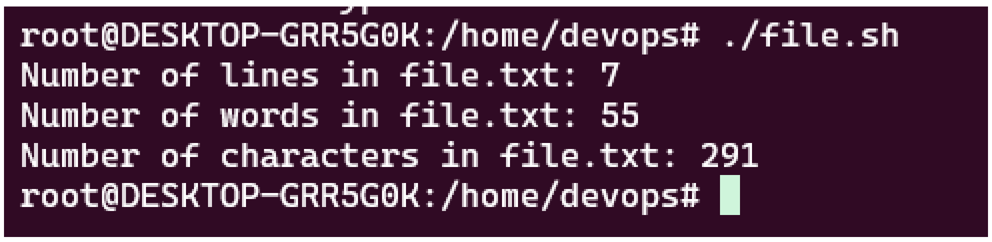

```
root@DESKTOP-GRR5G0K:/home/devops# cat file.txt
HI my name is bharat and i have created this file to check below conditions

Task 7: Text File Processing Create a script that
- Takes a text file as input
 - Counts and displays the number of lines, words, and characters in the file
- Finds and displays the longest word in the file.

root@DESKTOP-GRR5G0K:/home/devops#
```

After then we will create following script



```bash
#!/bin/bash

# Count the number of lines in the file "document.txt"
line_count=$(wc -l < /home/devops/file.txt)

echo "Number of lines in file.txt: $line_count"


# Count the number of words in the file "document.txt"
word_count=$(wc -w < /home/devops/file.txt)

echo "Number of words in file.txt: $word_count"

# Count the number of characters in the file "document.txt"
char_count=$(wc -m < /home/devops/file.txt)

echo "Number of characters in file.txt: $char_count"
```

Then we can check by running the script



```
root@DESKTOP-GRR5G0K:/home/devops# ./file.sh
Number of lines in file.txt: 7
Number of words in file.txt: 55
Number of characters in file.txt: 291
root@DESKTOP-GRR5G0K:/home/devops#
```

Thank you.