

LINUX CHALLENGE FOR DEVOPS

By: Sana thabassum

ersanathabassum06@gmail.com

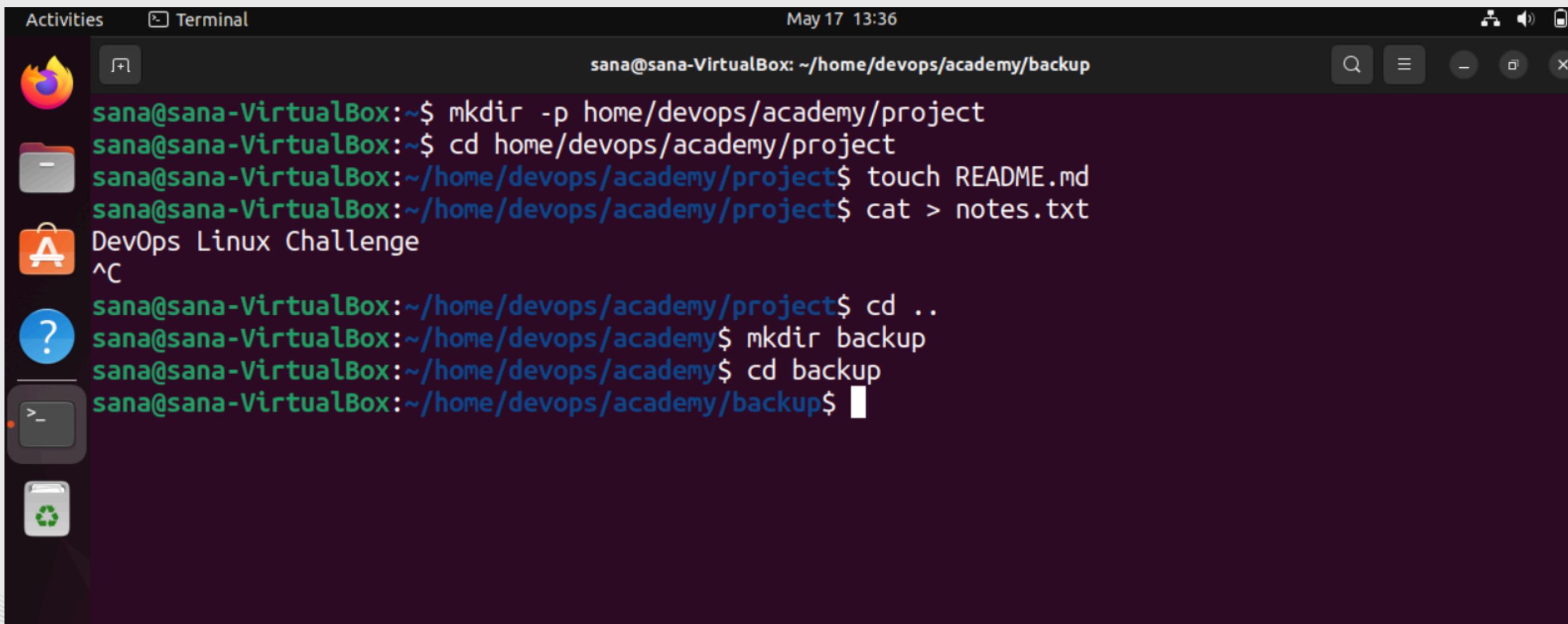
TASK 1: BASIC LINUX COMMANDS

1. Create and Navigate Directories:

- Create a directory structure follows:/home/devops/academy/project.
- Navigate into the project directory.

2. File Management:

- Create an empty file named README.md inside the project directory.
- Create another file named notes.txt and add the text "DevOps Linux Challenge" into it.

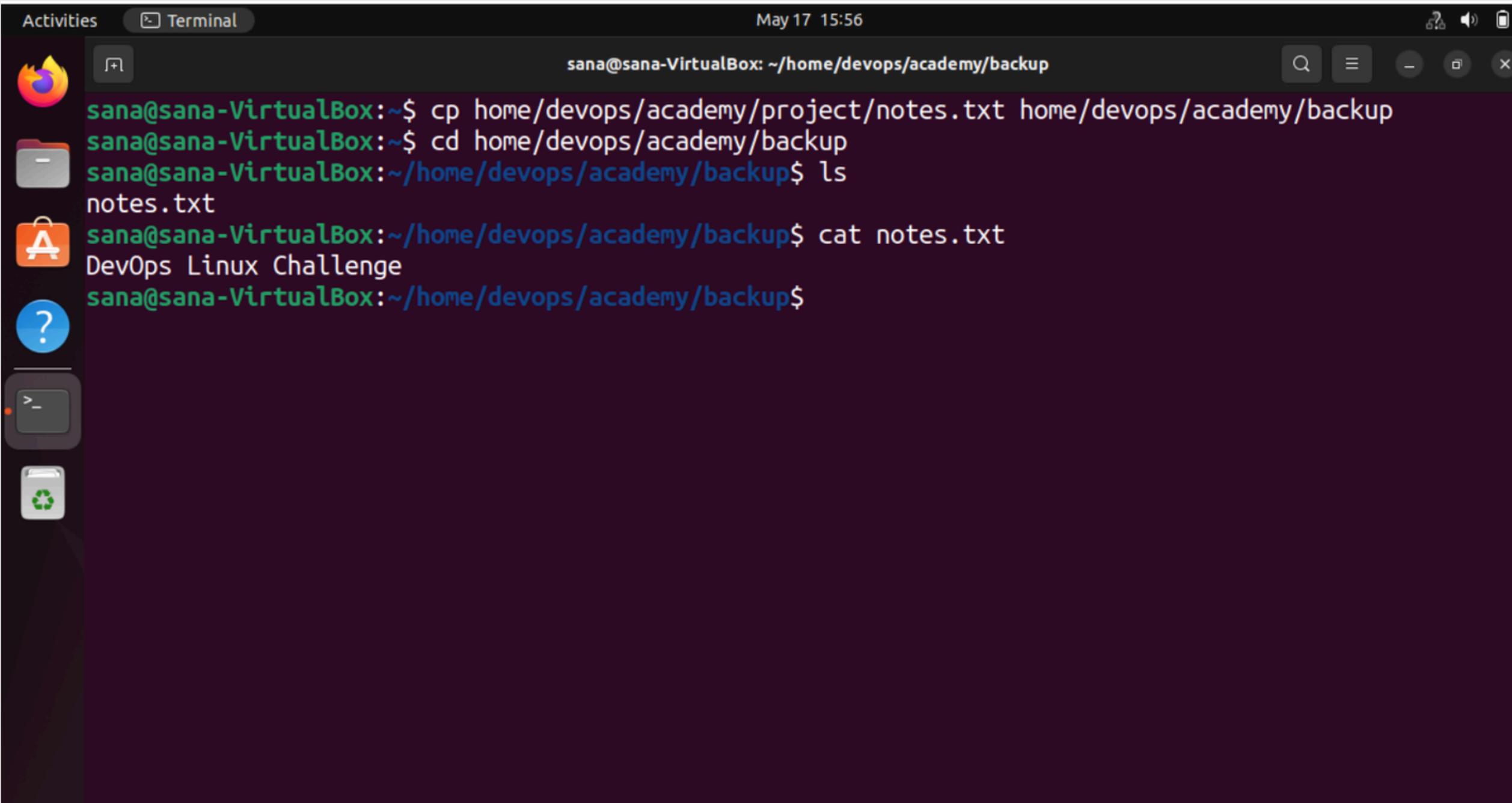


The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and the status bar shows the date and time as "May 17 13:36". The user's session is identified as "sana@sana-VirtualBox: ~". The terminal window contains the following command history:

```
sana@sana-VirtualBox:~$ mkdir -p home/devops/academy/project
sana@sana-VirtualBox:~$ cd home/devops/academy/project
sana@sana-VirtualBox:~/home/devops/academy/project$ touch README.md
sana@sana-VirtualBox:~/home/devops/academy/project$ cat > notes.txt
DevOps Linux Challenge
^C
sana@sana-VirtualBox:~/home/devops/academy/project$ cd ..
sana@sana-VirtualBox:~/home/devops/academy$ mkdir backup
sana@sana-VirtualBox:~/home/devops/academy$ cd backup
sana@sana-VirtualBox:~/home/devops/academy/backup$
```

The terminal window has a dark background with light-colored text. The left side of the screen shows a dock with various icons, including a browser, file manager, application launcher, help, and terminal.

- Copy notes.txt to /home/devops/academy/backup/.



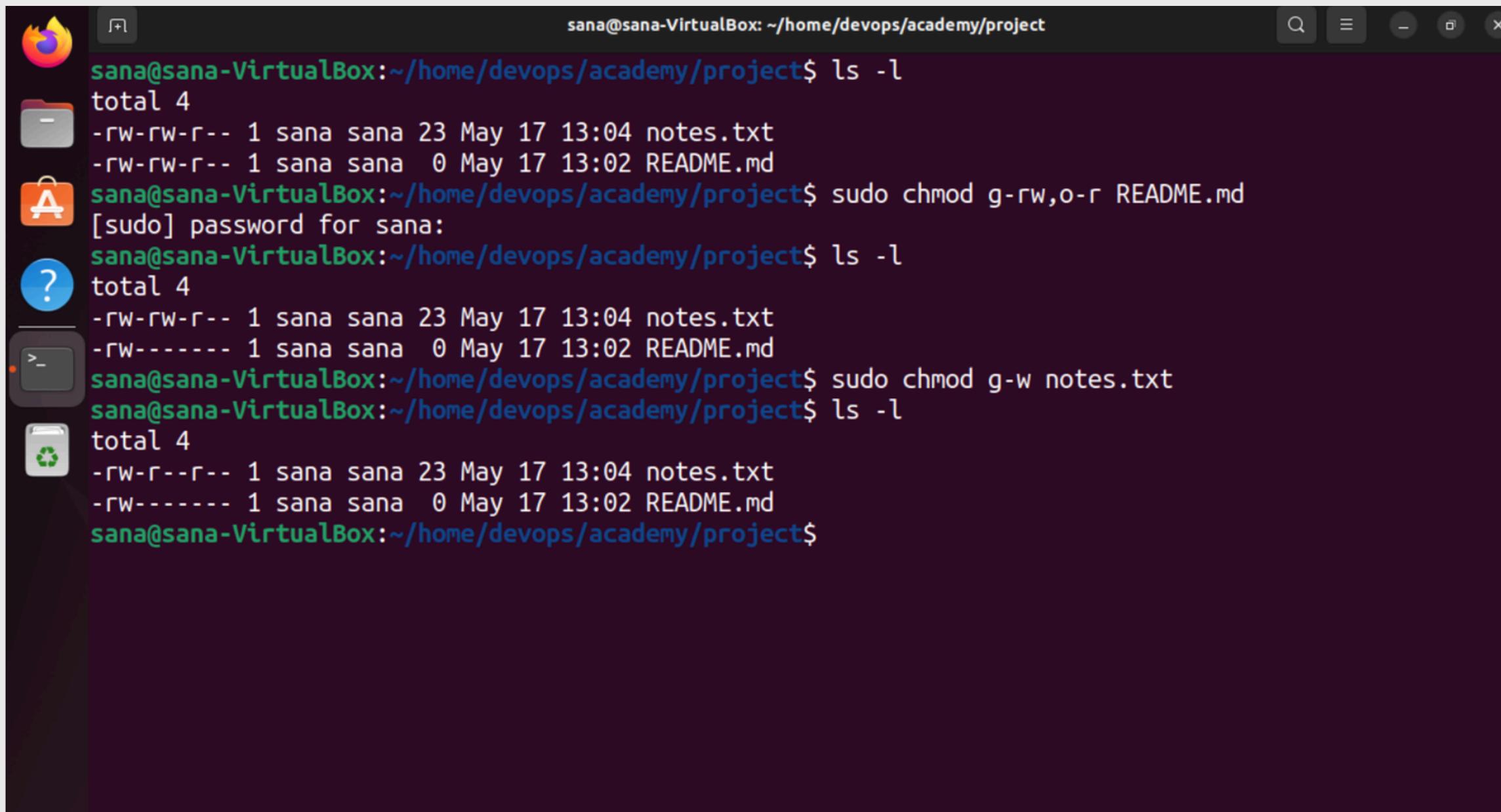
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window is titled "Terminal" and has the command line interface (CLI) visible. The CLI shows the user performing the following steps:

```
sana@sana-VirtualBox:~$ cp home/devops/academy/project/notes.txt home/devops/academy/backup
sana@sana-VirtualBox:~$ cd home/devops/academy/backup
sana@sana-VirtualBox:~/home/devops/academy/backup$ ls
notes.txt
sana@sana-VirtualBox:~/home/devops/academy/backup$ cat notes.txt
DevOps Linux Challenge
sana@sana-VirtualBox:~/home/devops/academy/backup$
```

The terminal window is part of a desktop environment with a dark theme. On the left, there is a vertical dock containing icons for various applications: a browser (Firefox), a file manager (Nautilus), a text editor (gedit), a help center (Ubuntu Help), a terminal (Terminal), and a system monitor (System Monitor). The desktop background features a subtle geometric pattern of thin, light gray lines.

3. Permissions:

- Change the permissions of README.md to be readable and writable only by the owner.
- Make notes.txt readable by everyone, but writable only by the owner.



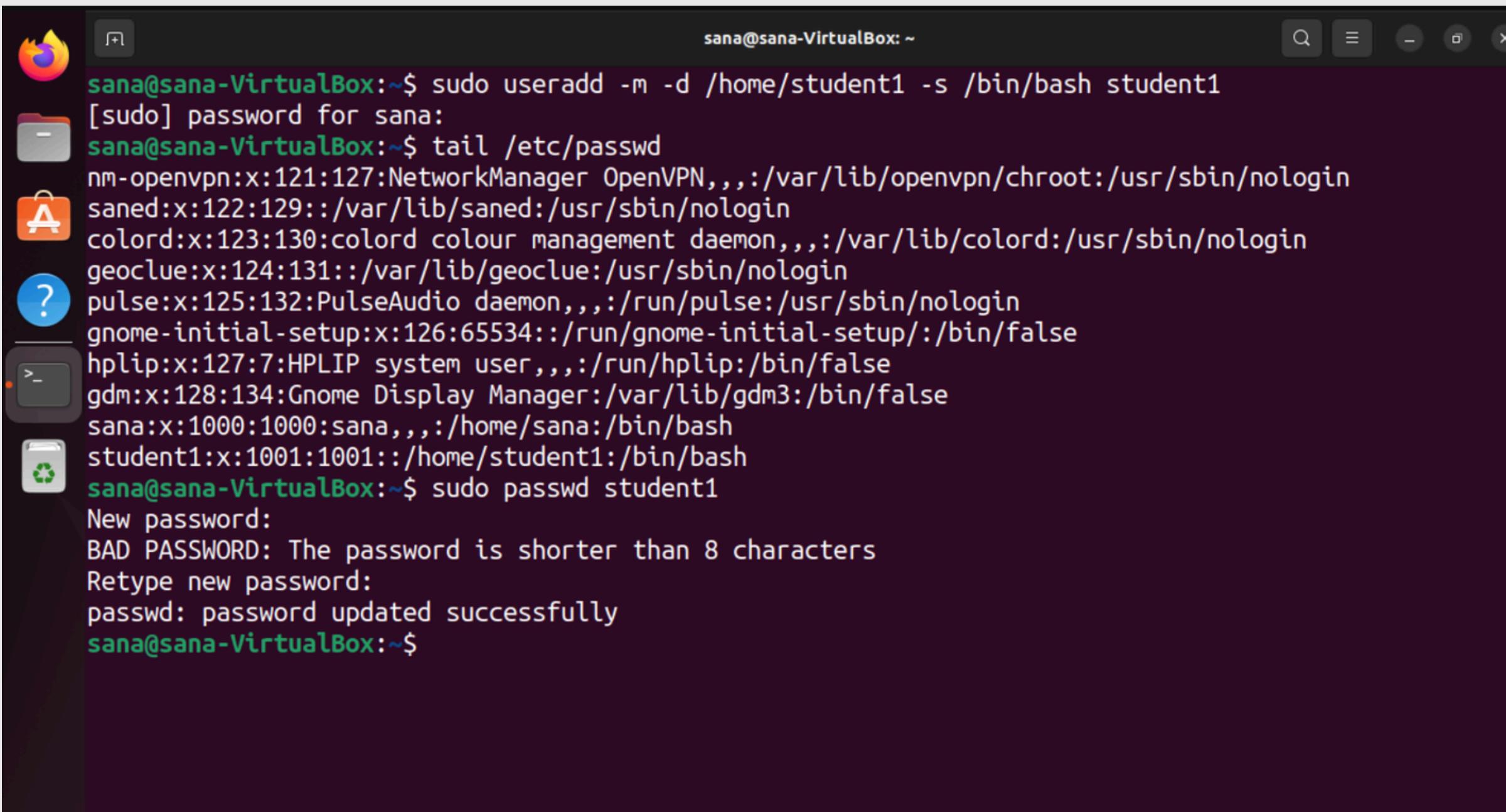
The screenshot shows a terminal window with a dark theme. The terminal title is "sana@sana-VirtualBox: ~/home/devops/academy/project". The user runs the command "ls -l" to list files, showing two files: "notes.txt" and "README.md". Both files have permissions "-rw-rw-r--" (owner readable/writable, group readable/writable, others readable). The user then runs "sudo chmod g-rw,o-r README.md" to change the permissions to "-r--r--r--" (owner readable, group writable, others readable). After entering the sudo password, they run "ls -l" again, showing the updated permissions for "README.md". Finally, they run "sudo chmod g-w notes.txt" to change the permissions of "notes.txt" to "-r--r--r--" (owner readable, group writable, others readable).

```
sana@sana-VirtualBox:~/home/devops/academy/project$ ls -l
total 4
-rw-rw-r-- 1 sana sana 23 May 17 13:04 notes.txt
-rw-rw-r-- 1 sana sana 0 May 17 13:02 README.md
sana@sana-VirtualBox:~/home/devops/academy/project$ sudo chmod g-rw,o-r README.md
[sudo] password for sana:
sana@sana-VirtualBox:~/home/devops/academy/project$ ls -l
total 4
-rw-rw-r-- 1 sana sana 23 May 17 13:04 notes.txt
-rw----- 1 sana sana 0 May 17 13:02 README.md
sana@sana-VirtualBox:~/home/devops/academy/project$ sudo chmod g-w notes.txt
sana@sana-VirtualBox:~/home/devops/academy/project$ ls -l
total 4
-rw-r--r-- 1 sana sana 23 May 17 13:04 notes.txt
-rw----- 1 sana sana 0 May 17 13:02 README.md
sana@sana-VirtualBox:~/home/devops/academy/project$
```

TASK 2: USER AND GROUP MANAGEMENT

1. Create Users:

- Create a new user named student1.
- Set a password for student1.

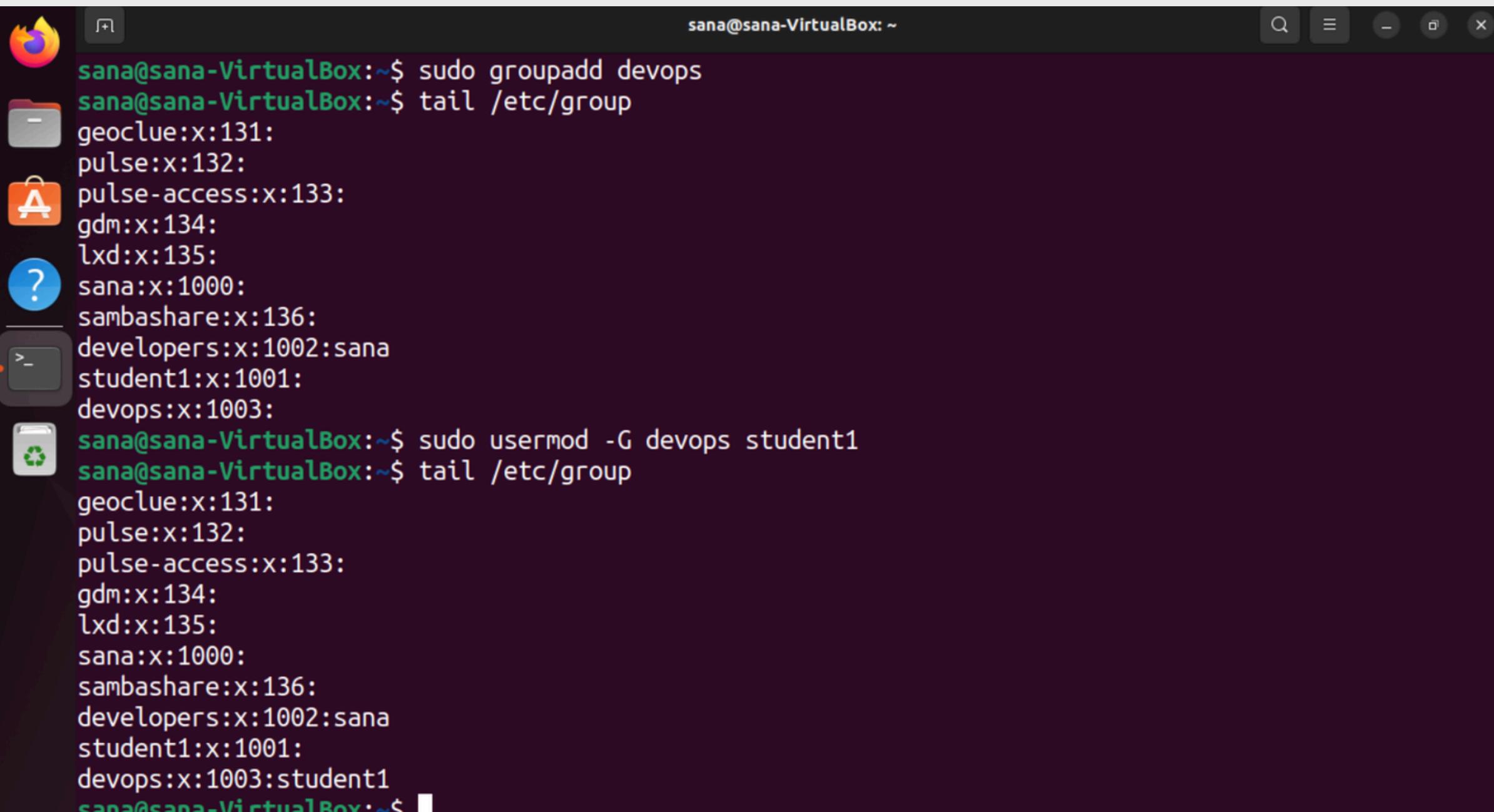


The screenshot shows a terminal window with a dark background and light-colored text. The window title is "sana@sana-VirtualBox: ~". The terminal displays the following commands and output:

```
sana@sana-VirtualBox:~$ sudo useradd -m -d /home/student1 -s /bin/bash student1
[sudo] password for sana:
sana@sana-VirtualBox:~$ tail /etc/passwd
nm-openvpn:x:121:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
saned:x:122:129::/var/lib/saned:/usr/sbin/nologin
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
sana:x:1000:1000:sana,,,:/home/sana:/bin/bash
student1:x:1001:1001::/home/student1:/bin/bash
sana@sana-VirtualBox:~$ sudo passwd student1
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
sana@sana-VirtualBox:~$
```

2. Groups:

- Create a new group named devops.
- Add student1 to the devops group.

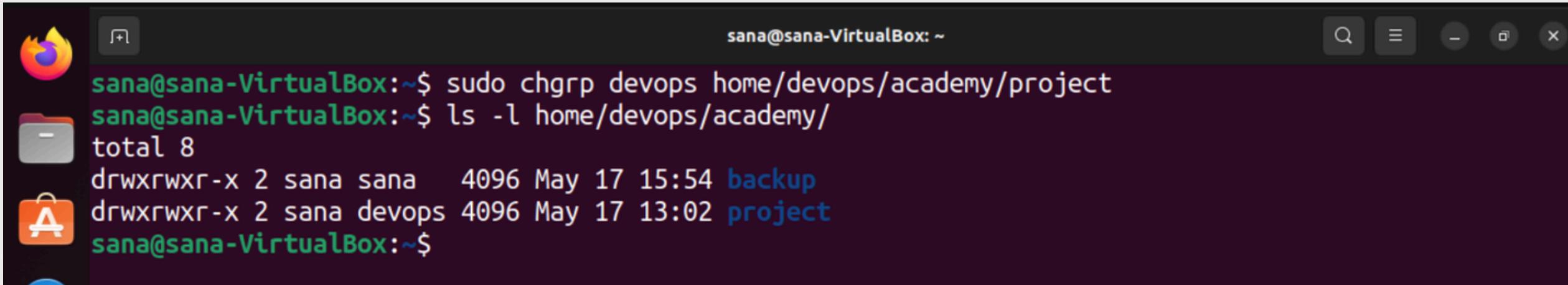


The screenshot shows a terminal window with a dark background and light-colored text. The terminal title is "sana@sana-VirtualBox: ~". The user runs two commands: "sudo groupadd devops" and "tail /etc/group". The output shows the creation of the "devops" group and its addition to the "/etc/group" file. The terminal window has a standard Linux desktop interface with icons for a browser, file manager, terminal, and help.

```
sana@sana-VirtualBox:~$ sudo groupadd devops
sana@sana-VirtualBox:~$ tail /etc/group
geoclue:x:131:
pulse:x:132:
pulse-access:x:133:
gdm:x:134:
lxde:x:135:
sana:x:1000:
sambashare:x:136:
developers:x:1002:sana
student1:x:1001:
devops:x:1003:
sana@sana-VirtualBox:~$ sudo usermod -G devops student1
sana@sana-VirtualBox:~$ tail /etc/group
geoclue:x:131:
pulse:x:132:
pulse-access:x:133:
gdm:x:134:
lxde:x:135:
sana:x:1000:
sambashare:x:136:
developers:x:1002:sana
student1:x:1001:
devops:x:1003:student1
sana@sana-VirtualBox:~$
```

3. User Permissions:

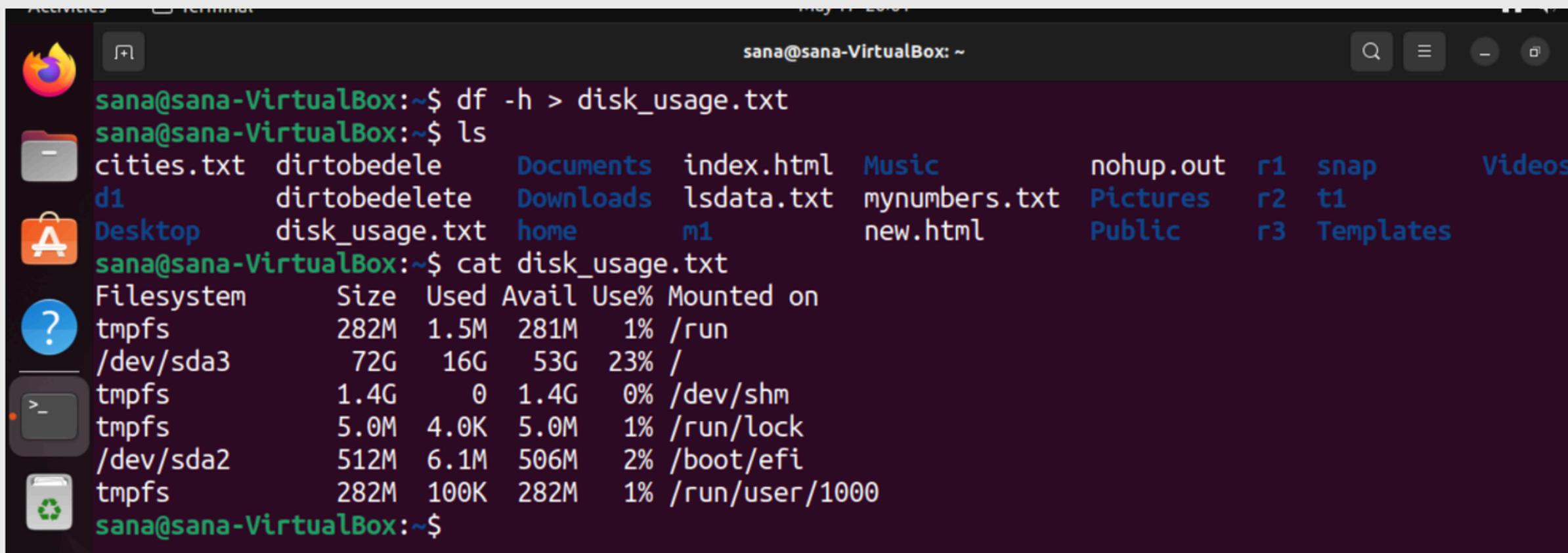
- Ensure that the project directory is accessible to members of the devops group.



A screenshot of a Linux terminal window titled "sana@sana-VirtualBox: ~". The terminal shows two commands being run:
1. `sudo chgrp devops home/devops/academy/project`
2. `ls -l home/devops/academy/`
The output of the second command shows a directory named "project" with permissions `drwxrwxr-x` and ownership `sana:devops`. The "project" directory contains files "backup" and "project".

4. Disk Usage:

- Display disk usage in human-readable format for /home directory and save the output to a file named disk_usage.txt.



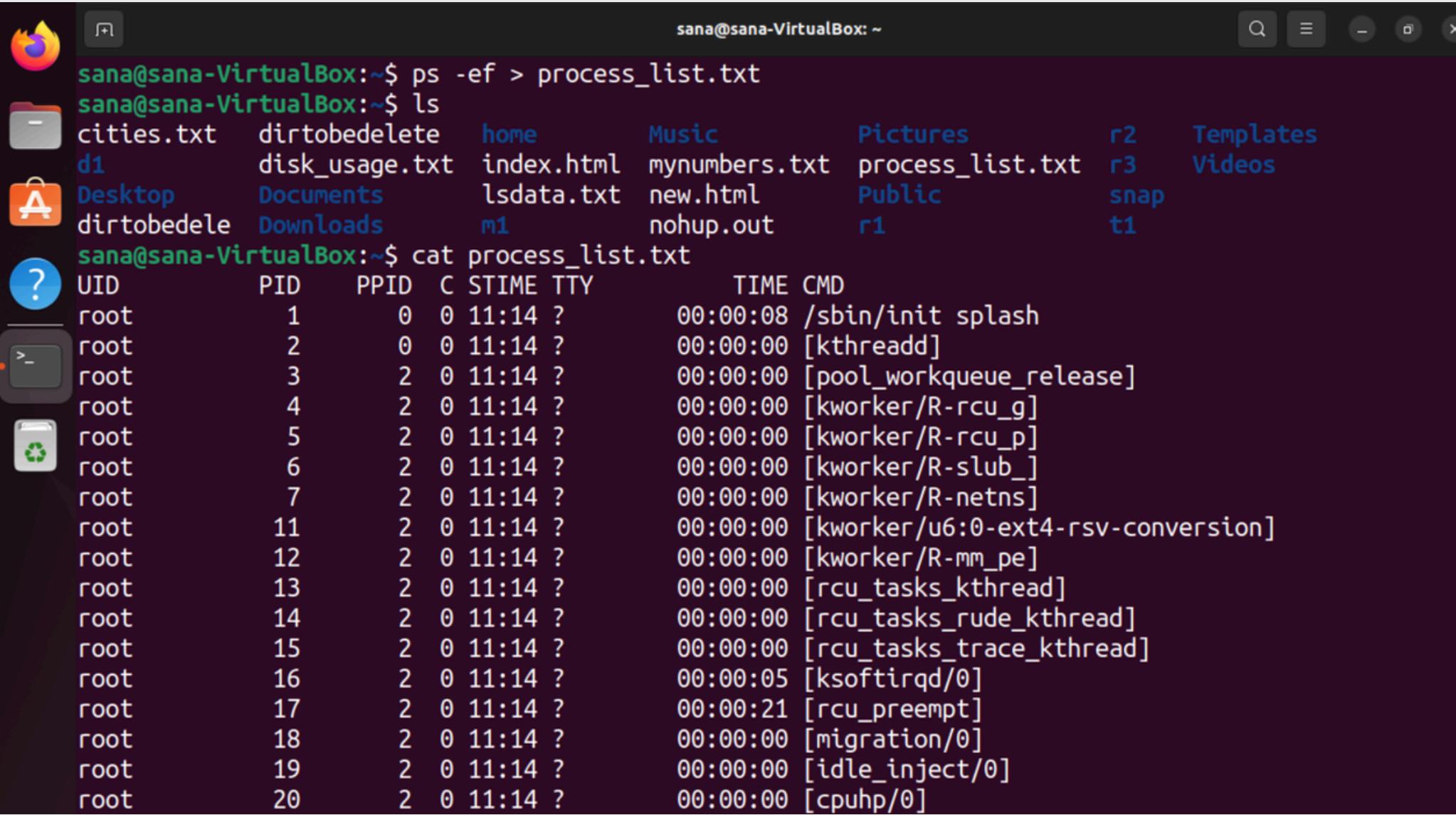
A screenshot of a Linux terminal window titled "sana@sana-VirtualBox: ~". The terminal shows three commands:
1. `df -h > disk_usage.txt`
2. `ls` (listing files in the current directory)
3. `cat disk_usage.txt`
The "ls" command output shows various files and directories in the home directory. The "cat" command output shows a table of disk usage for different filesystems:

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	282M	1.5M	281M	1%	/run
/dev/sda3	72G	16G	53G	23%	/
tmpfs	1.4G	0	1.4G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
/dev/sda2	512M	6.1M	506M	2%	/boot/efi
tmpfs	282M	100K	282M	1%	/run/user/1000

TASK 3: PROCESS MANAGEMENT

1. List Processes:

- Display all running processes and redirect the output to a file named process_list.txt.



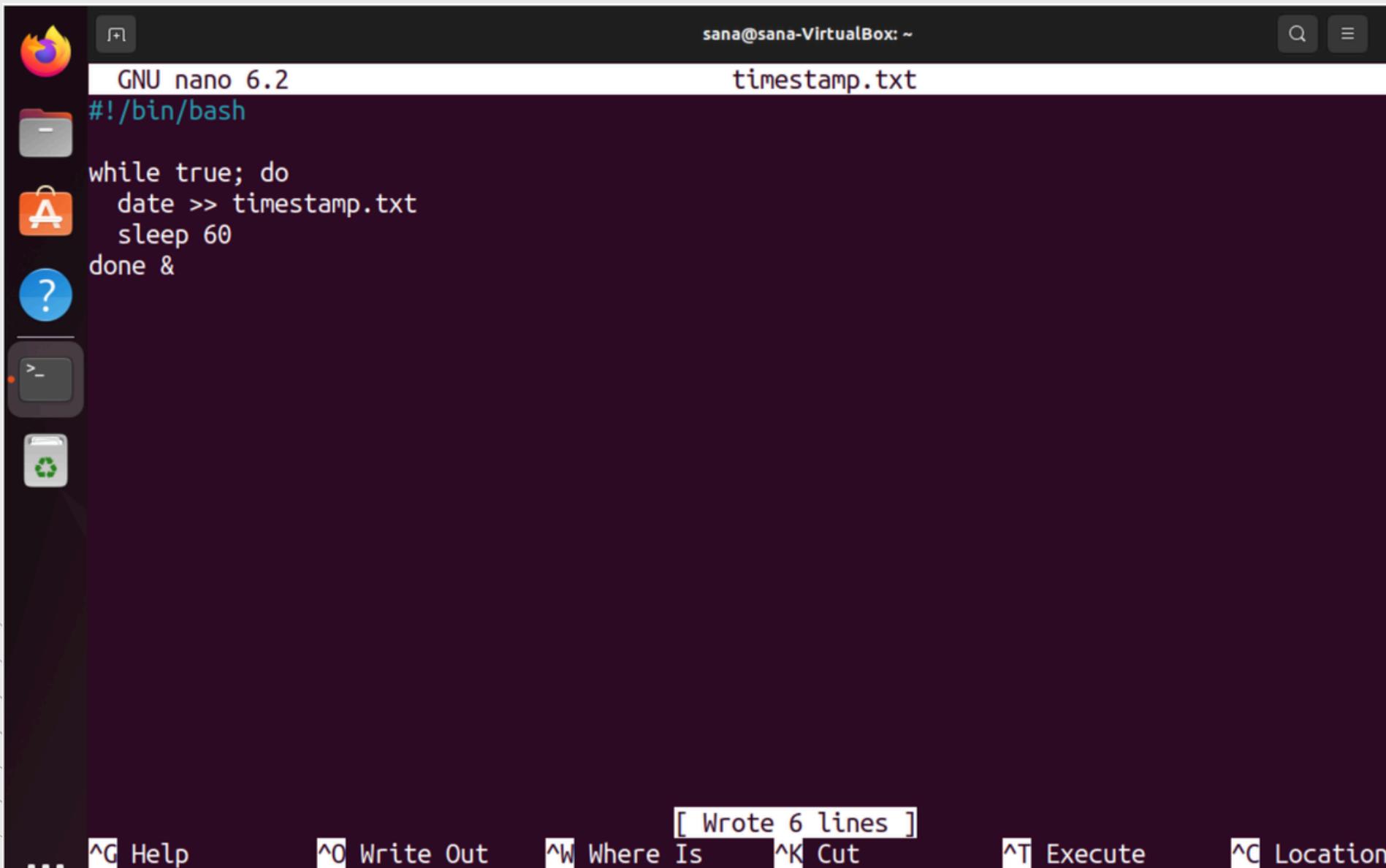
The screenshot shows a terminal window with a dark theme. The terminal title is "sana@sana-VirtualBox:~". The user runs the command "ps -ef > process_list.txt" to list all processes and then "ls" to show the contents of the current directory. The directory contains several files and folders: cities.txt, dirtobedelete, home, Music, Pictures, r2, Templates, d1, disk_usage.txt, index.html, mynumbers.txt, process_list.txt, r3, Videos, Desktop, Documents, lsdata.txt, new.html, Public, snap, dirtobedele, Downloads, m1, nohup.out, r1, t1. Finally, the user runs "cat process_list.txt" to view the contents of the file, which lists the system's processes with columns for UID, PID, PPID, C, STIME, TTY, TIME, and CMD.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	11:14	?	00:00:08	/sbin/init splash
root	2	0	0	11:14	?	00:00:00	[kthreadd]
root	3	2	0	11:14	?	00:00:00	[pool_workqueue_release]
root	4	2	0	11:14	?	00:00:00	[kworker/R-rcu_g]
root	5	2	0	11:14	?	00:00:00	[kworker/R-rcu_p]
root	6	2	0	11:14	?	00:00:00	[kworker/R-slub_]
root	7	2	0	11:14	?	00:00:00	[kworker/R-netns]
root	11	2	0	11:14	?	00:00:00	[kworker/u6:0-ext4-rsv-conversion]
root	12	2	0	11:14	?	00:00:00	[kworker/R-mm_pe]
root	13	2	0	11:14	?	00:00:00	[rcu_tasks_kthread]
root	14	2	0	11:14	?	00:00:00	[rcu_tasks_rude_kthread]
root	15	2	0	11:14	?	00:00:00	[rcu_tasks_trace_kthread]
root	16	2	0	11:14	?	00:00:05	[ksoftirqd/0]
root	17	2	0	11:14	?	00:00:21	[rcu_preempt]
root	18	2	0	11:14	?	00:00:00	[migration/0]
root	19	2	0	11:14	?	00:00:00	[idle_inject/0]
root	20	2	0	11:14	?	00:00:00	[cpuhp/0]

```
root      3380  2  0 18:37 ?  00:00:00 [kworker/1:2H-kblockd]
systemd+  3535  1  0 19:05 ?  00:00:04 /lib/systemd/systemd-resolved
root      3537  2  0 19:05 ?  00:00:05 [kworker/0:1-mm_percpu_wq]
root      3546  1  0 19:05 ?  00:00:03 /usr/lib/snapd/snapd
root      3616  2  0 19:26 ?  00:00:00 [kworker/u8:1-events_power_efficient]
root      3618  2  0 19:27 ?  00:00:02 [kworker/2:0-mm_percpu_wq]
root      3620  2  0 19:27 ?  00:00:01 [kworker/u7:0-events_power_efficient]
root      3716  2  0 19:34 ?  00:00:01 [kworker/1:1-mm_percpu_wq]
root      4333  1  0 19:38 ?  00:00:01 /usr/libexec/fwupd/fwupd
root      4341  2  0 19:38 ?  00:00:00 [kworker/u8:2-events_unbound]
root      4373  2  0 19:50 ?  00:00:00 [kworker/u7:2-events_unbound]
root      4513  2  0 20:31 ?  00:00:00 [kworker/0:2H-kblockd]
root      4517  2  0 20:49 ?  00:00:00 [kworker/u7:3-events_power_efficient]
root      4518  2  0 20:49 ?  00:00:00 [kworker/u9:1-events_power_efficient]
root      4519  2  0 20:52 ?  00:00:00 [kworker/2:1-cgroup_destroy]
root      4523  2  0 20:52 ?  00:00:00 [kworker/1:2-cgroup_destroy]
root      4524  1  0 20:52 ?  00:00:00 /lib/systemd/systemd-udevd
root      4527  2  0 21:28 ?  00:00:00 [kworker/1:1H-kblockd]
root      4528  2  0 21:28 ?  00:00:00 [kworker/2:1H-kblockd]
root      4530  1  0 21:28 ?  00:00:00 /lib/systemd/systemd-journald
root      4536  2  0 21:31 ?  00:00:00 [kworker/u8:0-events_unbound]
sana     4559  1403 0 21:32 ?  00:00:00 gjs /usr/share/gnome-shell/extensions/ding@rastersoft.com/ding.js -E -P /usr/share/gnome-shell/extensions/ding@rastersoft.com -M 0 -D 0:0:1280:800 :1:27:0:70:0:0
root      4601      2  0 21:32 ?  00:00:00 [kworker/1:3-cgroup_destroy]
root      4604      2  0 21:33 ?  00:00:00 [kworker/u9:3-events_unbound]
sana     4605  2033 0 21:36 pts/0  00:00:00 ps -ef
sana@sana-VirtualBox:~$
```

2. Background Process:

- Start a simple background process that writes the current date and time to a file named timestamp.txt every minute using a while loop and sleep.



GNU nano 6.2

sana@sana-VirtualBox: ~

timestamp.txt

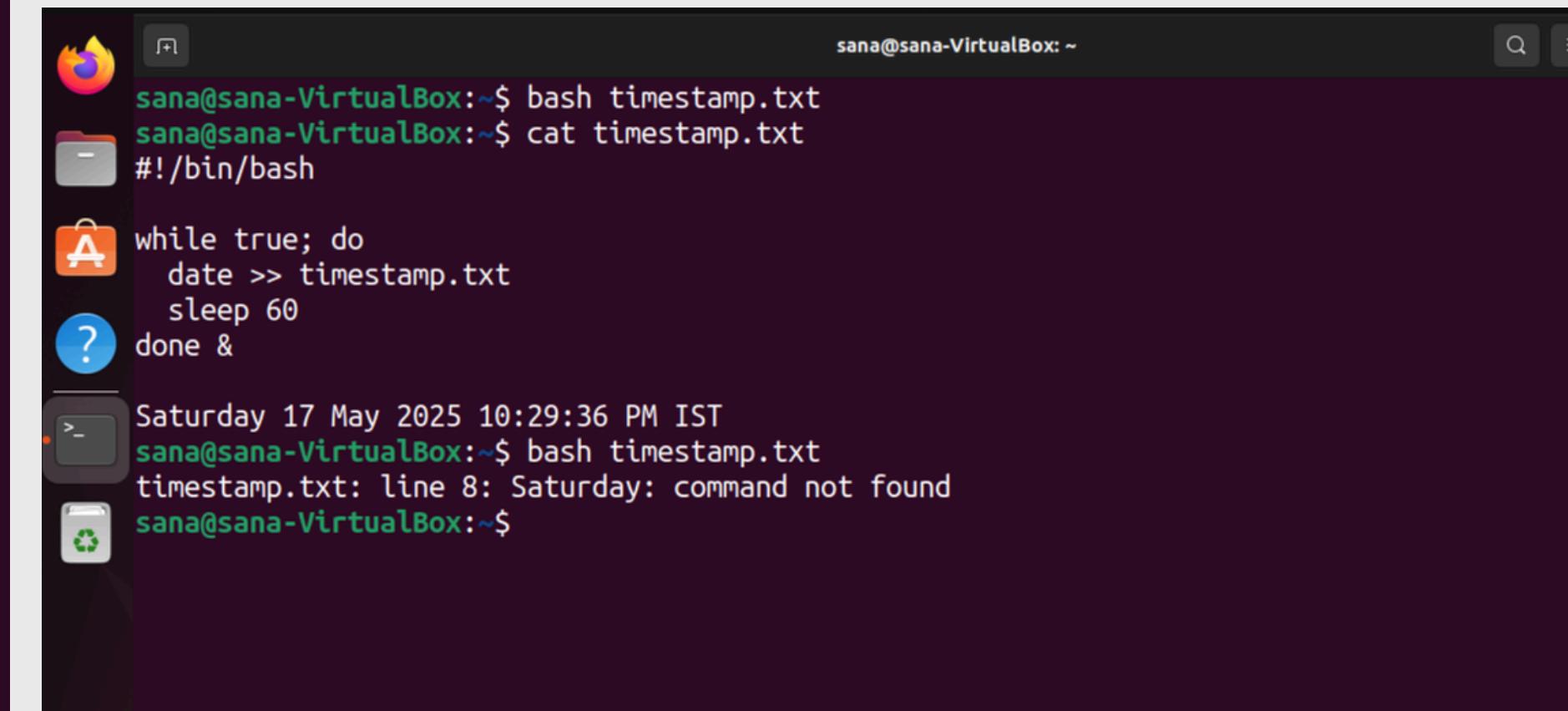
```
#!/bin/bash

while true; do
    date >> timestamp.txt
    sleep 60
done &
```

[Wrote 6 lines]

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location

This screenshot shows a terminal window with the nano text editor open. The file is named 'timestamp.txt'. The script contains a while loop that runs indefinitely, writing the current date and time to the file 'timestamp.txt' and then sleeping for 60 seconds. The status bar at the bottom indicates that 6 lines were written.



sana@sana-VirtualBox:~\$ bash timestamp.txt

sana@sana-VirtualBox:~\$ cat timestamp.txt

```
#!/bin/bash

while true; do
    date >> timestamp.txt
    sleep 60
done &
```

Saturday 17 May 2025 10:29:36 PM IST

sana@sana-VirtualBox:~\$ bash timestamp.txt

timestamp.txt: line 8: Saturday: command not found

sana@sana-VirtualBox:~\$

This screenshot shows a terminal window with the command 'bash timestamp.txt' run. The output shows the script being executed and the current date and time being printed to the file. However, the command 'Saturday' is treated as a command name rather than a variable, resulting in an error message about a command not found.

3. Kill Process:

- Find the process ID (PID) of the background process started in the previous step and terminate it.

```
sana@sana-VirtualBox:~$ nano timestamp.txt
sana@sana-VirtualBox:~$ ps -ef
```

UID	PID	PPID	C	S	TIME	TTY	CMD
root	1	0	0	22:16	?		00:00:03 /sbin/init splash
root	2	0	0	22:16	?		00:00:00 [kthreadd]
root	3	2	0	22:16	?		00:00:00 [pool_workqueue_release]
root	4	2	0	22:16	?		00:00:00 [kworker/R-rcu_g]
root	5	2	0	22:16	?		00:00:00 [kworker/R-rcu_p]
root	6	2	0	22:16	?		00:00:00 [kworker/R-slub_]
root	7	2	0	22:16	?		00:00:00 [kworker/R-netns]
root	8	2	0	22:16	?		00:00:00 [kworker/0:0-ata_sff]
root	11	2	0	22:16	?		00:00:00 [kworker/u6:0-ext4-rsv-conversion]
root	12	2	0	22:16	?		00:00:00 [kworker/R-mm_pe]
root	13	2	0	22:16	?		00:00:00 [rcu_tasks_kthread]
root	14	2	0	22:16	?		00:00:00 [rcu_tasks_rude_kthread]
root	15	2	0	22:16	?		00:00:00 [rcu_tasks_trace_kthread]
root	16	2	0	22:16	?		00:00:00 [ksoftirqd/0]
root	17	2	0	22:16	?		00:00:01 [rcu_preempt]
root	18	2	0	22:16	?		00:00:00 [migration/0]
root	19	2	0	22:16	?		00:00:00 [idle_inject/0]
root	20	2	0	22:16	?		00:00:00 [cpuhp/0]
root	21	2	0	22:16	?		00:00:00 [cpuhp/1]
root	22	2	0	22:16	?		00:00:00 [idle_inject/1]
root	23	2	0	22:16	?		00:00:00 [migration/1]
root	24	2	0	22:16	?		00:00:00 [ksoftirqd/1]

```
sana@sana-VirtualBox:~$ kill 2157
sana@sana-VirtualBox:~$ Terminated
```

4. CPU and Memory Usage:

- Display the current CPU and memory usage using top or htop, save a snapshot of this information to a file named cpu_mem_usage.txt.

```
sana@sana-VirtualBox:~$ top > cpu_mem_usage.txt
sana@sana-VirtualBox:~$ ls
cities.txt      dirtobedelete  index.html    new.html      r1   Templates
cpu_mem_usage.txt disk_usage.txt  lsdata.txt    nohup.out    r2   timestamp.txt
d1              Documents       m1           Pictures     r3   Videos
Desktop          Downloads      Music        process_list.txt snap
dirtobedelete   home          mynumbers.txt Public      t1
sana@sana-VirtualBox:~$ cat cpu_mem_usage.txt
```

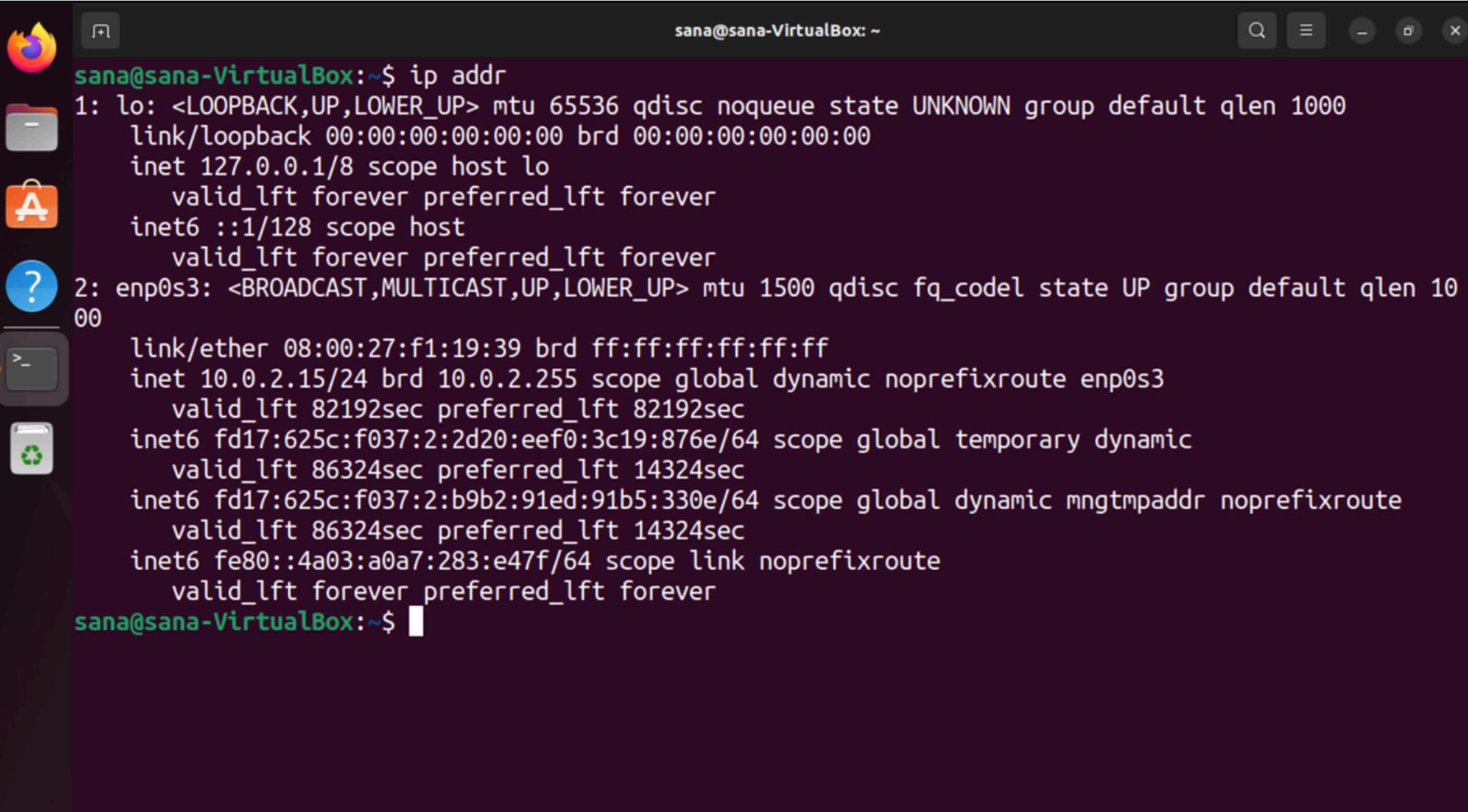
```
top - 22:04:20 up 10:49, 1 user, load average: 0.56, 0.21, 0.15
Tasks: 204 total, 1 running, 201 sleeping, 2 stopped, 0 zombie
%Cpu(s): 0.5 us, 2.0 sy, 0.9 ni, 95.5 id, 0.6 wa, 0.0 hi, 0.6 si, 0.0 st
MiB Mem : 2818.9 total, 125.4 free, 790.5 used, 1903.0 buff/cache
MiB Swap: 5134.0 total, 5134.0 free, 0.0 used. 1823.8 avail Mem

          PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
        1761 sana      39  19  721528 39380 23196 S  5.3  1.4  0:10.04 tracker-miner-f
        4945 sana      39  19  620420 32068 25796 S  2.0  1.1  0:00.34 tracker-extract
        1403 sana      20   0  4834876 393268 135656 S  1.0 13.6 14:09.74 gnome-shell
        1274 sana      20   0   17512 13568  4096 S  0.7  0.5  0:07.39 dbus-daemon
        187 root       20   0      0      0      0 S  0.3  0.0  0:04.54 jbd2/sda3-8
        1931 sana      20   0  555312 54004 41288 S  0.3  1.9  1:22.61 gnome-terminal-
        3716 root       20   0      0      0      0 I  0.3  0.0  0:02.57 kworker/1:1-events
        4528 root       0 -20      0      0      0 I  0.3  0.0  0:00.46 kworker/2:1H-kblockd
        4672 root       20   0      0      0      0 I  0.3  0.0  0:00.92 kworker/u8:2-events_unbound
        4943 sana      20   0  13080 3968  3200 R  0.3  0.1  0:00.11 top
          1 root       20   0 166696 11624  8296 S  0.0  0.4  0:08.44 systemd
          2 root       20   0      0      0      0 S  0.0  0.0  0:00.30 kthreadd
          3 root       20   0      0      0      0 S  0.0  0.0  0:00.00 pool_workqueue_release
          4 root       0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/R-rcu_g
          5 root       0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/R-rcu_p
          6 root       0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/R-slub_
          7 root       0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/R-netns
          11 root      20   0      0      0      0 I  0.0  0.0  0:00.53 kworker/u6:0-ext4-rsv-conv+
          12 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/R-mm_pe
          13 root      20   0      0      0      0 I  0.0  0.0  0:00.00 rcu_tasks_kthread
sana@sana-VirtualBox:~$
```

Task 4: Networking

1. Network Configuration:

- Display the current network configuration using ifconfig or ip addr.

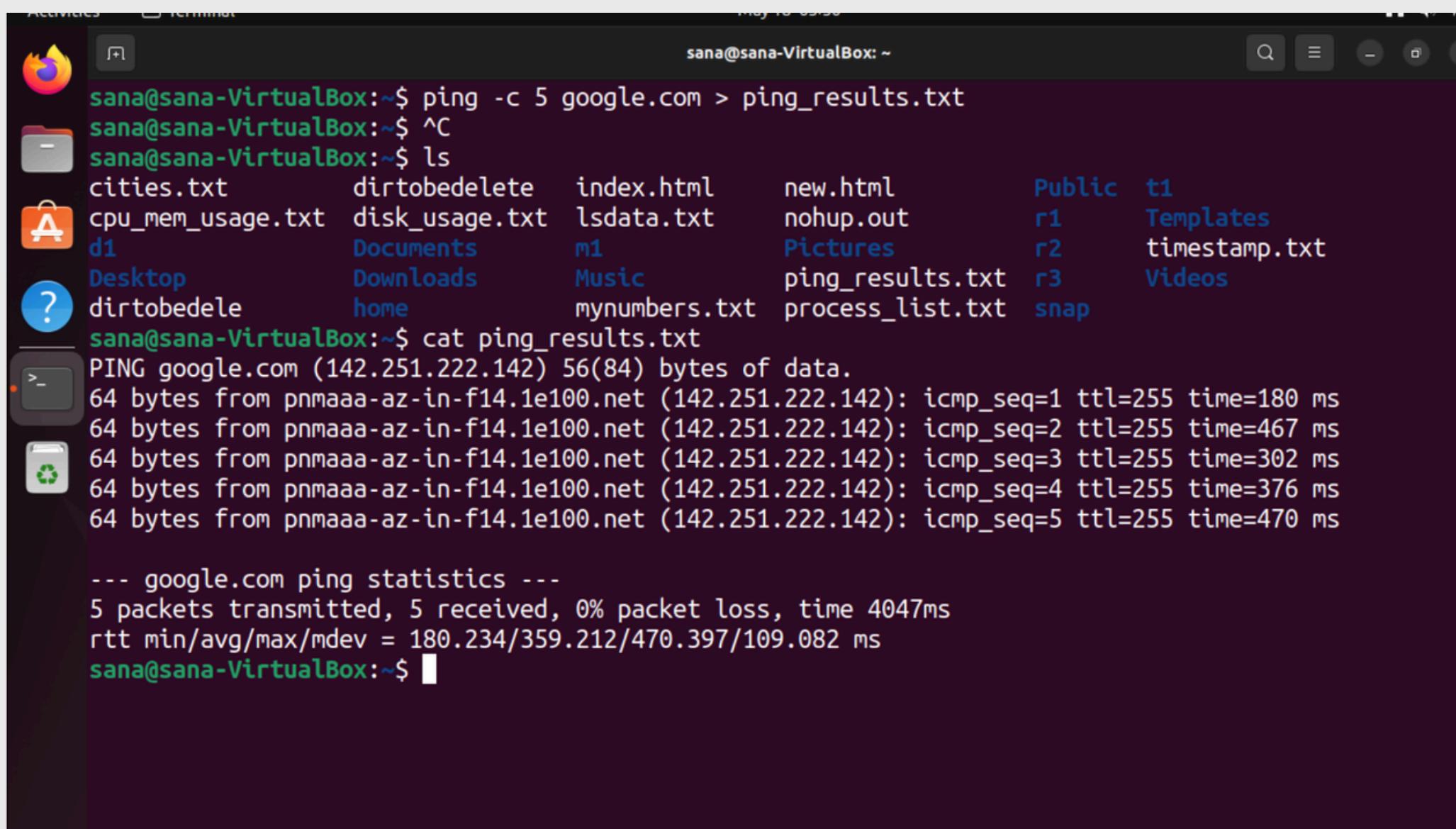


A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the output of the command 'ip addr'. The output shows two network interfaces: 'lo' (loopback) and 'enp0s3' (ethernet). The 'lo' interface has an IPv4 address of 127.0.0.1/8 and an IPv6 address of ::1/128. The 'enp0s3' interface has an IPv4 address of 10.0.2.15/24 and three IPv6 addresses: fd17:625c:f037:2:d20:eeff:fe/64, fd17:625c:f037:2:b9b2:91ed:91b5:330e/64, and fe80::4a03:a0a7:283:e47f/64. The terminal window also shows icons for various applications in the dock on the left side.

```
sana@sana-VirtualBox:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f1:19:39 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 82192sec preferred_lft 82192sec
    inet6 fd17:625c:f037:2:d20:eeff:fe/64 scope global temporary dynamic
        valid_lft 86324sec preferred_lft 14324sec
    inet6 fd17:625c:f037:2:b9b2:91ed:91b5:330e/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 86324sec preferred_lft 14324sec
    inet6 fe80::4a03:a0a7:283:e47f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
sana@sana-VirtualBox:~$
```

2. Ping Test:

- Ping google.com and save the output to a file named ping_results.txt.



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following command and its output:

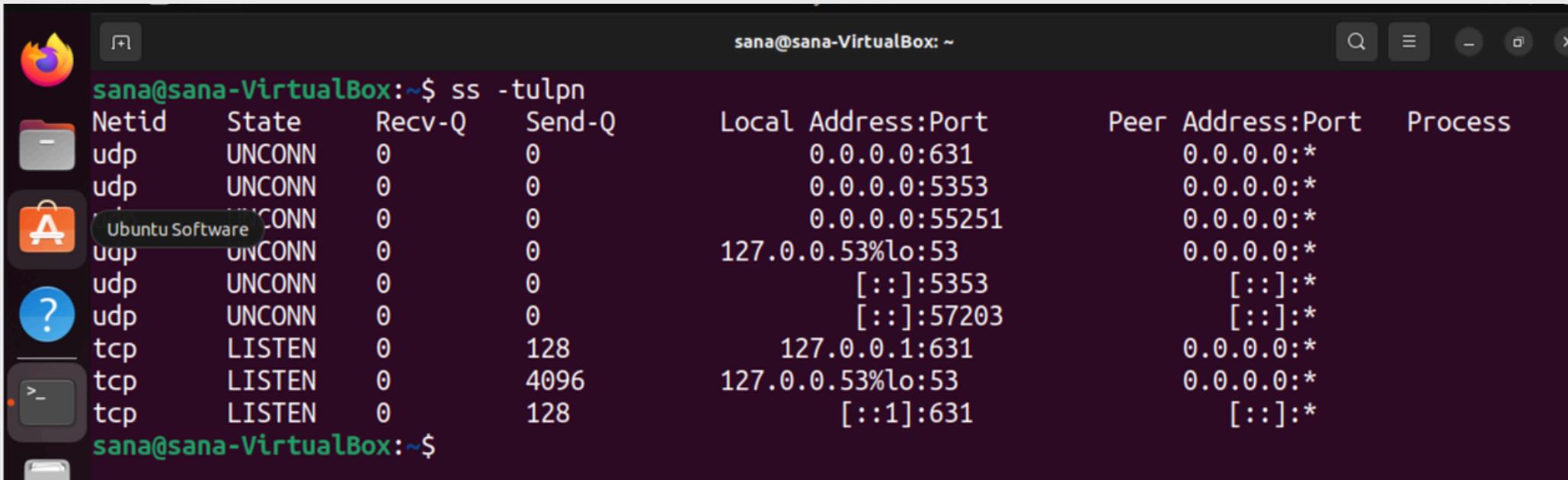
```
sana@sana-VirtualBox:~$ ping -c 5 google.com > ping_results.txt
sana@sana-VirtualBox:~$ ^C
sana@sana-VirtualBox:~$ ls
cities.txt      dirtobedelete  index.html    new.html      Public   t1
cpu_mem_usage.txt disk_usage.txt lsdatal.txt  nohup.out    r1       Templates
d1              Documents       m1           Pictures     r2       timestamp.txt
Desktop         Downloads      Music        ping_results.txt r3       Videos
dirtobedele    home          mynumbers.txt process_list.txt snap
sana@sana-VirtualBox:~$ cat ping_results.txt
PING google.com (142.251.222.142) 56(84) bytes of data.
64 bytes from pnmaaaa-az-in-f14.1e100.net (142.251.222.142): icmp_seq=1 ttl=255 time=180 ms
64 bytes from pnmaaaa-az-in-f14.1e100.net (142.251.222.142): icmp_seq=2 ttl=255 time=467 ms
64 bytes from pnmaaaa-az-in-f14.1e100.net (142.251.222.142): icmp_seq=3 ttl=255 time=302 ms
64 bytes from pnmaaaa-az-in-f14.1e100.net (142.251.222.142): icmp_seq=4 ttl=255 time=376 ms
64 bytes from pnmaaaa-az-in-f14.1e100.net (142.251.222.142): icmp_seq=5 ttl=255 time=470 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4047ms
rtt min/avg/max/mdev = 180.234/359.212/470.397/109.082 ms
sana@sana-VirtualBox:~$
```

The terminal window is part of a desktop environment with icons for various applications like a browser, file manager, and system tools visible on the left.

3. Open Ports:

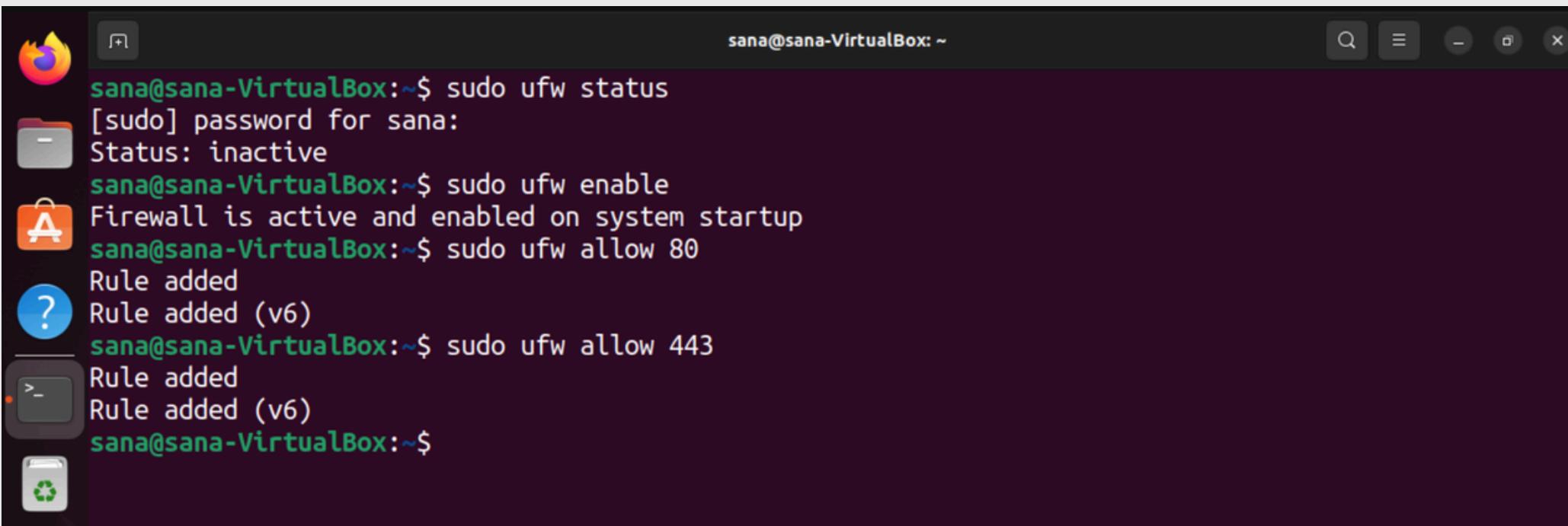
- List all open ports on the system using netstat or ss command.



```
sana@sana-VirtualBox:~$ ss -tulpn
Netid      State    Recv-Q   Send-Q     Local Address:Port      Peer Address:Port  Process
udp        UNCONN   0          0          0.0.0.0:631           0.0.0.0:*
udp        UNCONN   0          0          0.0.0.0:5353          0.0.0.0:*
tcp        LISTEN   0          128         127.0.0.1:631        0.0.0.0:*
tcp        LISTEN   0          4096        127.0.0.53%lo:53      0.0.0.0:*
tcp        LISTEN   0          128         [::]:631             [::]:*
```

4. Firewall Configuration:

- Check if the ufw (Uncomplicated Firewall) is installed and running. If not, install and enable
- Allow incoming connections on port 80 (HTTP) and 443 (HTTPS).



```
sana@sana-VirtualBox:~$ sudo ufw status
[sudo] password for sana:
Status: inactive

sana@sana-VirtualBox:~$ sudo ufw enable
Firewall is active and enabled on system startup

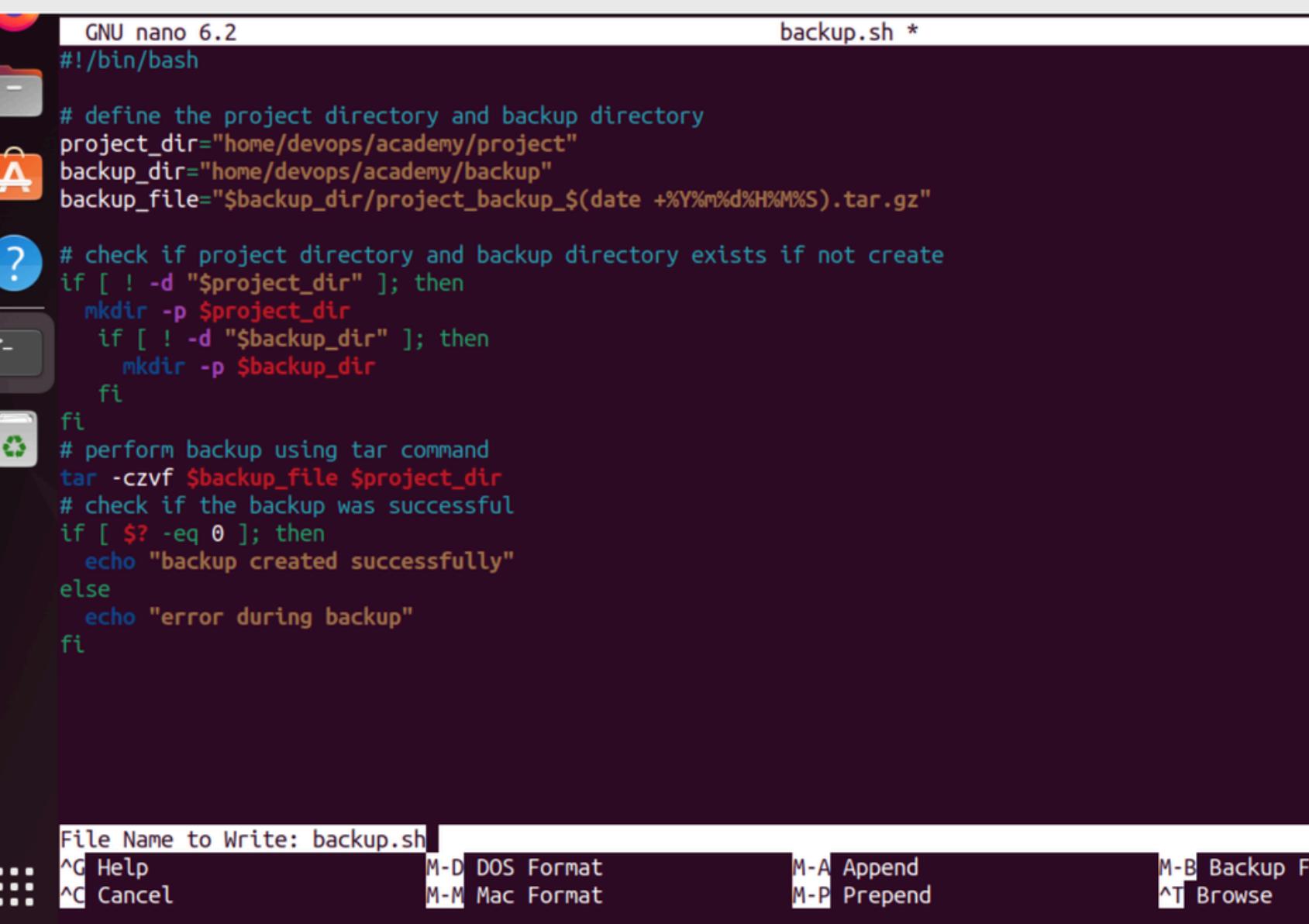
sana@sana-VirtualBox:~$ sudo ufw allow 80
Rule added

sana@sana-VirtualBox:~$ sudo ufw allow 443
Rule added
```

Task 5: Shell Scripting

1. Backup Script:

- Write a shell script named backup.sh that compresses the project directory into a tar.gz file and saves it in the /home/devops/academy/backup directory. The script should include error handling to ensure the backup only proceeds if the project directory exists.



The screenshot shows a terminal window with the nano 6.2 text editor open. The file is named "backup.sh". The script content is as follows:

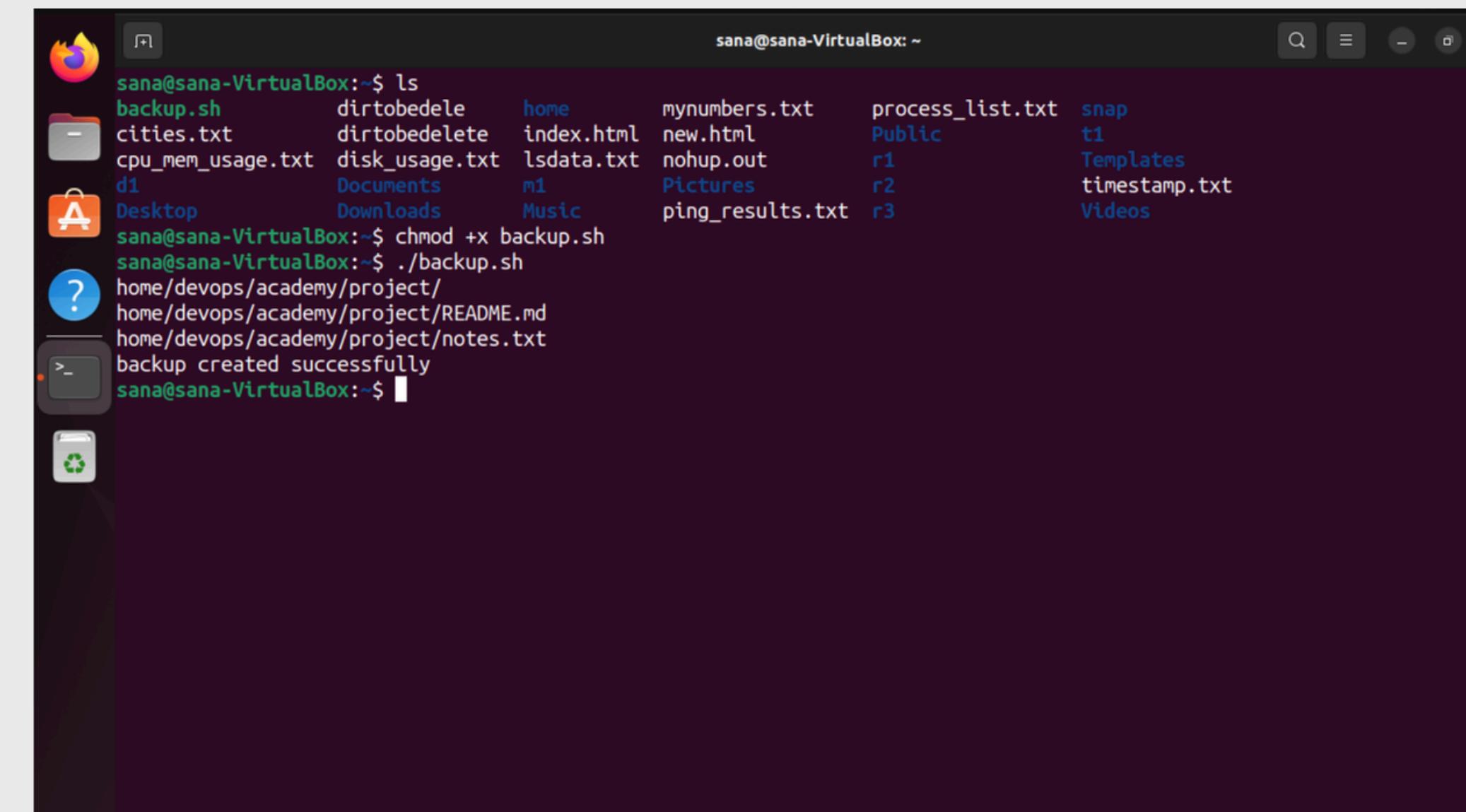
```
GNU nano 6.2
backup.sh *

#!/bin/bash

# define the project directory and backup directory
project_dir="/home/devops/academy/project"
backup_dir="/home/devops/academy/backup"
backup_file="$backup_dir/project_backup_$(date +\%Y\%m\%d\%H\%M\%S).tar.gz"

# check if project directory and backup directory exists if not create
if [ ! -d "$project_dir" ]; then
    mkdir -p $project_dir
    if [ ! -d "$backup_dir" ]; then
        mkdir -p $backup_dir
    fi
fi
# perform backup using tar command
tar -czvf $backup_file $project_dir
# check if the backup was successful
if [ $? -eq 0 ]; then
    echo "backup created successfully"
else
    echo "error during backup"
fi
```

At the bottom of the terminal window, there is a menu bar with the following options: File Name to Write: backup.sh, ^G Help, M-D DOS Format, M-A Append, M-B Backup Fi, ^C Cancel, M-M Mac Format, M-P Prepend, ^T Browse.



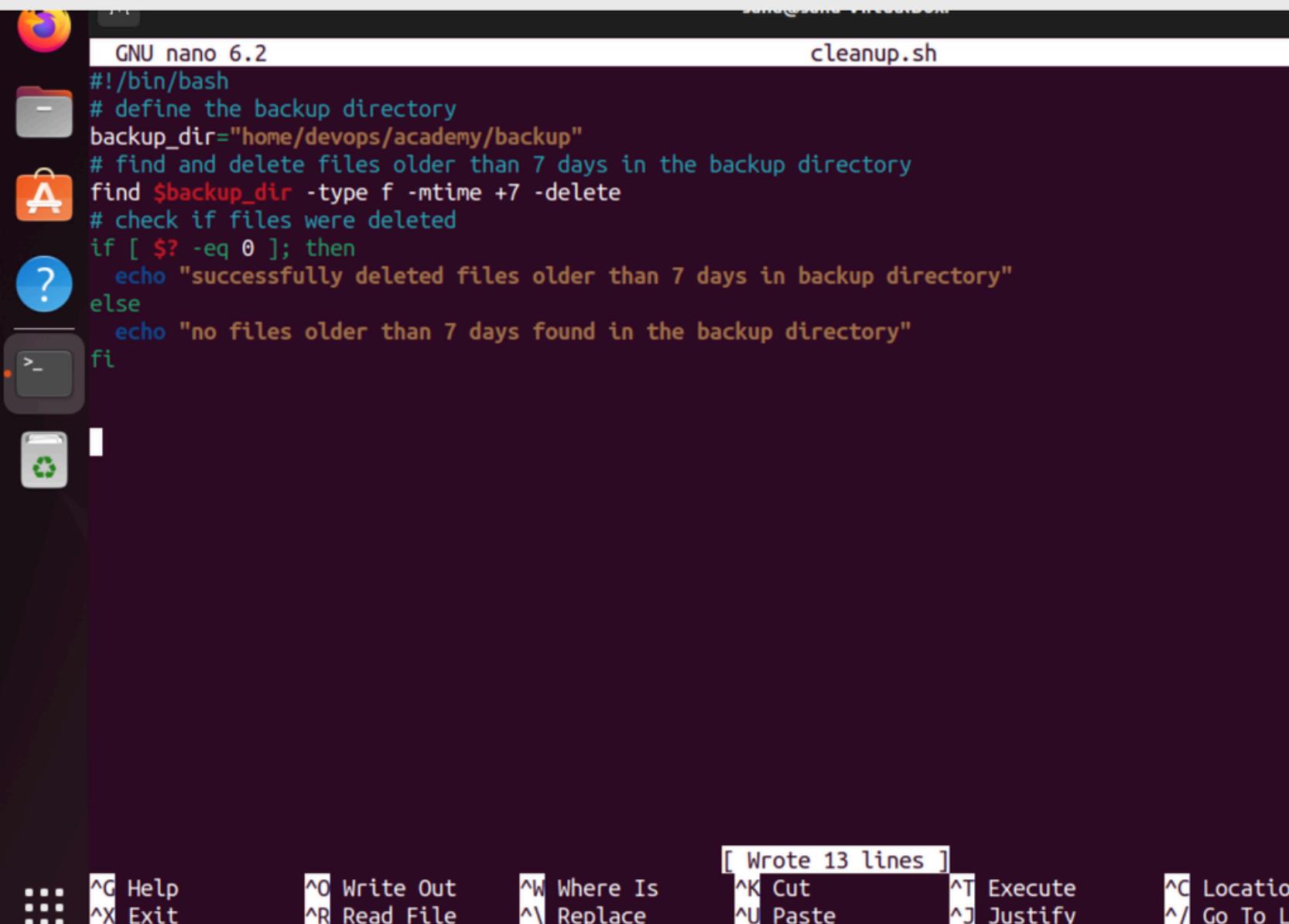
The screenshot shows a terminal window with the following session:

```
sana@sana-VirtualBox:~$ ls
backup.sh      dirtobedelete   home      mynumbers.txt  process_list.txt  snap
cities.txt     dirtobedelete  index.html new.html       Public          t1
cpu_mem_usage.txt disk_usage.txt lsdata.txt  nohup.out     r1             Templates
d1           Documents       m1         Pictures      r2             timestamp.txt
Desktop       Downloads       Music      ping_results.txt r3             Videos

sana@sana-VirtualBox:~$ chmod +x backup.sh
sana@sana-VirtualBox:~$ ./backup.sh
/home/devops/academy/project/
/home/devops/academy/project/README.md
/home/devops/academy/project/notes.txt
backup created successfully
sana@sana-VirtualBox:~$
```

2. Automation Script:

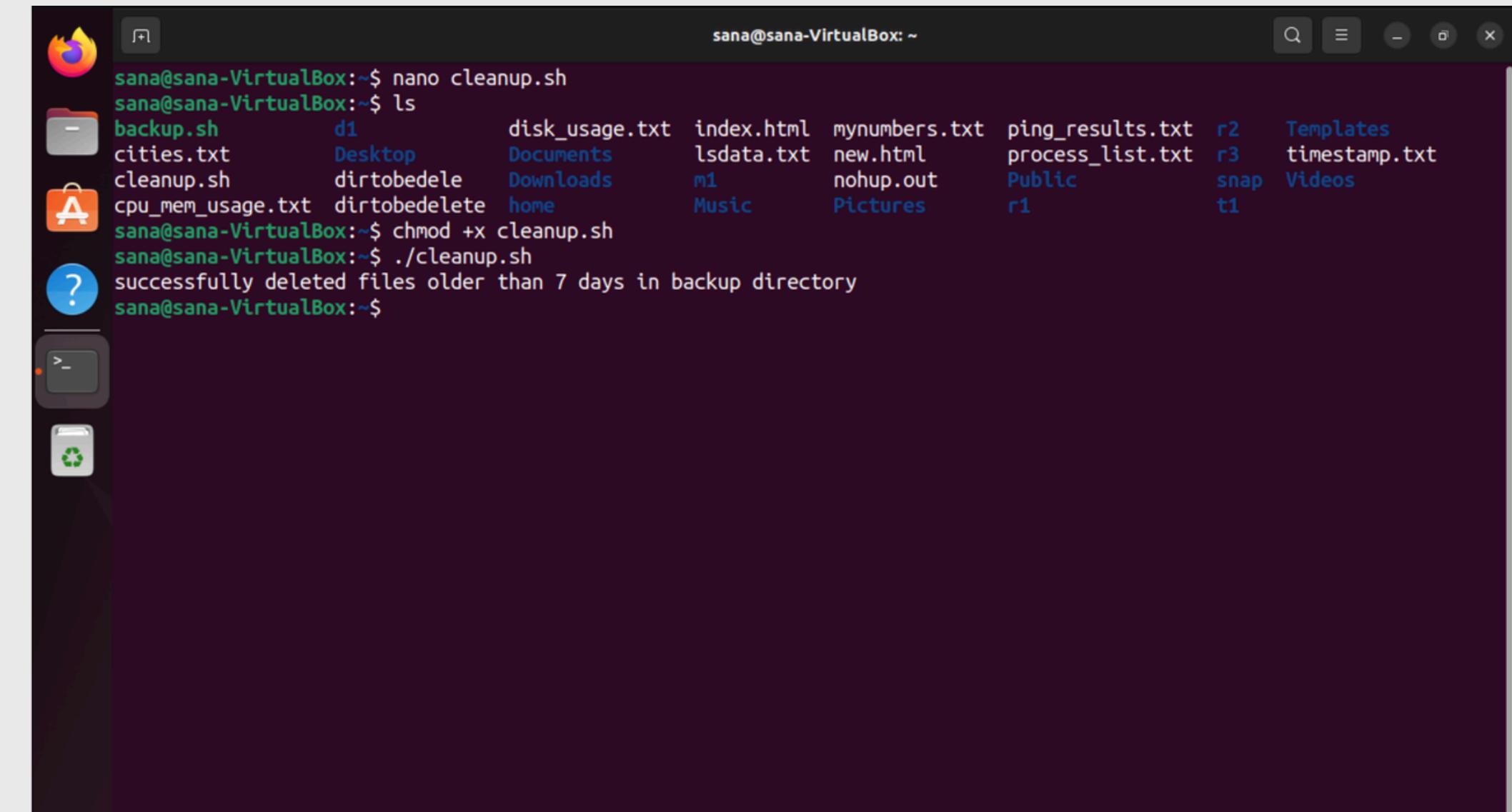
- Create a script named cleanup.sh that deletes all files in the /home/devops/academy/backup directory that are older than 7 days. Schedule this script to run daily using cron.



The screenshot shows a terminal window with the title "cleanup.sh". The file content is as follows:

```
GNU nano 6.2
cleanup.sh
#!/bin/bash
# define the backup directory
backup_dir="/home/devops/academy/backup"
# find and delete files older than 7 days in the backup directory
find $backup_dir -type f -mtime +7 -delete
# check if files were deleted
if [ $? -eq 0 ]; then
    echo "successfully deleted files older than 7 days in backup directory"
else
    echo "no files older than 7 days found in the backup directory"
fi
```

The terminal status bar at the bottom indicates "[Wrote 13 lines]".



The screenshot shows a terminal window with the title "sana@sana-VirtualBox: ~". The session log is as follows:

```
sana@sana-VirtualBox:~$ nano cleanup.sh
sana@sana-VirtualBox:~$ ls
backup.sh      d1          disk_usage.txt  index.html  mynumbers.txt  ping_results.txt  r2  Templates
cities.txt     Desktop      Documents       lsdata.txt   new.html      process_list.txt  r3  timestamp.txt
cleanup.sh     dirtobedelete  Downloads      m1          nohup.out    Public           snap  Videos
cpu_mem_usage.txt  dirtobedelete  home        Music       Pictures      r1
sana@sana-VirtualBox:~$ chmod +x cleanup.sh
sana@sana-VirtualBox:~$ ./cleanup.sh
successfully deleted files older than 7 days in backup directory
sana@sana-VirtualBox:~$
```

GNU nano 6.2 /tmp/crontab.eQayxS/crontab

```
# 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
0 4 * * * /home/cleanup.sh
```

[Read 25 lines]

^O Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^O Location M-U
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E

sana@sana-VirtualBox:~\$ crontab -e

```
No modification made
```

sana@sana-VirtualBox:~\$ crontab -l

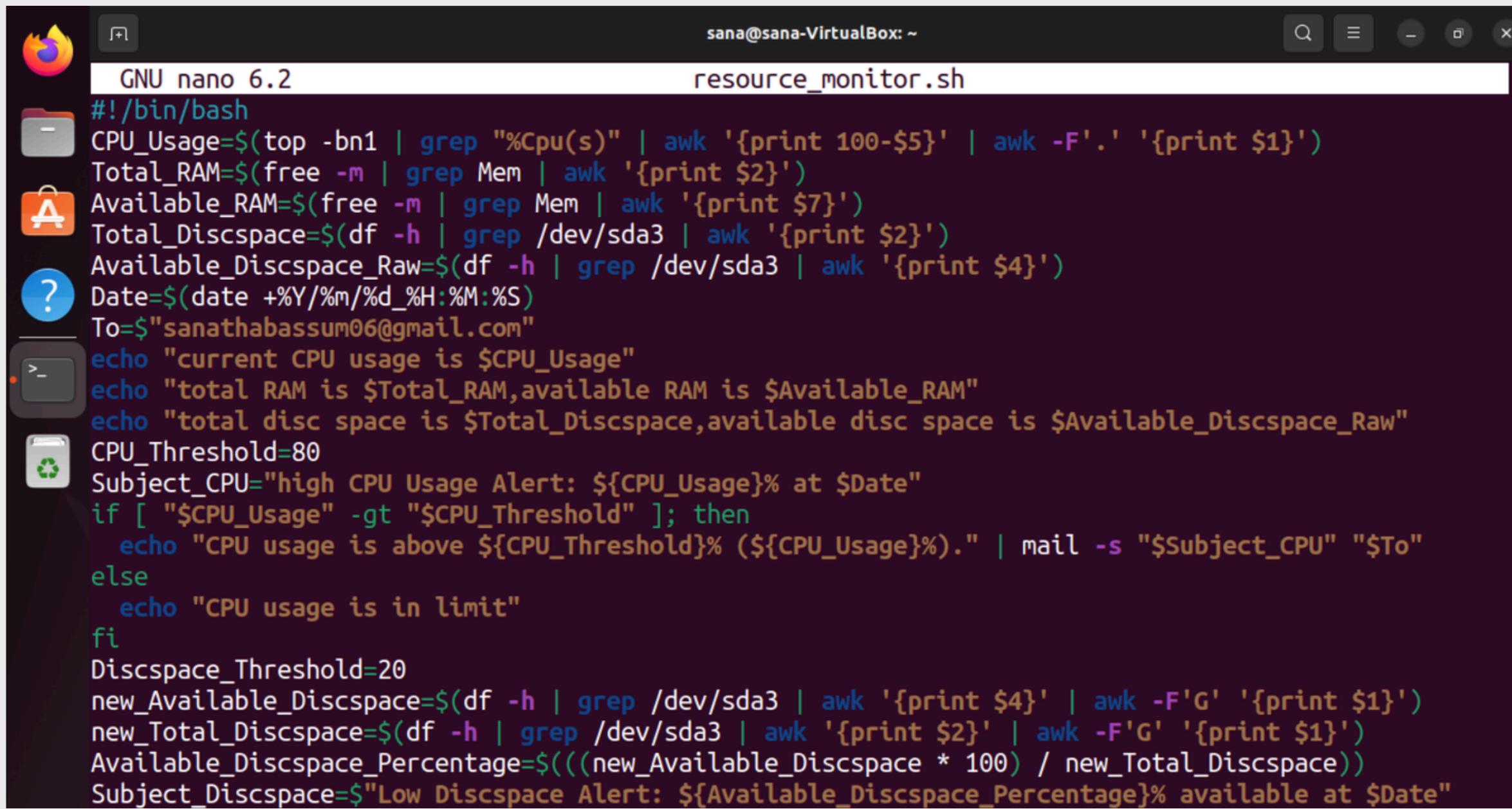
```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), 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
0 4 * * * /home/cleanup.sh
```

sana@sana-VirtualBox:~\$

Advanced Level

- **Advanced Shell Scripting:**

- Write a script named resource_monitor.sh that:
 1. Monitors CPU, memory, and disk usage
 2. Sends an alert email if CPU usage exceeds 80% or available disk space falls below 20%..



The screenshot shows a terminal window with the title bar "GNU nano 6.2" and the file name "resource_monitor.sh". The script content is as follows:

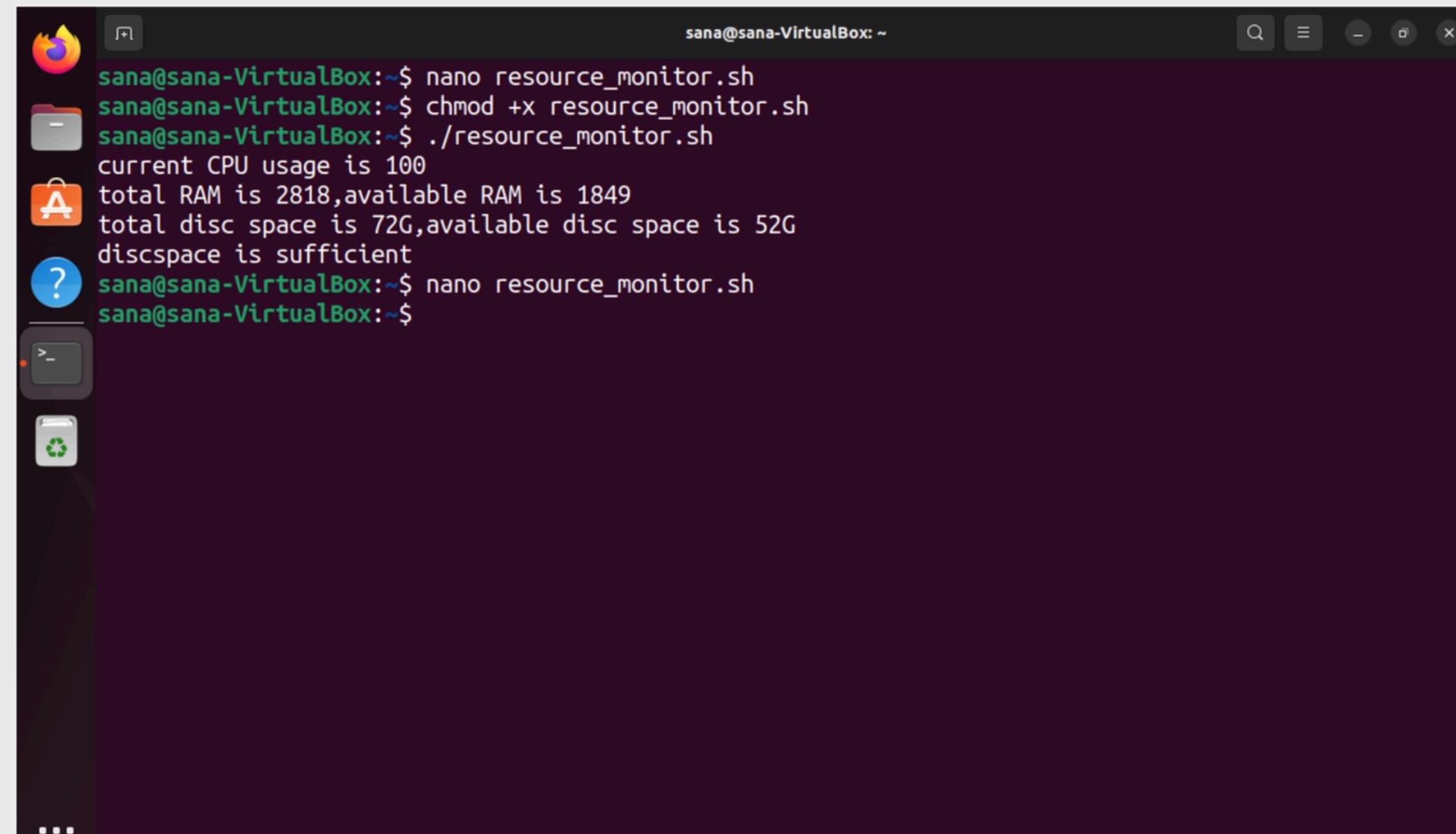
```
#!/bin/bash
CPU_Usage=$(top -bn1 | grep "%Cpu(s)" | awk '{print 100-$5}' | awk -F'.' '{print $1}')
Total_RAM=$(free -m | grep Mem | awk '{print $2}')
Available_RAM=$(free -m | grep Mem | awk '{print $7}')
Total_Discspace=$(df -h | grep /dev/sda3 | awk '{print $2}')
Available_Discspace_Raw=$(df -h | grep /dev/sda3 | awk '{print $4}')
Date=$(date +%Y/%m/%d_%H:%M:%S)
To="$sanathabassum06@gmail.com"
echo "current CPU usage is $CPU_Usage"
echo "total RAM is $Total_RAM,available RAM is $Available_RAM"
echo "total disc space is $Total_Discspace,available disc space is $Available_Discspace_Raw"
CPU_Threshold=80
Subject_CPU="high CPU Usage Alert: ${CPU_Usage}% at $Date"
if [ "$CPU_Usage" -gt "$CPU_Threshold" ]; then
    echo "CPU usage is above ${CPU_Threshold}% (${CPU_Usage}%)." | mail -s "$Subject_CPU" "$To"
else
    echo "CPU usage is in limit"
fi
Discspace_Threshold=20
new_Available_Discspace=$(df -h | grep /dev/sda3 | awk '{print $4}' | awk -F'G' '{print $1}')
new_Total_Discspace=$(df -h | grep /dev/sda3 | awk '{print $2}' | awk -F'G' '{print $1}')
Available_Discspace_Percentage=$((($new_Available_Discspace * 100) / new_Total_Discspace))
Subject_Discspace="$Low Discspace Alert: ${Available_Discspace_Percentage}% available at $Date"
```

```
if [ "$Available_Discspace_Percentage" -lt "$Discspace_Threshold" ]; then
    echo "available disc space is below ${Discspace_Threshold} (${Available_Discspace_Percentage}%)"
else
    echo "discspace is sufficient"
fi
```

... ^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location

```
if [ "$Available_Discspace_Percentage" -lt "$Discspace_Threshold" ]; then
    | mail -s "$Subject_Discspace" "$To"
else
    echo "discspace is sufficient"
fi
```

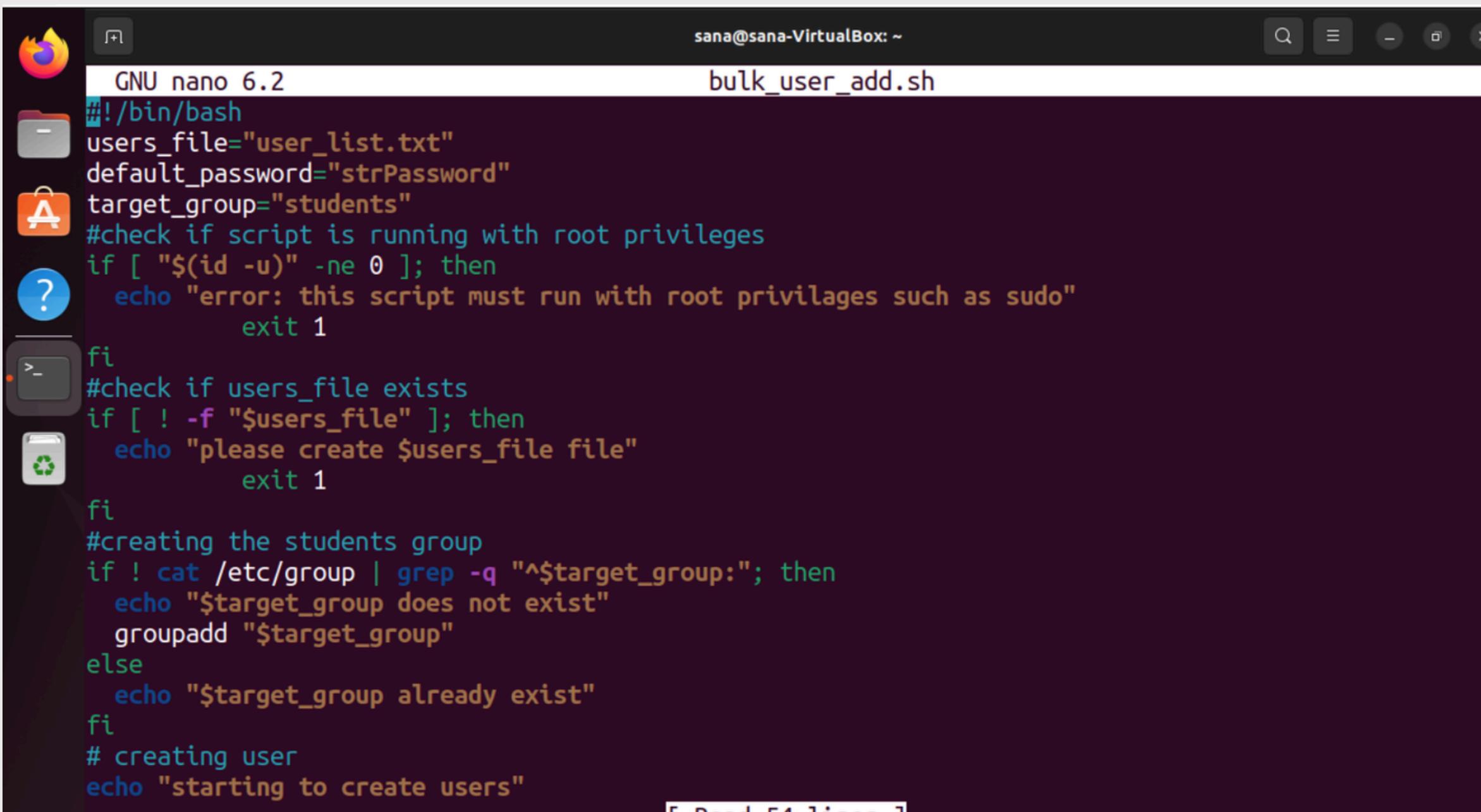
... ^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location



```
sana@sana-VirtualBox:~$ nano resource_monitor.sh
sana@sana-VirtualBox:~$ chmod +x resource_monitor.sh
sana@sana-VirtualBox:~$ ./resource_monitor.sh
current CPU usage is 100
total RAM is 2818,available RAM is 1849
total disc space is 72G,available disc space is 52G
discspace is sufficient
sana@sana-VirtualBox:~$ nano resource_monitor.sh
sana@sana-VirtualBox:~$
```

- **User Management Automation:**

- Write a script named bulk_user_add.sh that:
 1. Reads a list of usernames from a file named user_list.txt.
 2. Creates each user and sets a default password.
 3. Adds each user to a specific group (e.g., students).
 4. Ensures each user has a home directory created.



The screenshot shows a terminal window titled "bulk_user_add.sh" running on a Linux system. The window title bar indicates the session is "sana@sana-VirtualBox: ~". The terminal interface includes standard icons for file operations (trash, copy, paste, etc.) and a search bar. The main content of the terminal is the source code for the "bulk_user_add.sh" script, which is written in Bash. The script starts by defining variables for the users file, default password, and target group. It then checks if it's running with root privileges and if the users file exists. If either condition fails, it exits with an error message. Next, it checks if the target group exists; if not, it creates it using the "groupadd" command. Finally, it prints a message indicating it is starting to create users.

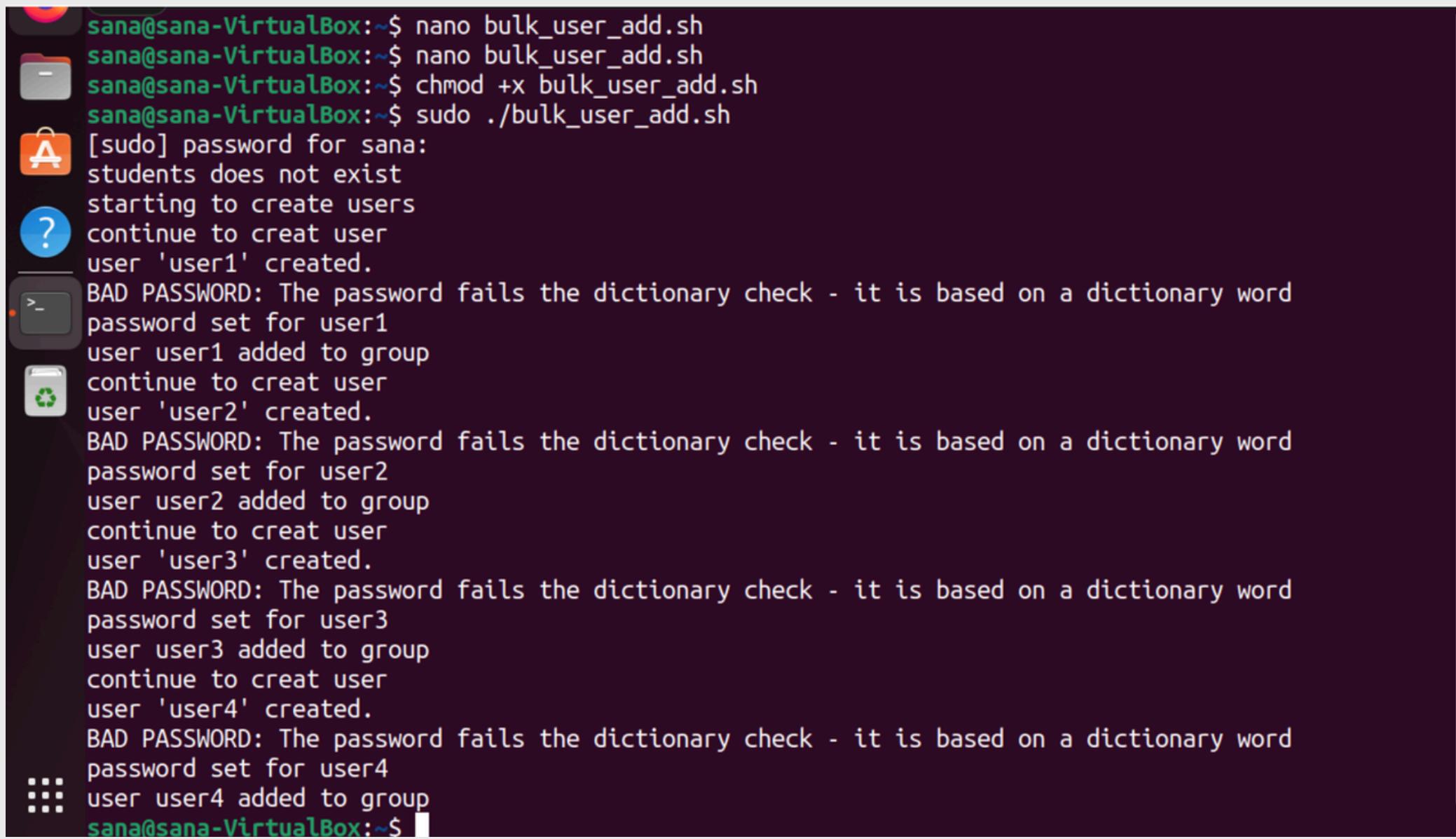
```
GNU nano 6.2
#!/bin/bash
users_file="user_list.txt"
default_password="strPassword"
target_group="students"
#check if script is running with root privileges
if [ "$(id -u)" -ne 0 ]; then
    echo "error: this script must run with root privileges such as sudo"
    exit 1
fi
#check if users_file exists
if [ ! -f "$users_file" ]; then
    echo "please create $users_file file"
    exit 1
fi
#creating the students group
if ! cat /etc/group | grep -q "^$target_group:"; then
    echo "$target_group does not exist"
    groupadd "$target_group"
else
    echo "$target_group already exist"
fi
# creating user
echo "starting to create users"
```

GNU nano 6.2 bulk_user_add.sh

```
#read user from file
while IFS= read -r username; do
    if cat /etc/passwd | grep -q "^$username:"; then
        echo "user already exists"
    else
        echo "continue to creat user"
        useradd -m "$username"
        if [ $? -eq 0 ]; then
            echo "user '$username' created."
        else
            echo "error: failed to create user $username."
        fi
    #set password
    echo "${username}:${default_password}" | chpasswd
    if [ $? -eq 0 ]; then
        echo "password set for $username"
    else
        echo "failed to set password for $username"
    fi
    #add user to group
    usermod -aG "$target_group" "$username"
    if [ $? -eq 0 ]; then
        echo "user $username added to group $target_group"
```

```
usermod -aG "$target_group" "$username"
if [ $? -eq 0 ]; then
    echo "user $username added to group $target_group"
else
    echo "error: failed to add $target_group group to user $username"
fi
done < "$users_file"

::: ^G Help      ^O Write Out   ^W Where Is     ^K Cut       ^T Execute   ^C Location
::: ^X Exit      ^R Read File   ^\ Replace      ^U Paste     ^J Justify   ^/ Go To Line
```



```
sana@sana-VirtualBox:~$ nano bulk_user_add.sh
sana@sana-VirtualBox:~$ nano bulk_user_add.sh
sana@sana-VirtualBox:~$ chmod +x bulk_user_add.sh
sana@sana-VirtualBox:~$ sudo ./bulk_user_add.sh
[sudo] password for sana:
students does not exist
starting to create users
continue to creat user
user 'user1' created.
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
password set for user1
user user1 added to group
continue to creat user
user 'user2' created.
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
password set for user2
user user2 added to group
continue to creat user
user 'user3' created.
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
password set for user3
user user3 added to group
continue to creat user
user 'user4' created.
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
password set for user4
user user4 added to group
sana@sana-VirtualBox:~$
```

THANK YOU