LINUX CHALLENGE FOR DEVOPS

By: Sana thabassum

ersanathabassum06@gmail.com

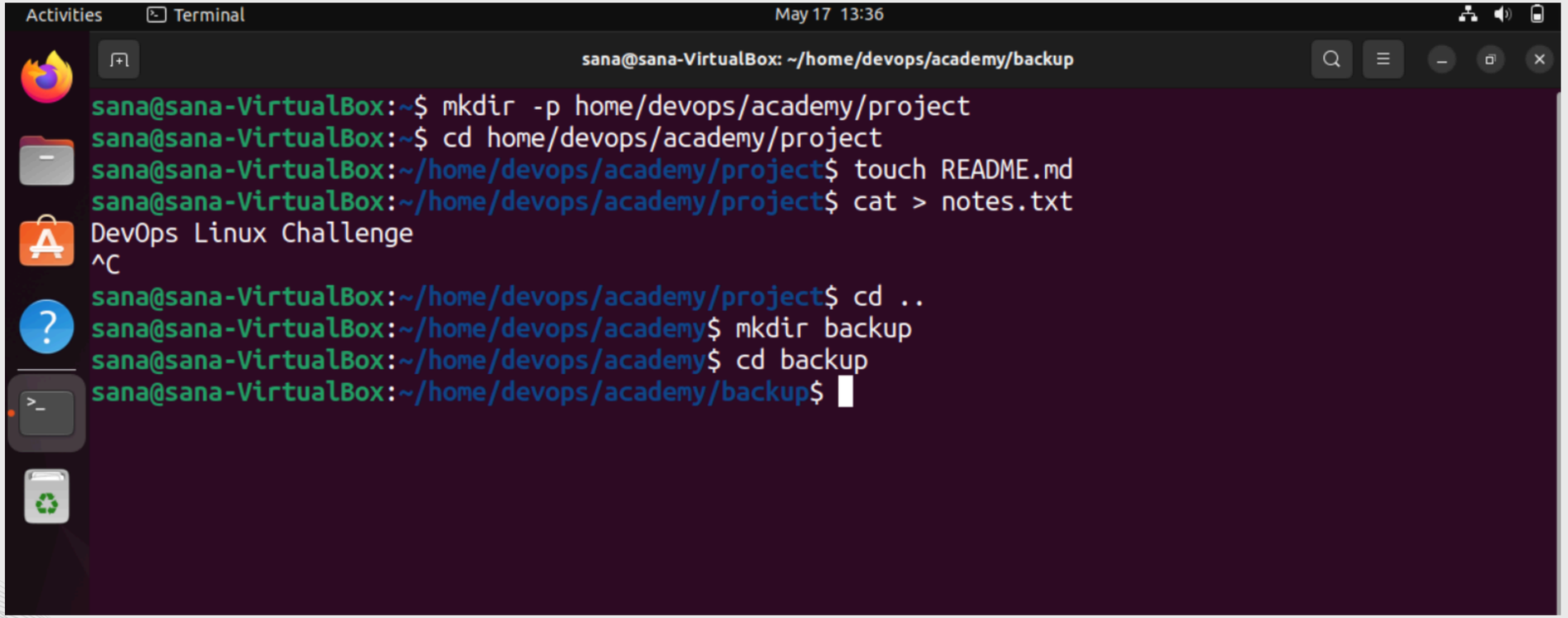
TASK 1: BASIC LINUX COMMANDS

1. Create and Navigate Directories:

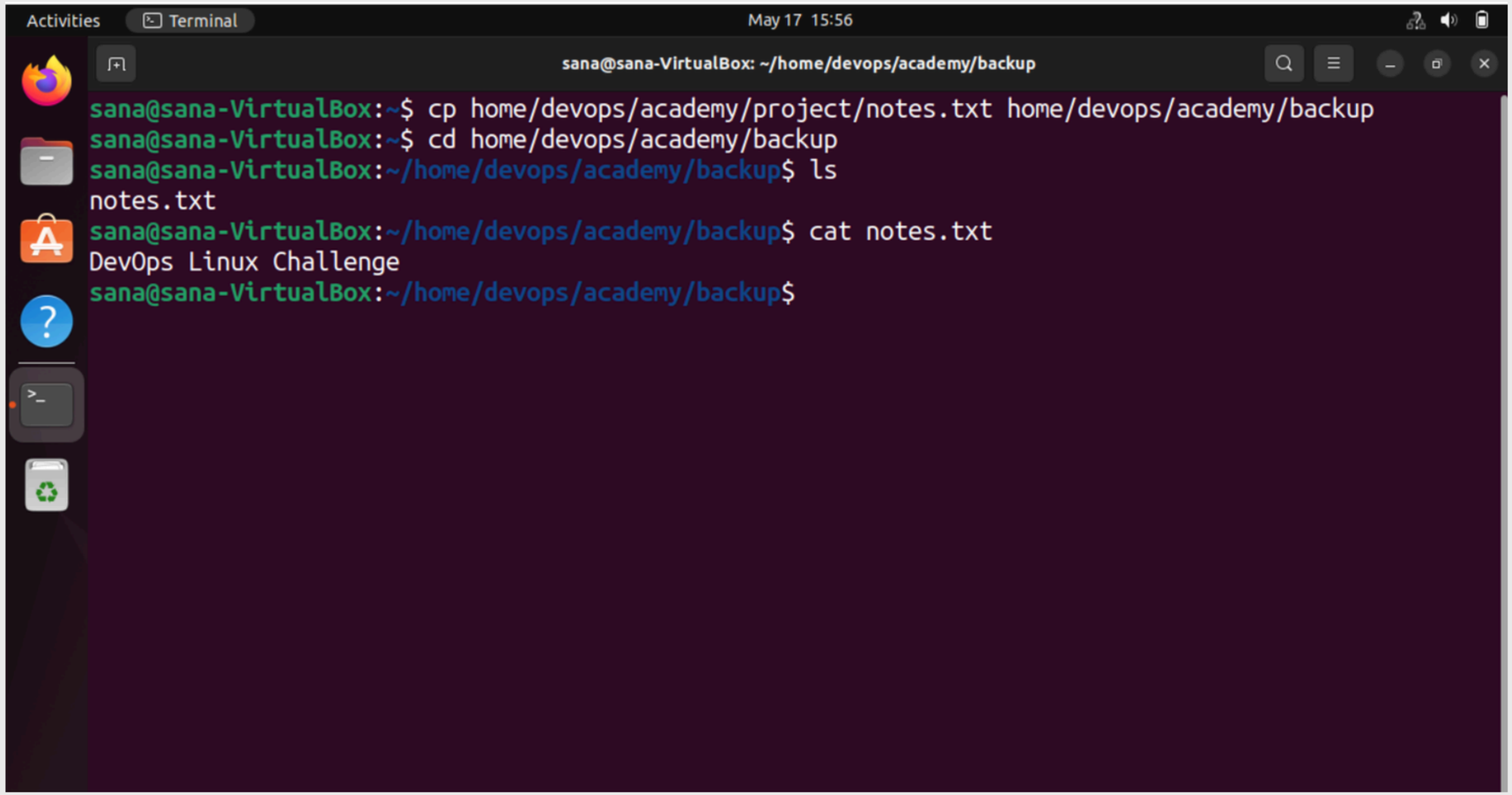
- Create a directory structure follows:/home/devops/academy/project.
- Navigate into the project directory.

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.

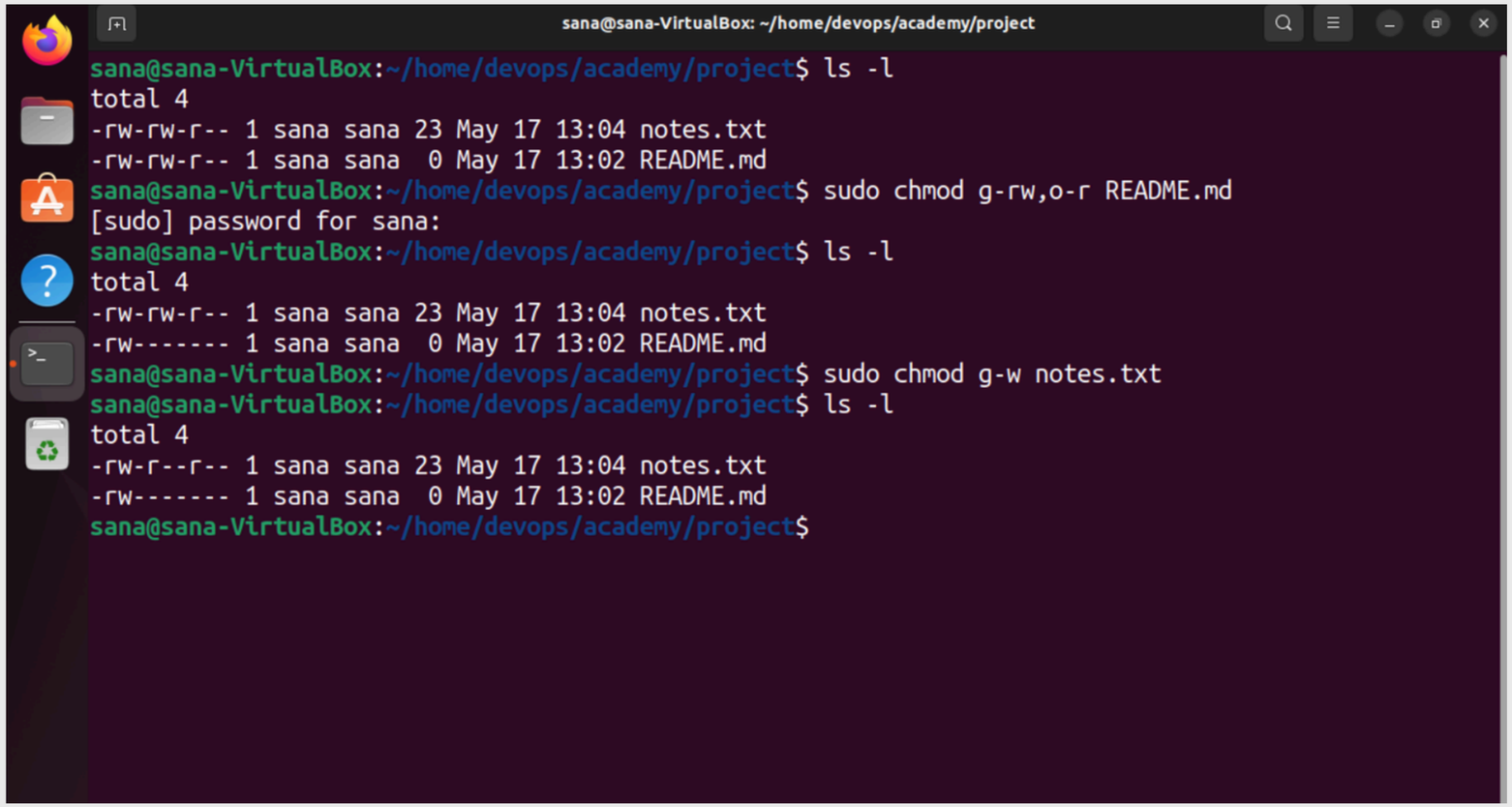


- Copy notes.txt to /home/devops/academy/backup/.



3. Permissions:

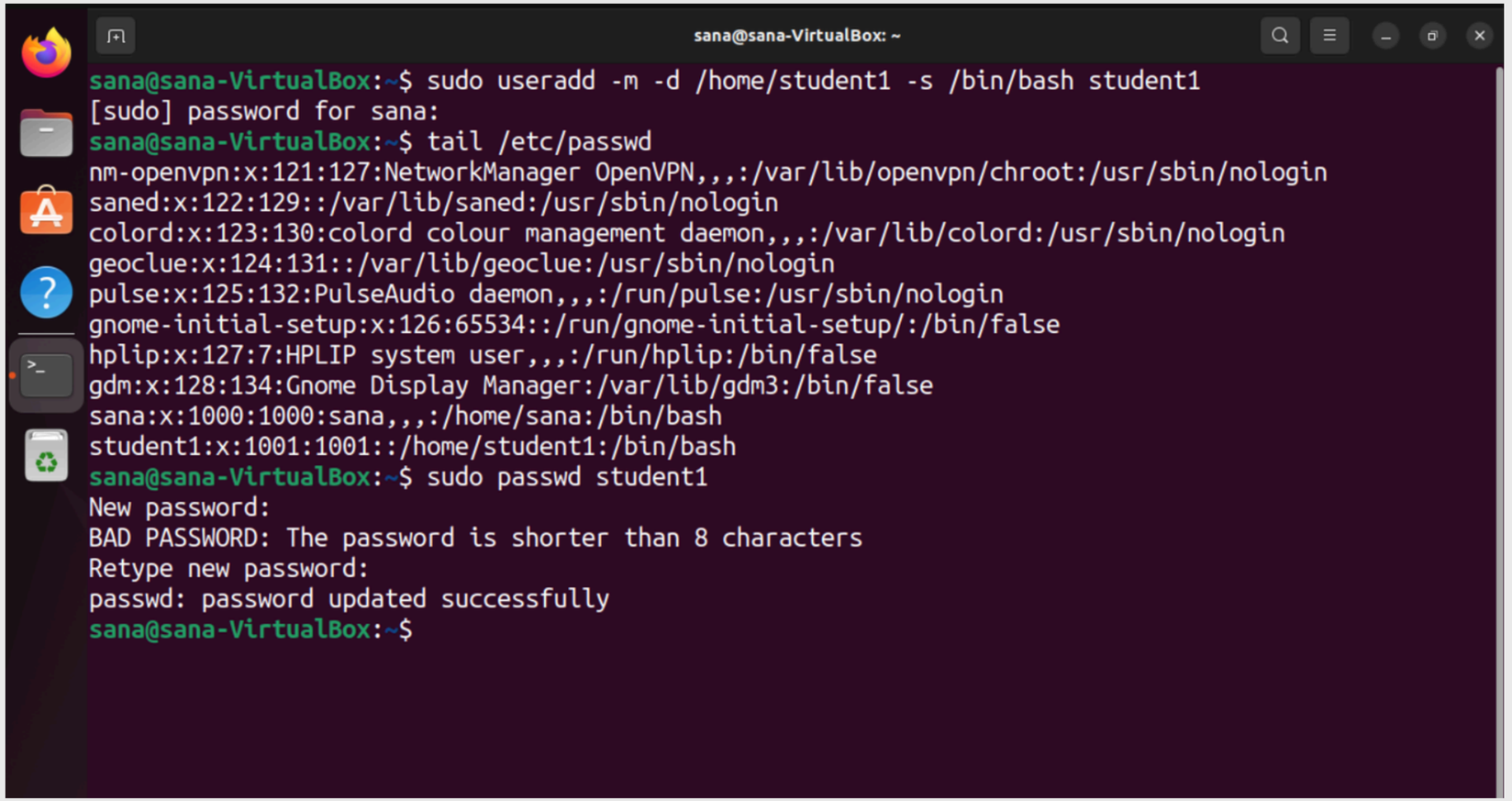
- Change the permissions of README.md to be readable and writable only by the owner.
- Make notes.txt readable by everyone, but writable only by the owner.



TASK 2: USER AND GROUP MANAGEMENT

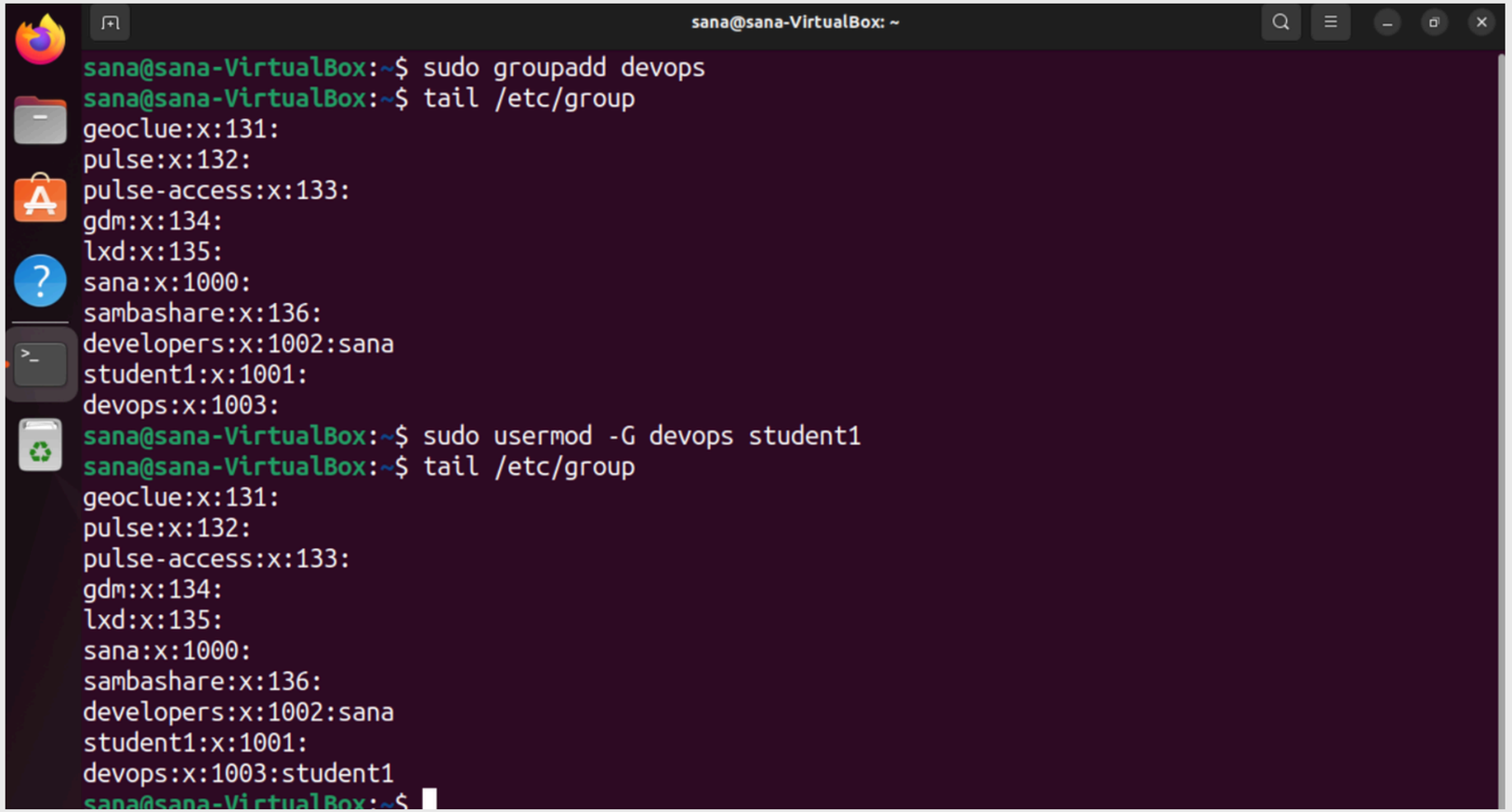
1. Create Users:

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



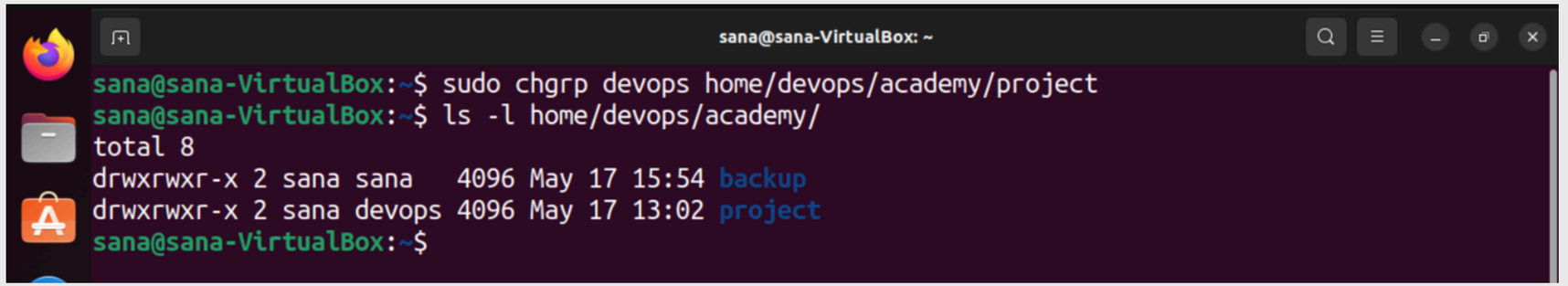
2. Groups:

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



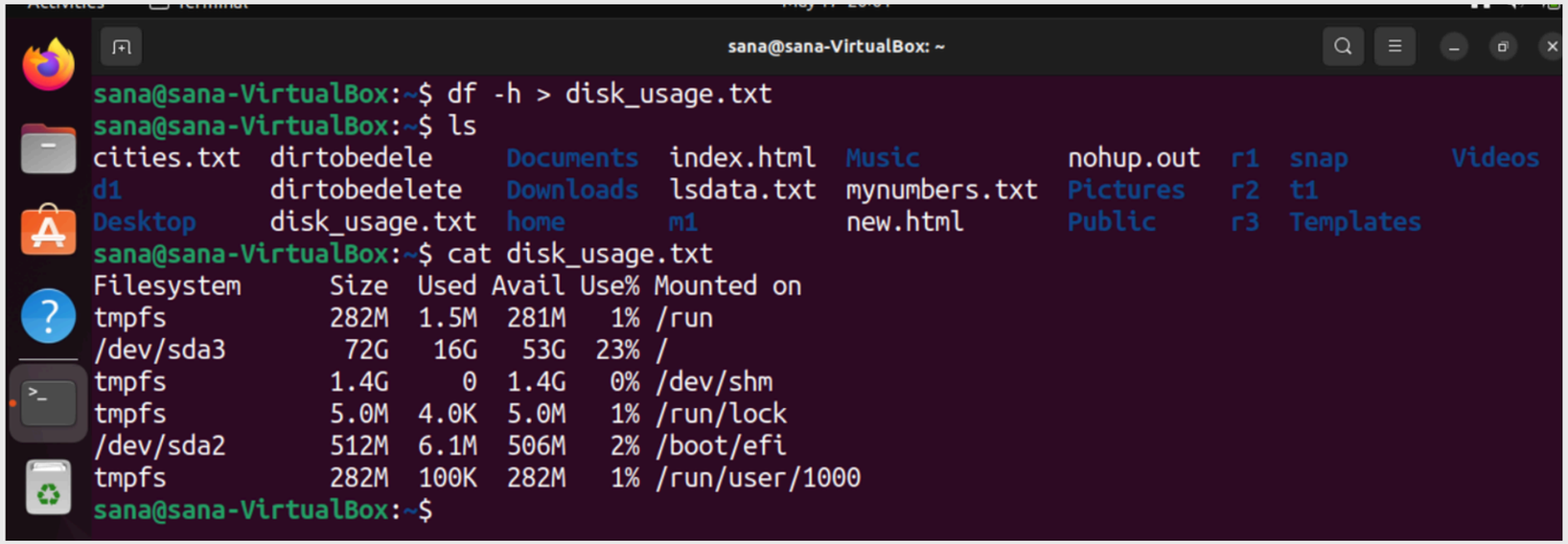
3. User Permissions:

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



4. Disk Usage:

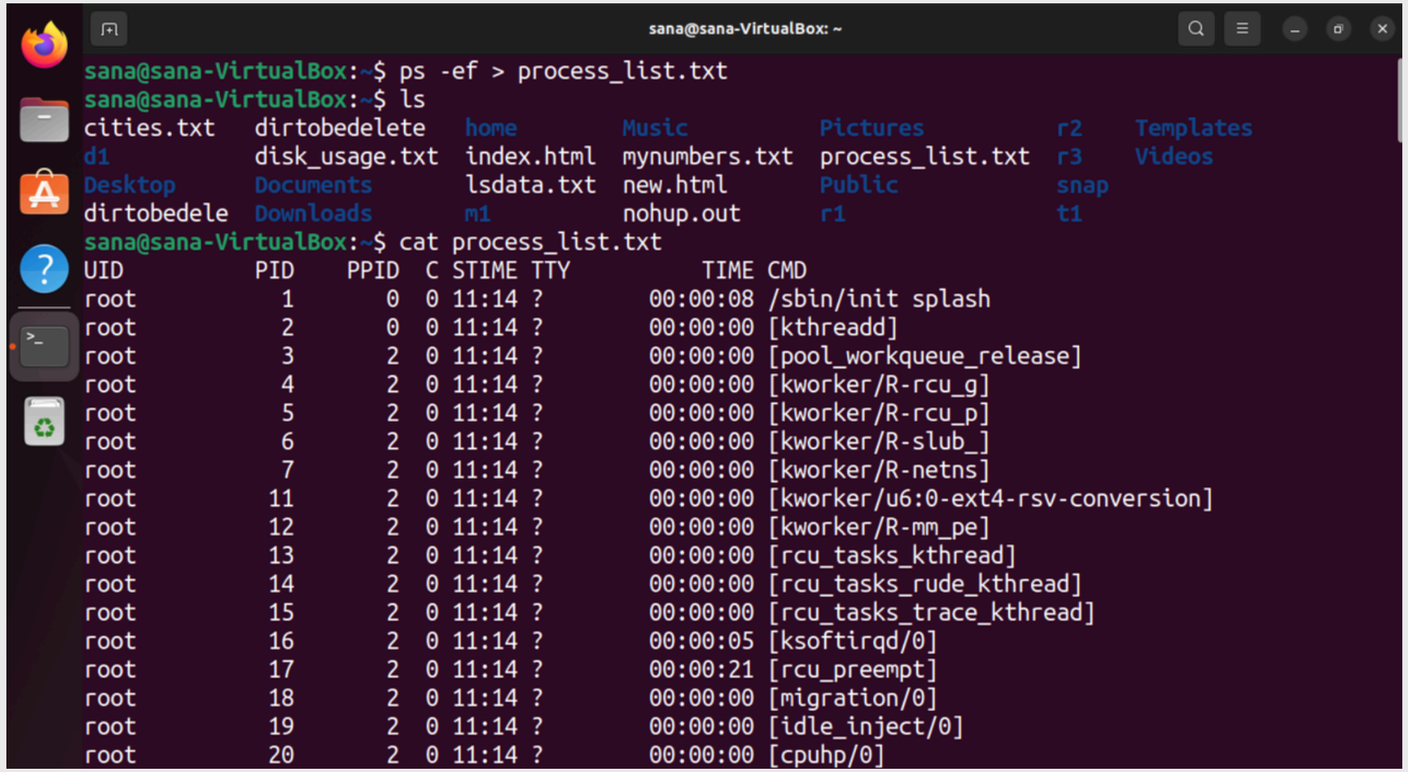
- Display disk usage in human-readable format for /home directory and save the output to a file named disk_usage.txt.

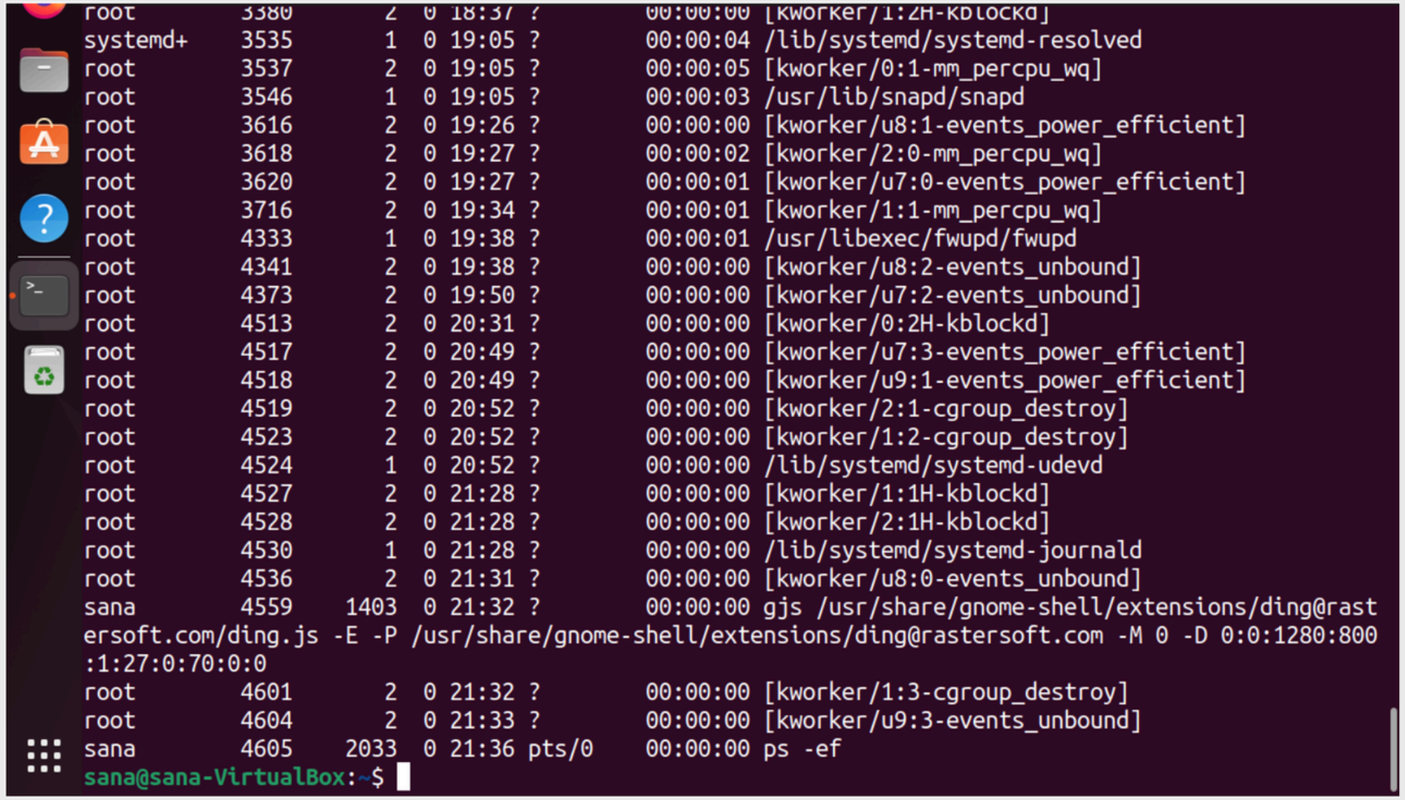


TASK 3: PROCESS MANAGEMENT

1. List Processes:

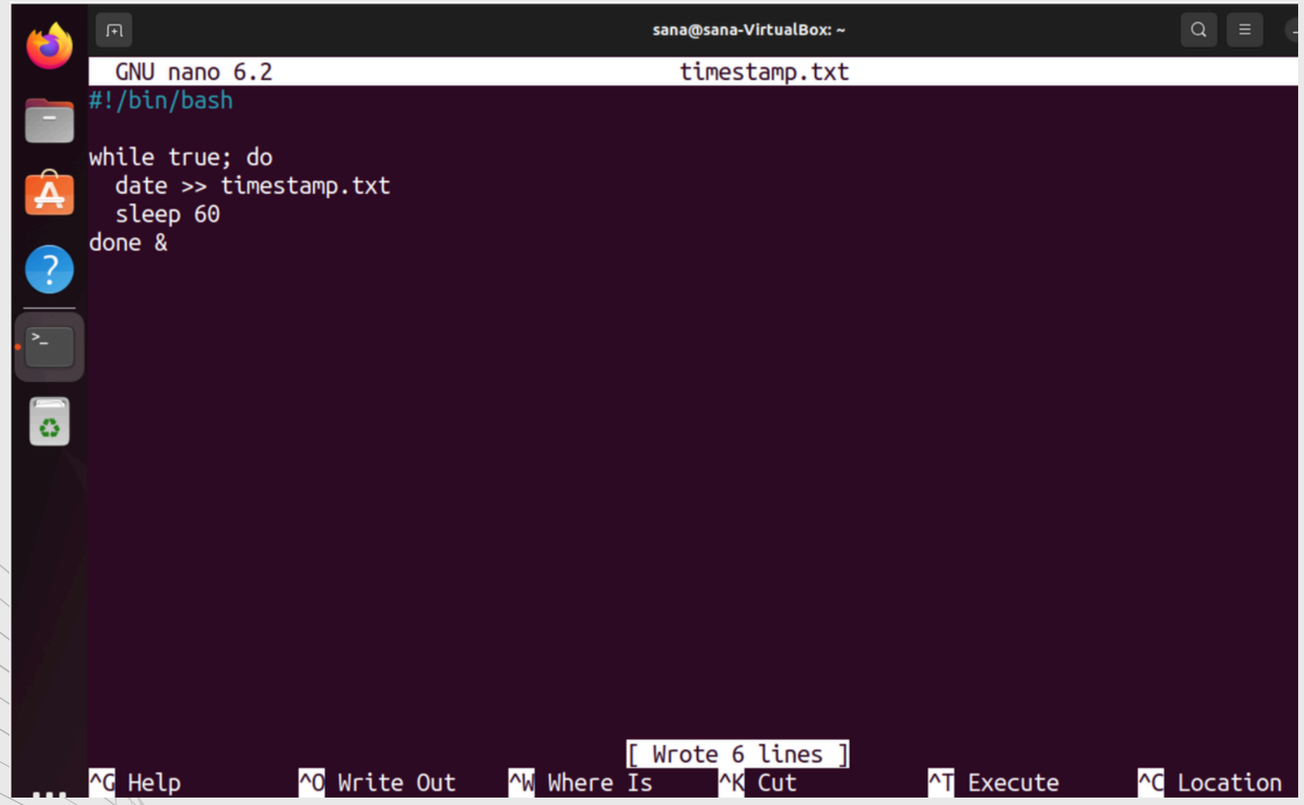
- Display all running processes and redirect the output to a file named process_list.txt.

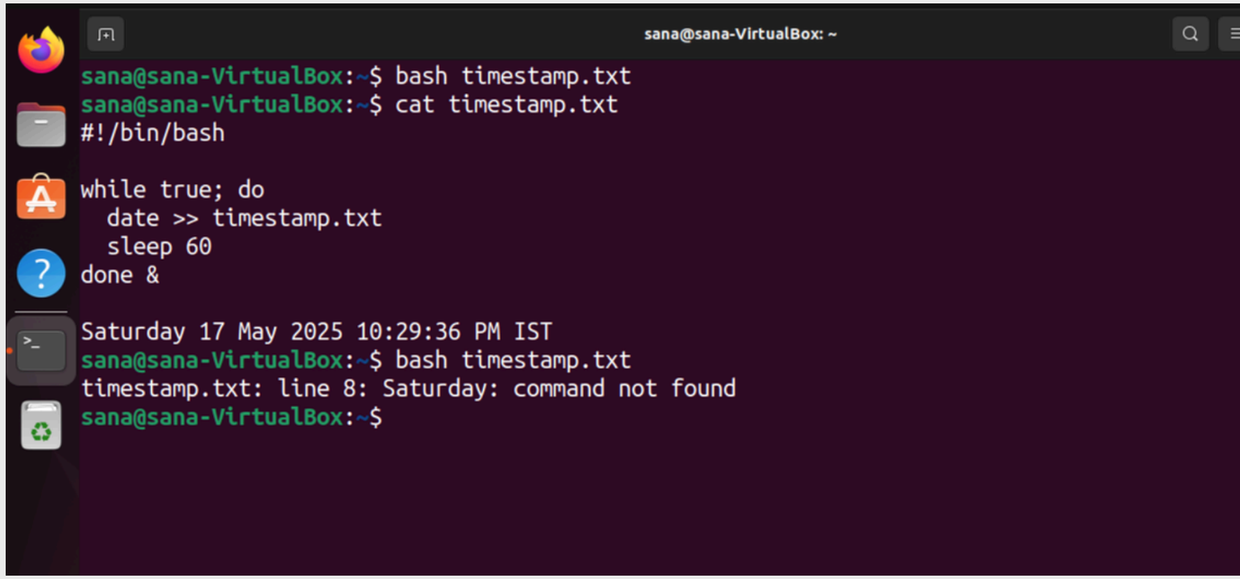



root 3380 2 0 18:37 ? 00:00:00 [kworker/1:2H-kblockd]
systemd+ 3535 1 0 19:05 ? 00:00:04 /lib/systemd/systemd-resolved
root 3537 2 0 19:05 ? 00:00:05 [kworker/0:1-mm_percpu_wq]
root 3546 1 0 19:05 ? 00:00:03 /usr/lib/snapd/snapd
root 3616 2 0 19:26 ? 00:00:00 [kworker/u8:1-events_power_efficient]
root 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:

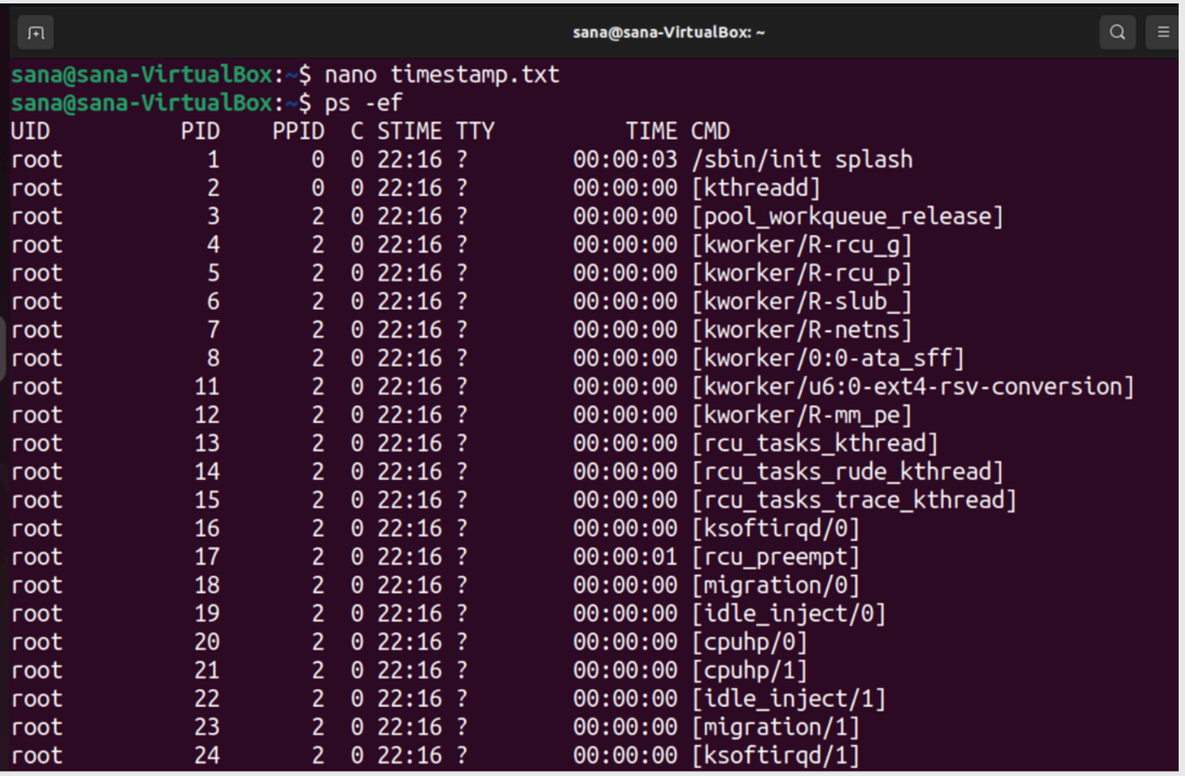
- Start a simple background process that writes the current date and time to a file named timestamp.txt every minute using a while loop and sleep.

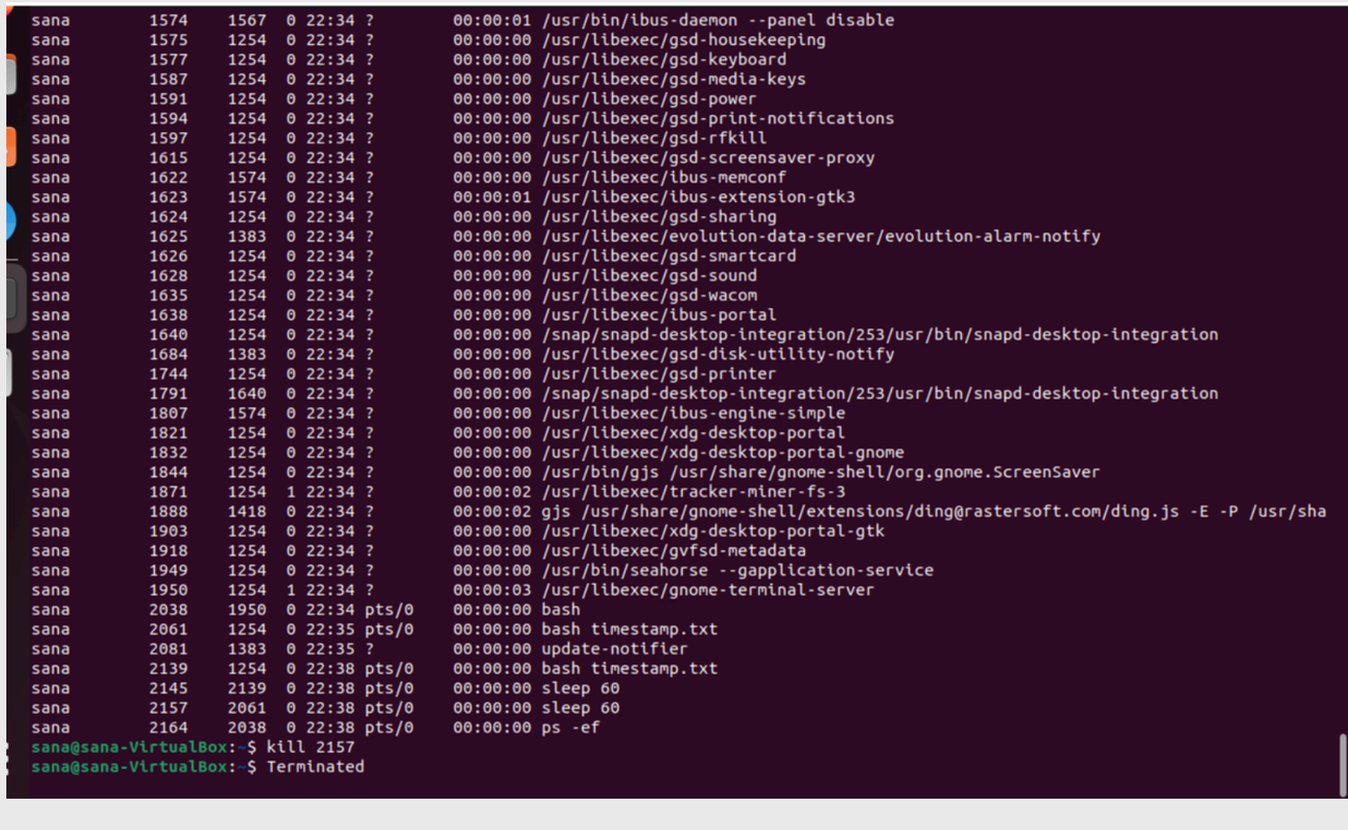




3. Kill Process:

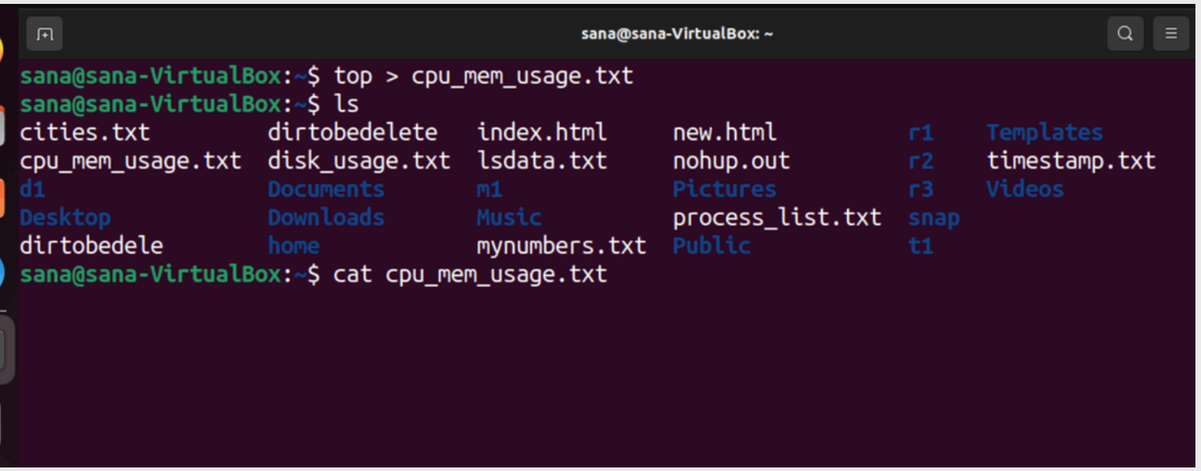
- Find the process ID (PID) of the background process started in the previous step and terminate it.

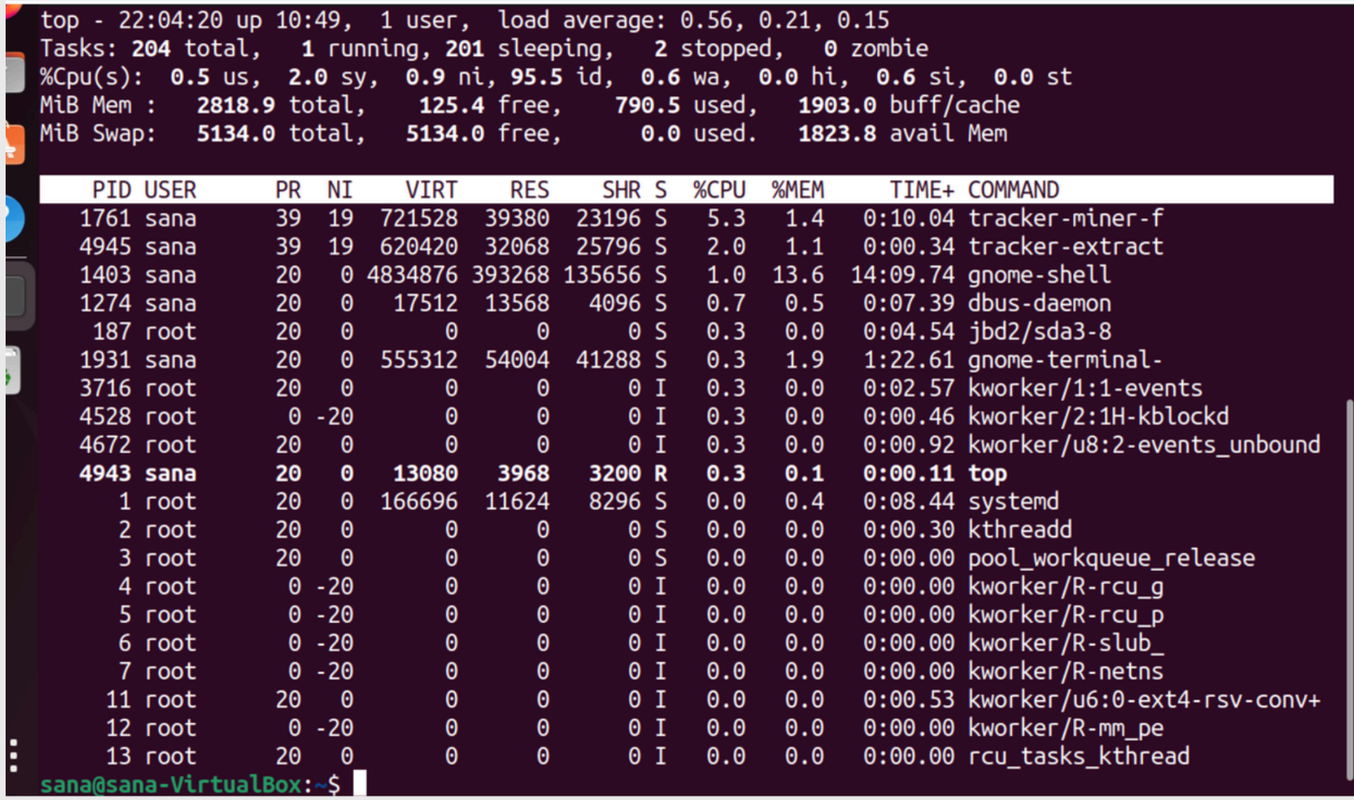




4. CPU and Memory Usage:

- 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.

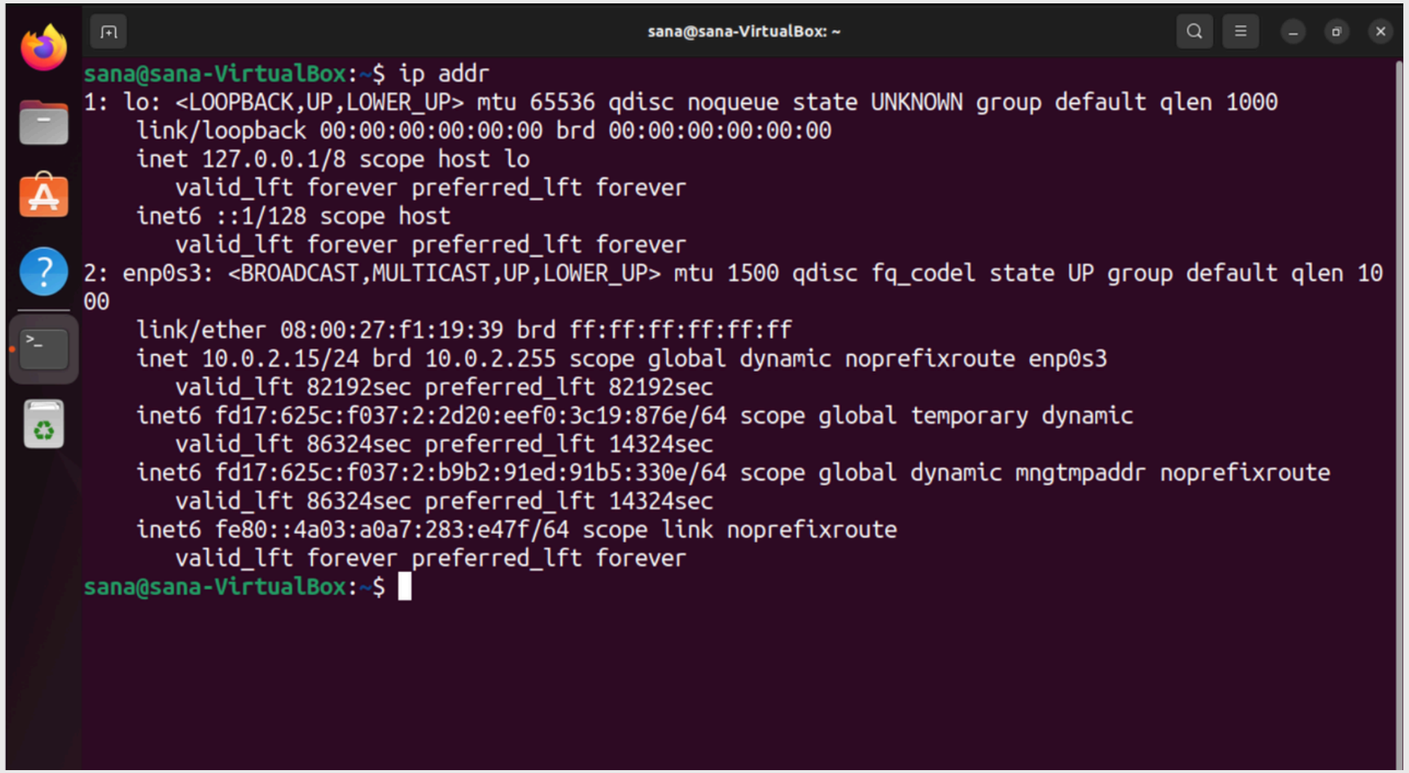




Task 4: Networking

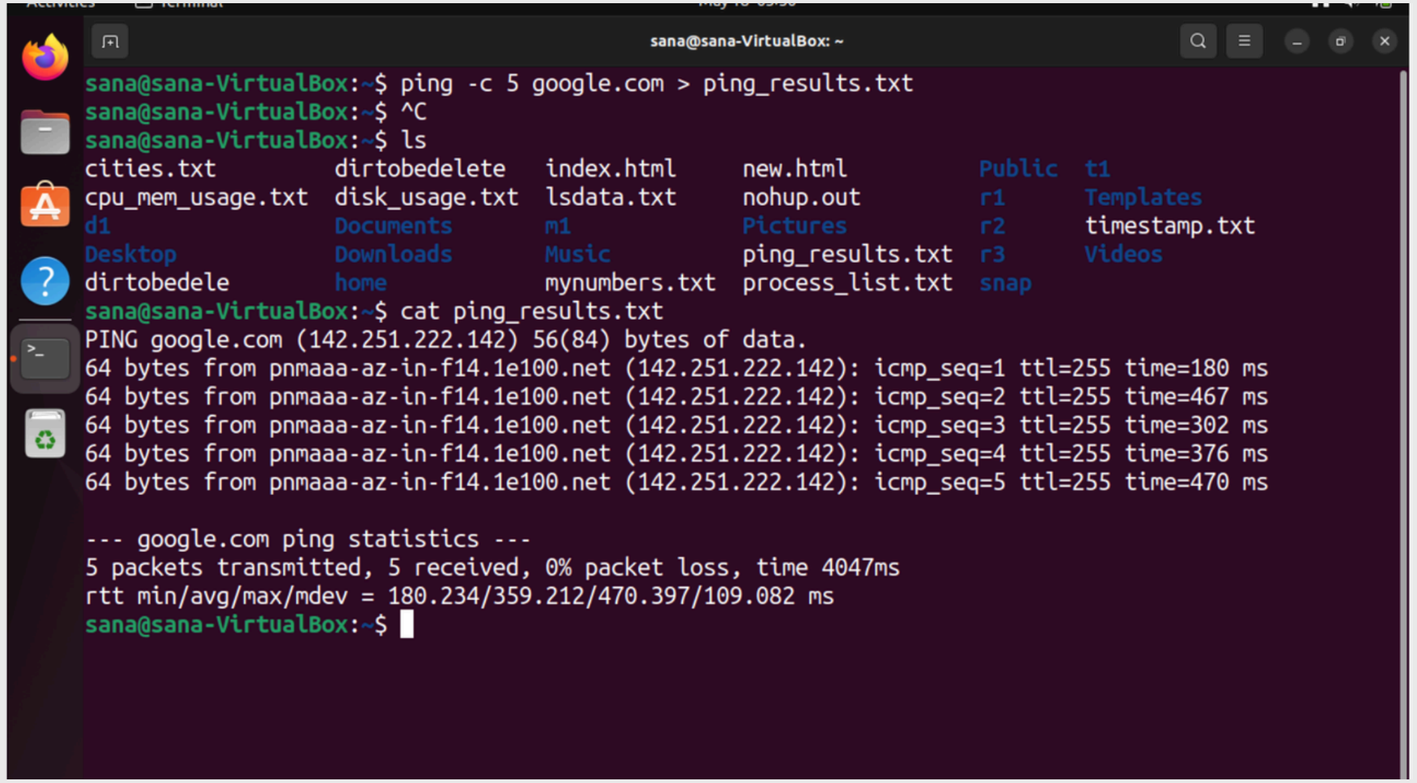
1. Network Configuration:

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



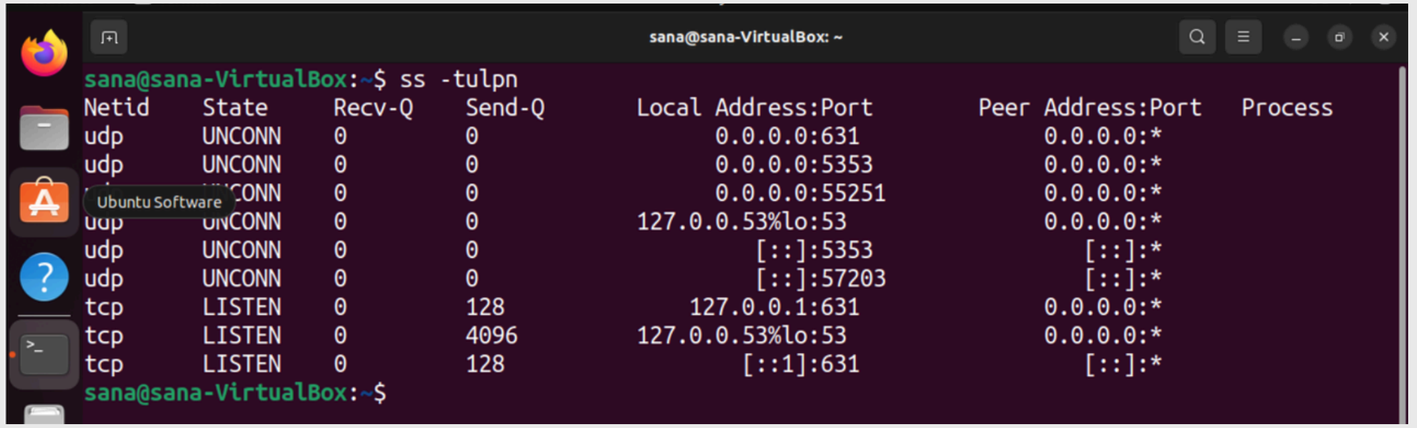
2. Ping Test:

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



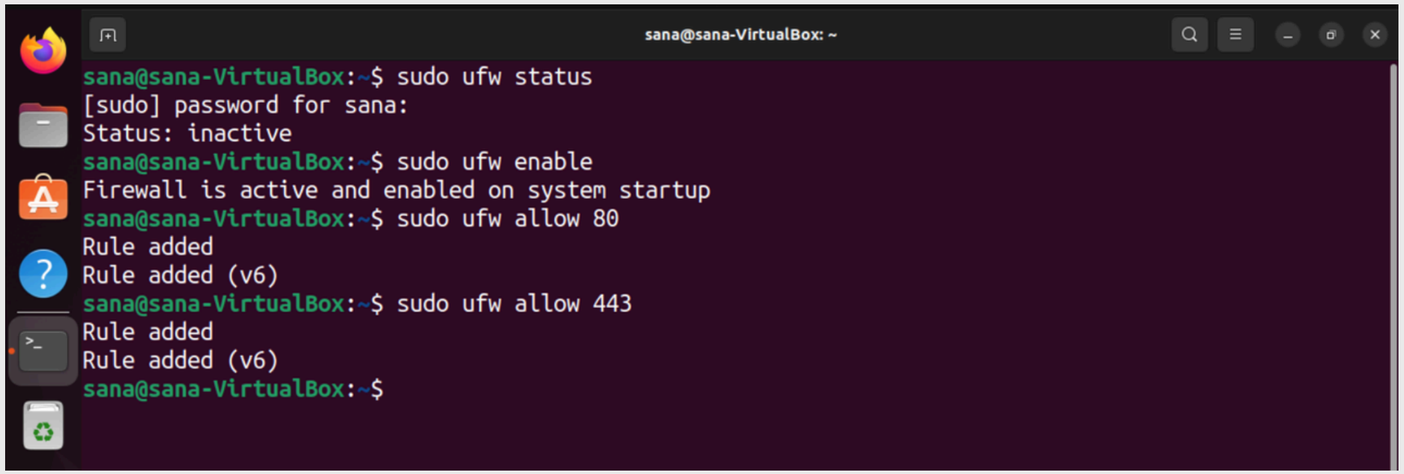
3. Open Ports:

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



4. Firewall Configuration:

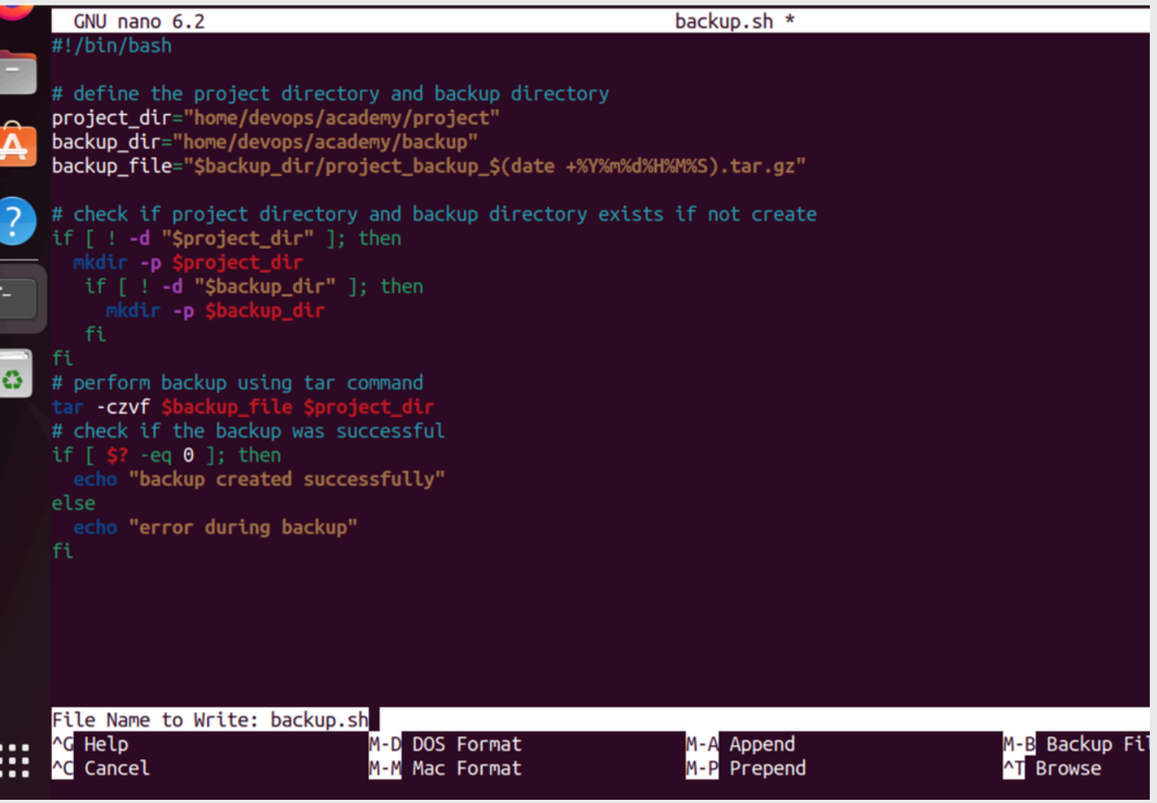
- Check if the ufw (Uncomplicated Firewall) is installed and running. If not, install and enable
- Allow incoming connections on port 80 (HTTP) and 443 (HTTPS).

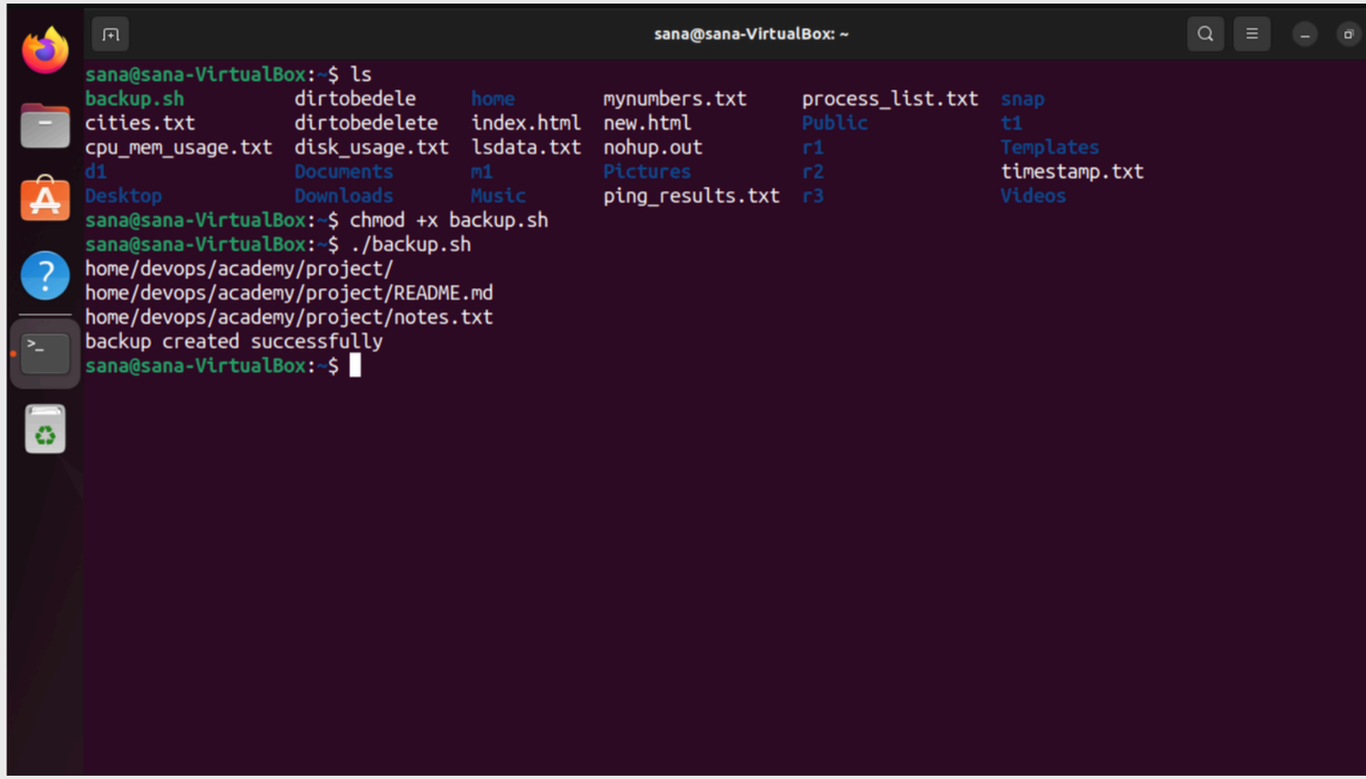


Task 5: Shell Scripting

1. Backup Script:

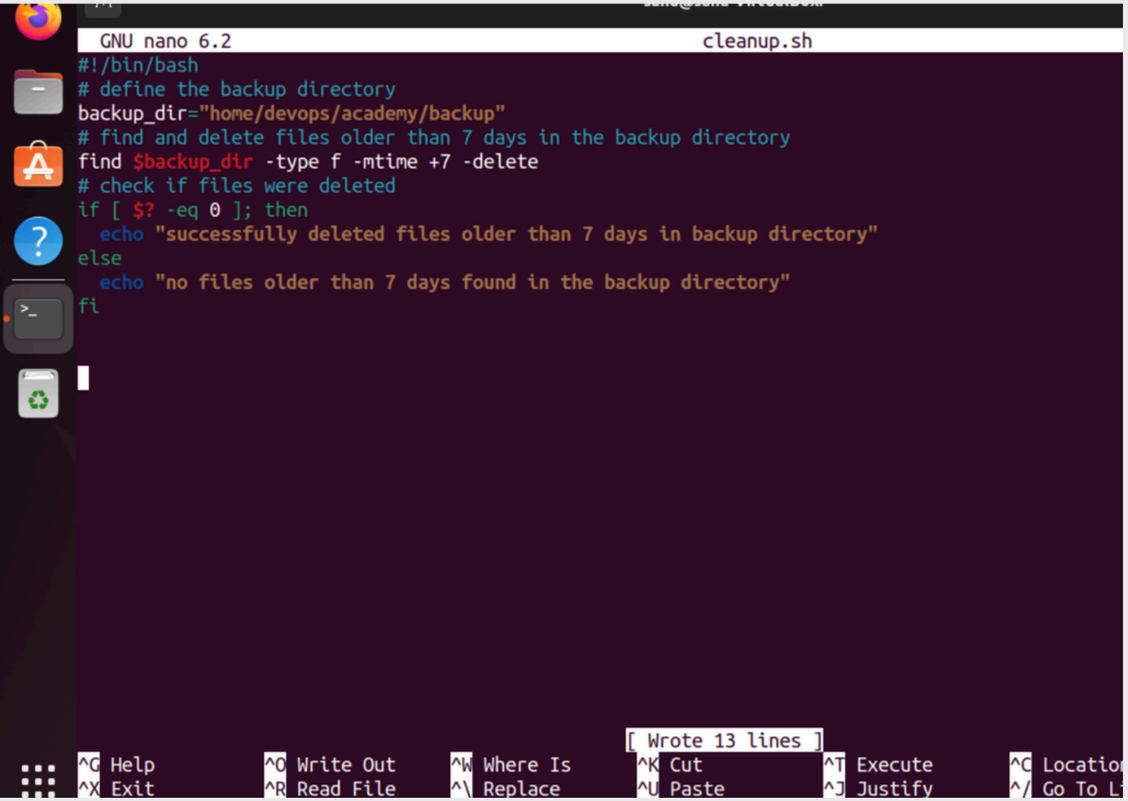
- Write a shell script named backup.sh that compresses the project directory into a tar.gz file and saves it in the /home/devops/academy/backup directory. The script should include error handling to ensure the backup only proceeds if the project directory exists.

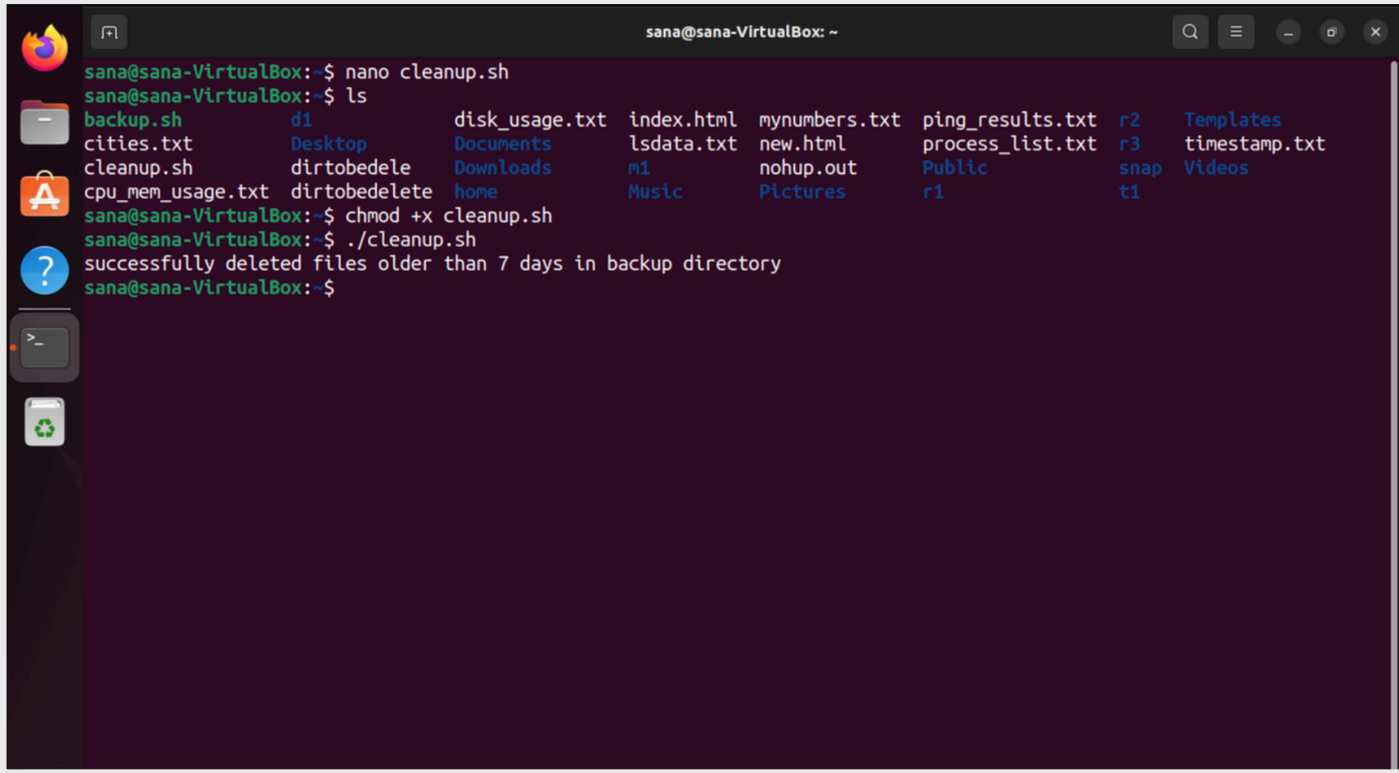


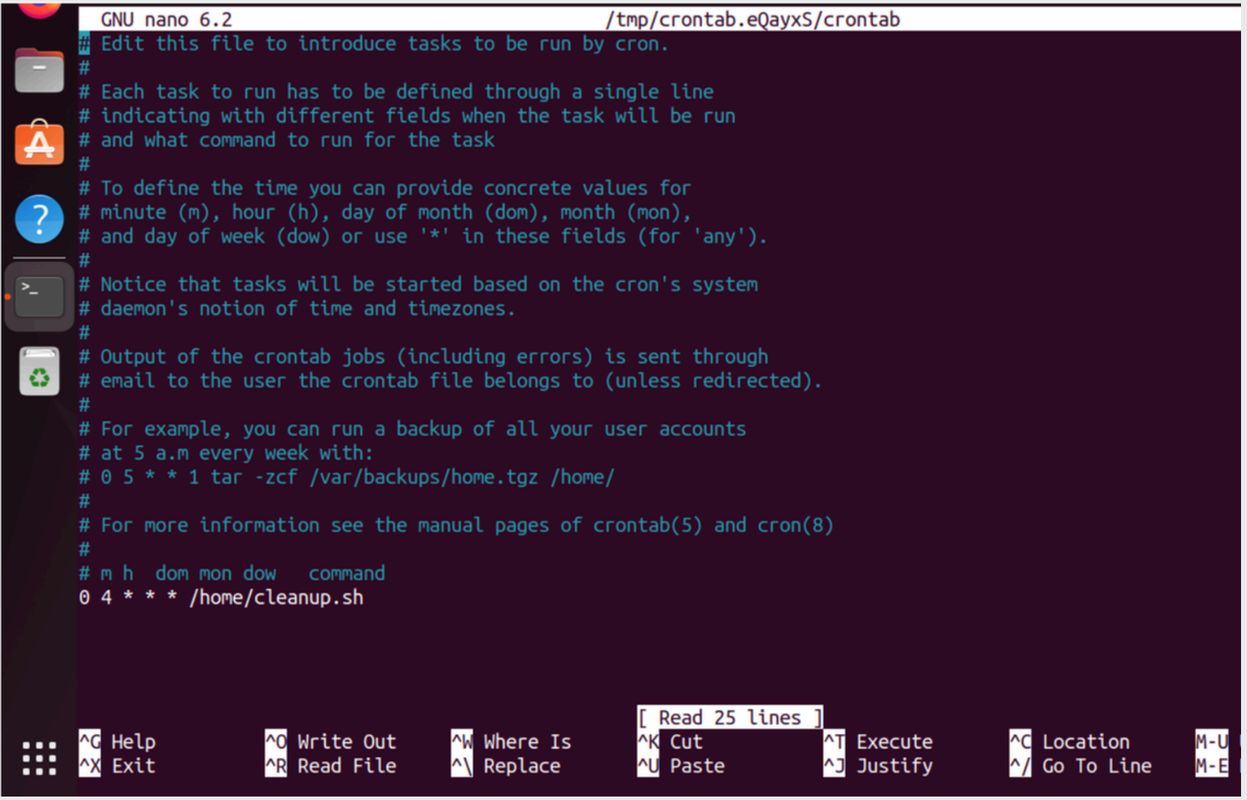


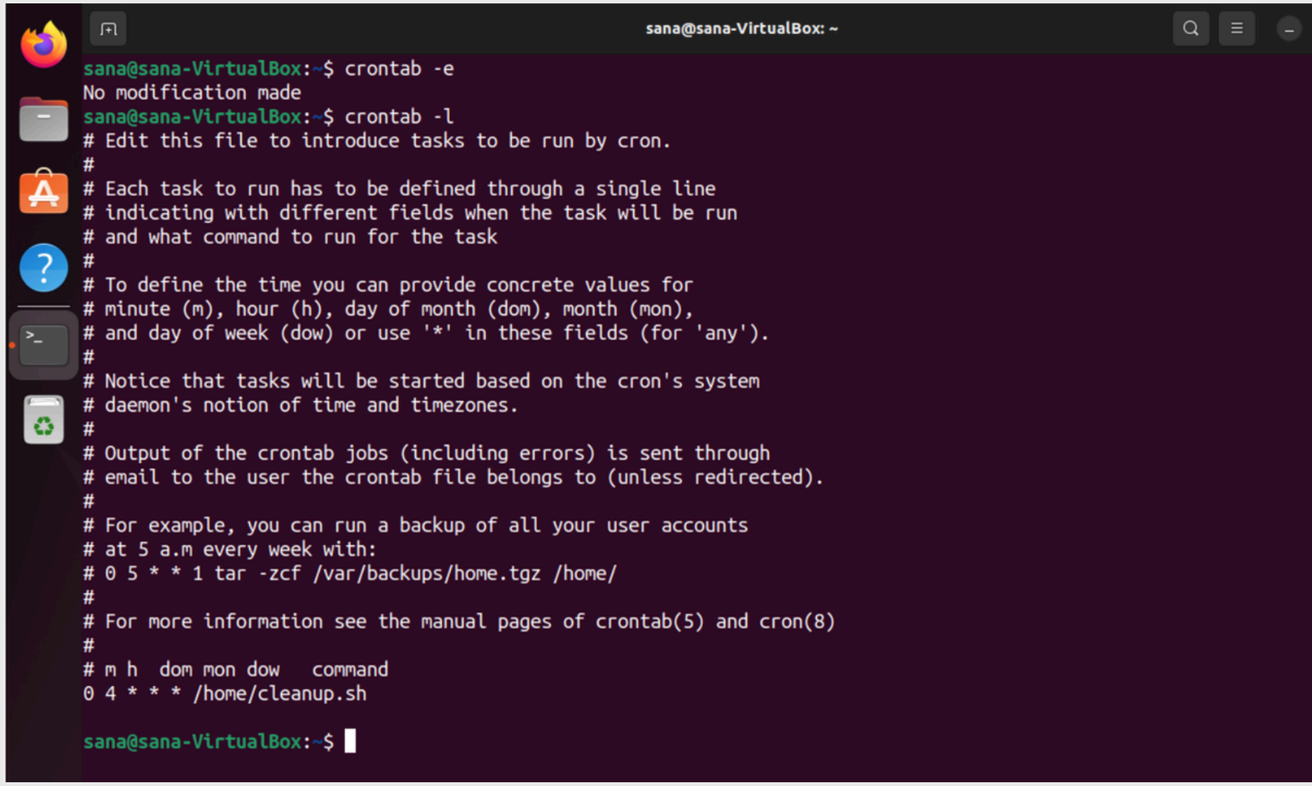
2. Automation Script:

- 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.



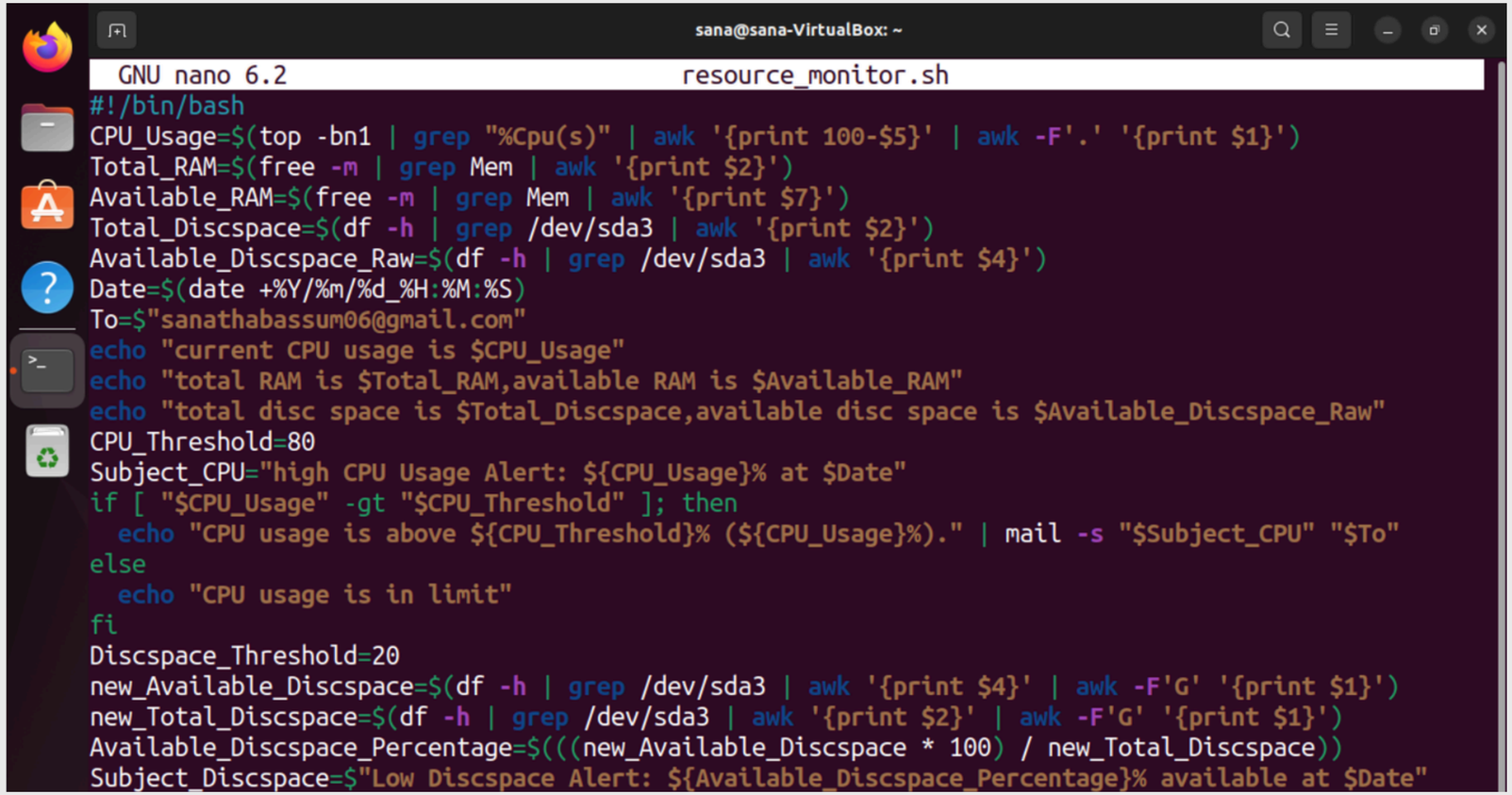



GNU nano 6.2 /tmp/crontab.eQayxS/crontab
Edit this file to introduce tasks to be run by cron.
#
Each task to run has to be defined through a single line
indicating with different fields when the task will be run
and what command to run for the task
#
To define the time you can provide concrete values 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]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E


sana@sana-VirtualBox: ~
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:~$

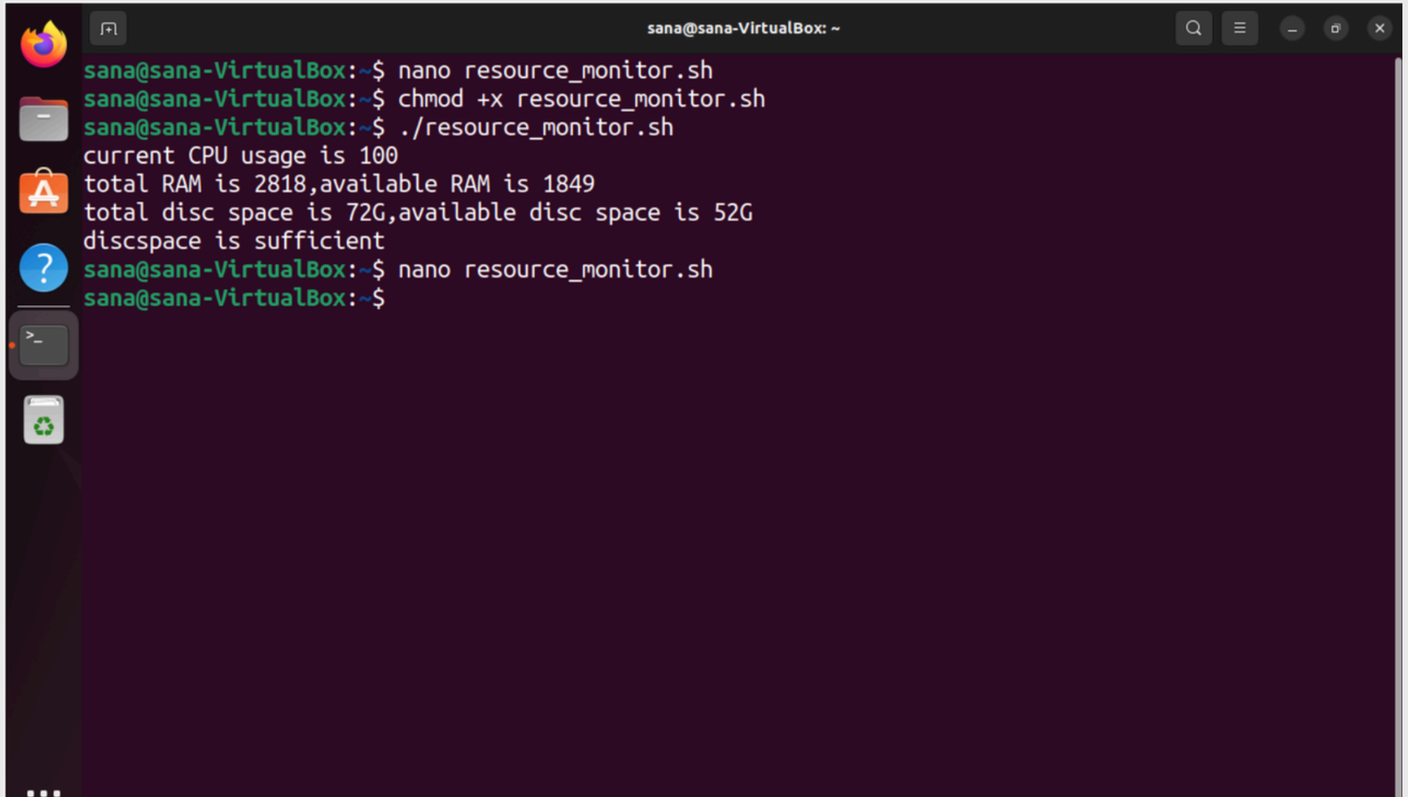
Advanced Level

- **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%..

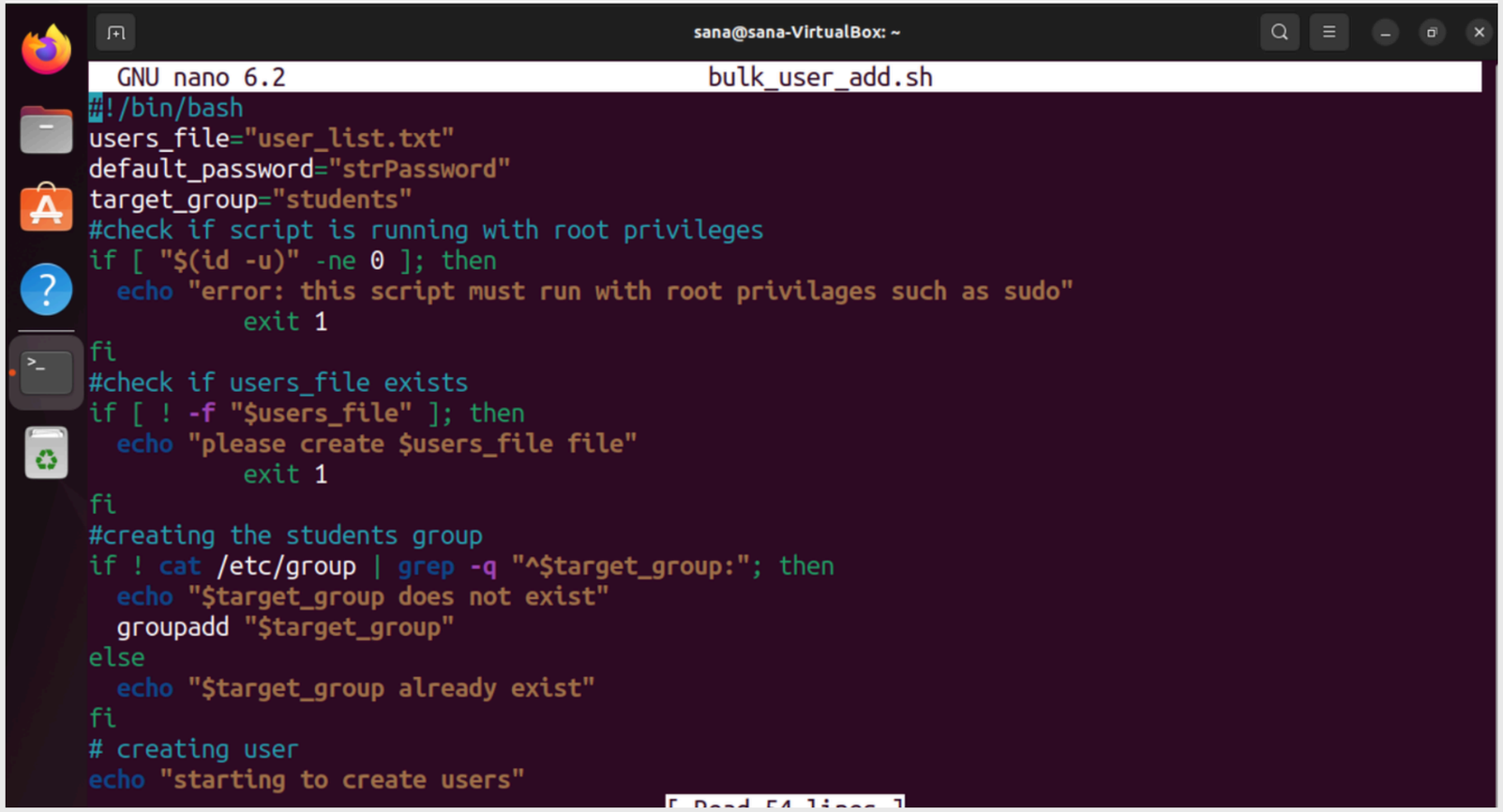


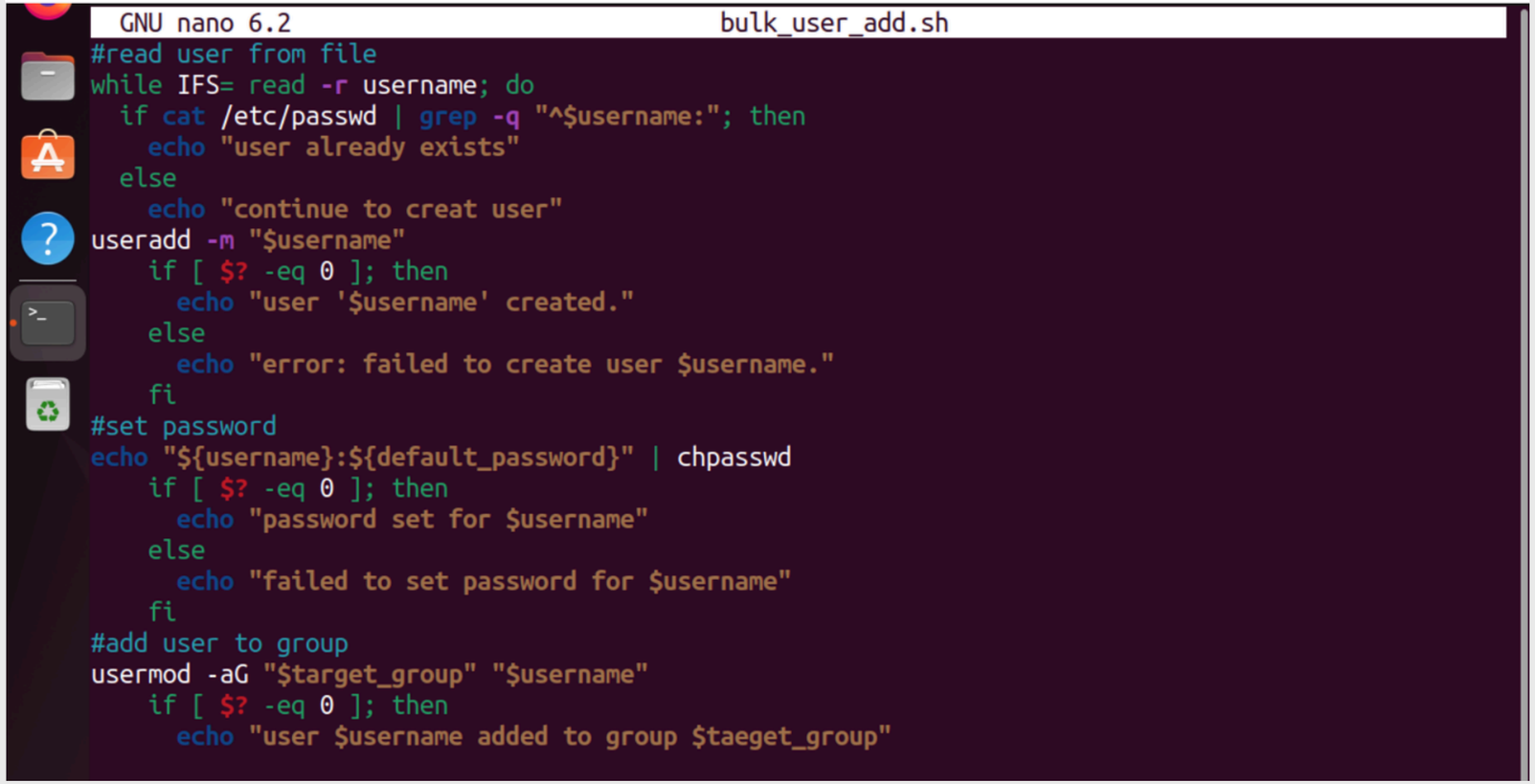
```
if [ "$Available_Discspace_Percentage" -lt "$Discspace_Threshold" ]; then
  echo "available disc space is below ${Discspace_Threshold}% (${Available_Discspace_Percentage}%)>
else
  echo "discspace is sufficient"
fi


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

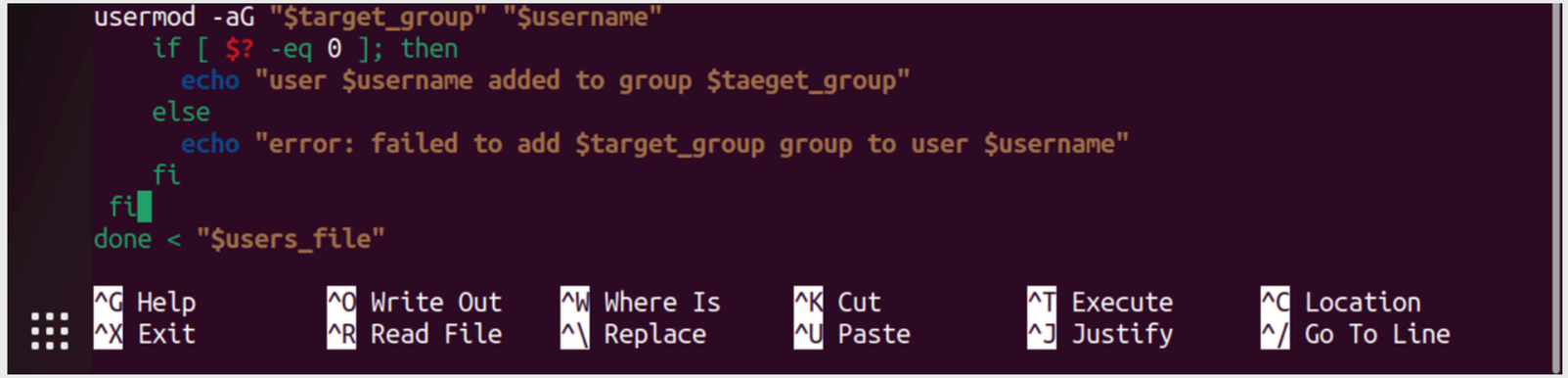
```
if [ "$Available_Discspace_Percentage" -lt "$Discspace_Threshold" ]; then
<age}%)." | mail -s "$Subject_Discspace" "$To"
else
  echo "discspace is sufficient"
fi

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

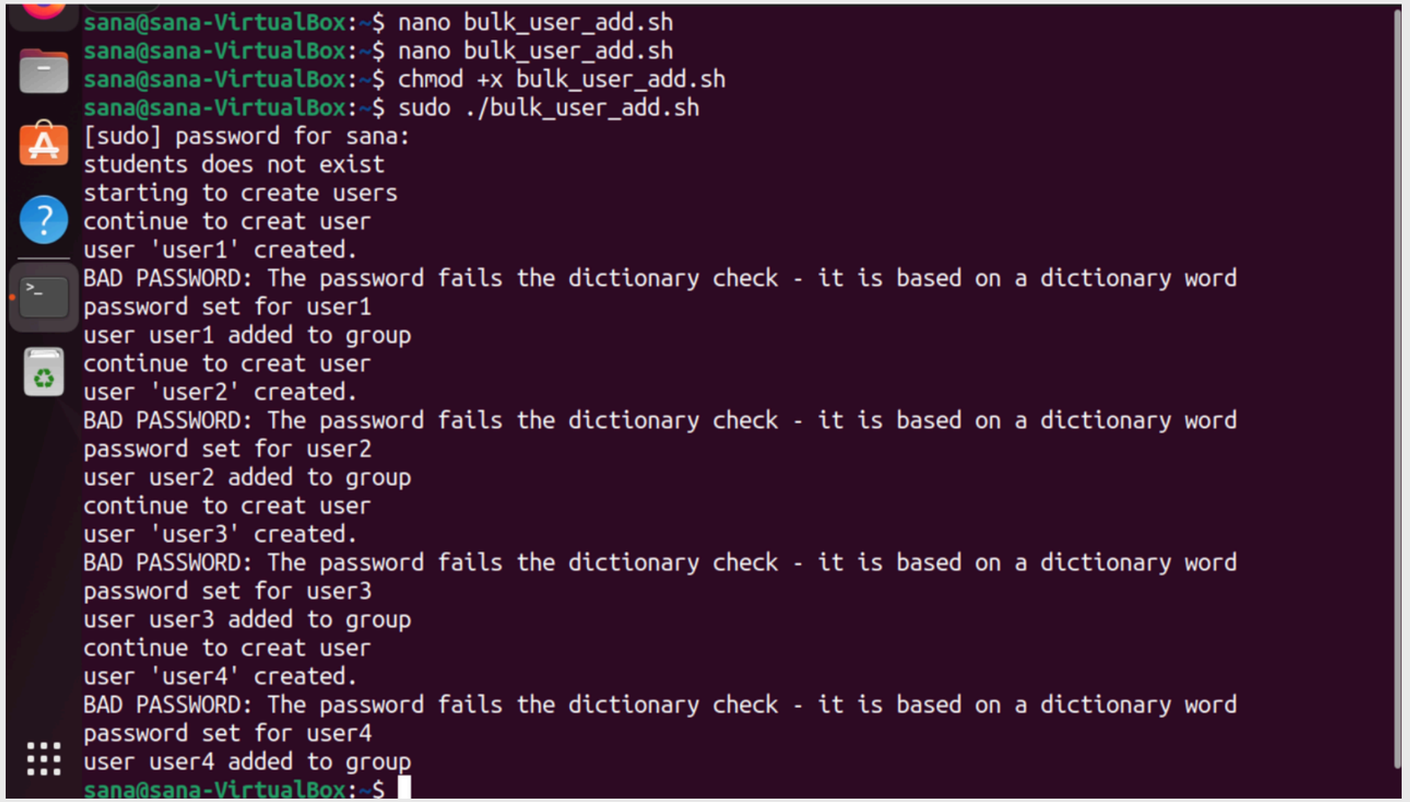


- **User Management Automation:**
  - Write a script named bulk_user_add.sh that:
    1. Reads a list of usernames from a file named user_list.txt.
    2. Creates each user and sets a default password.
    3. Adds each user to a specific group (e.g., students).
    4. Ensures each user has a home directory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




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 $taeget_group"


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