

CLOUD COMPUTING ASSIGNMENT 01

Name: Ayesha Saif

Hawk Id: asaif@hawk.iit.edu

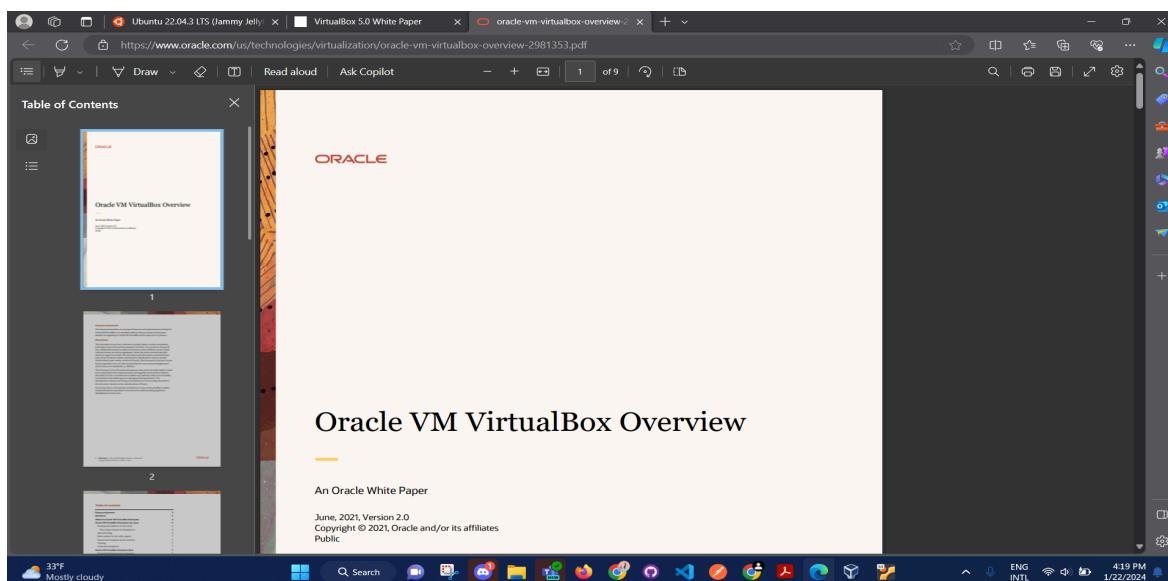
Subject: CS 553 Cloud Computing

Submission To: Prof. Ioan Raicu

CS553 Homework #1

LINUX & COMPUTER SYSTEMS

1. Setup VM, Linux, and basic testing (Screenshot of each step are attached)
 - a. Read Oracle VirtualBox White Paper

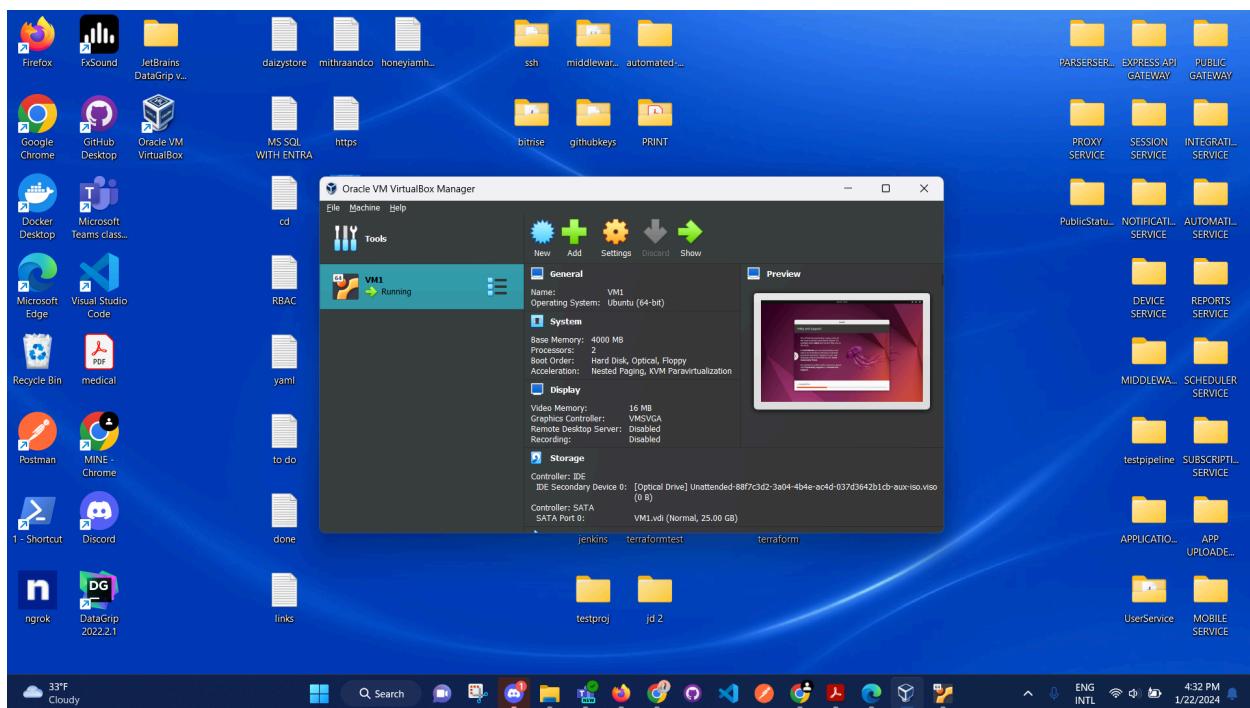


- b. Download Oracle VirtualBox 7.0

CLOUD COMPUTING ASSIGNMENT 01

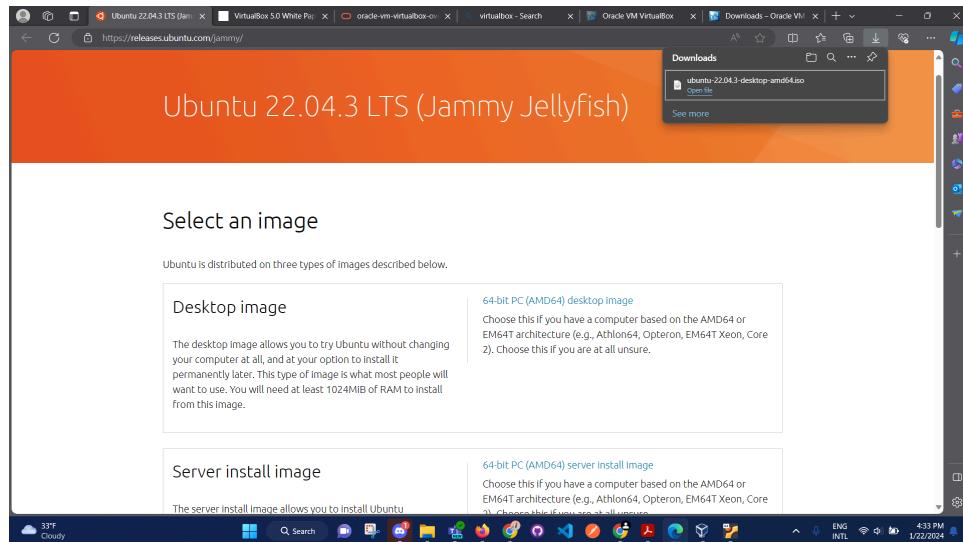


c. Install VirtualBox 7.0

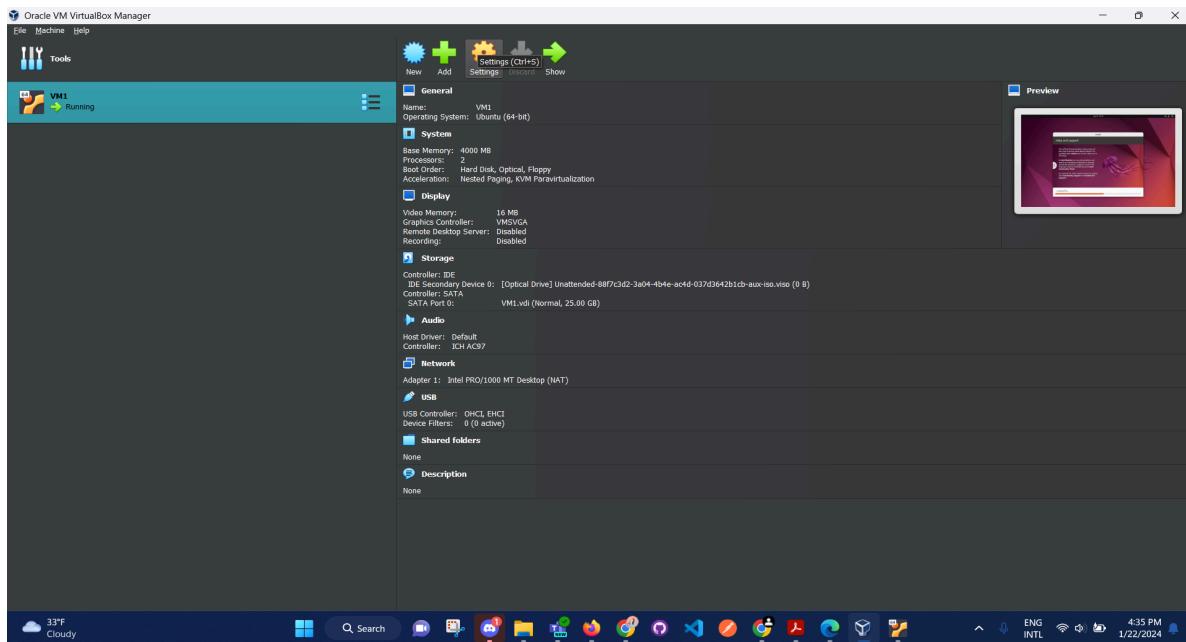


d. Download Ubuntu 22.04 Linux ISO Image

CLOUD COMPUTING ASSIGNMENT 01

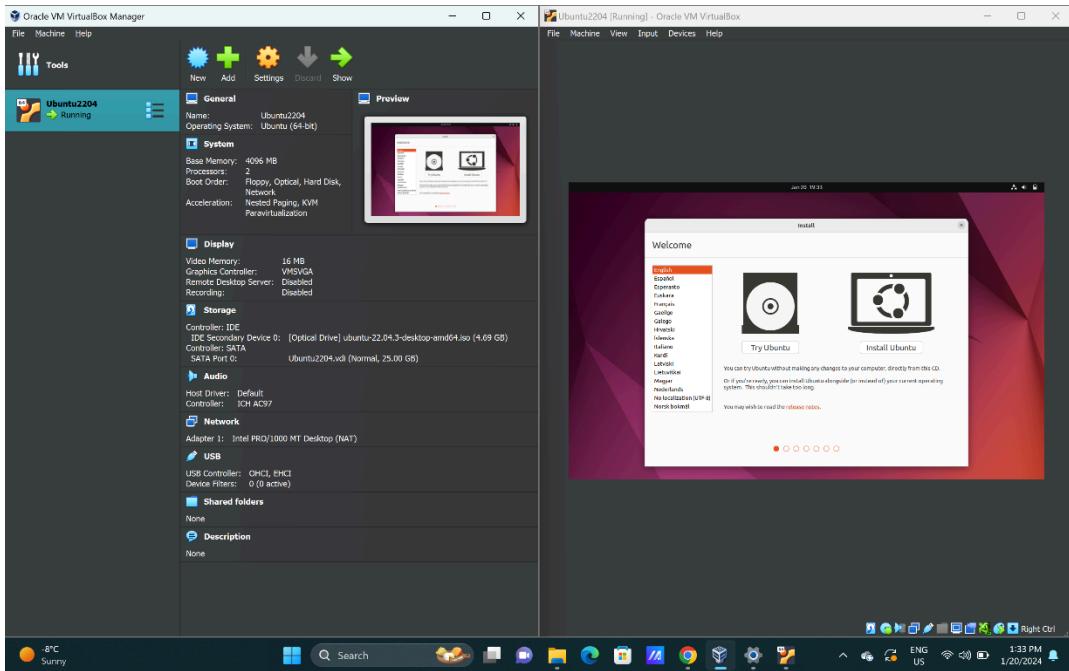


- e. Create Virtual Machine (VM), to support Linux, Ubuntu, 64-bit, 4GB RAM, Virtual Disk 25GB, VDI image, dynamically allocated, 2-core, and a network interface (1GbE or Wifi) with Bridged Adapter

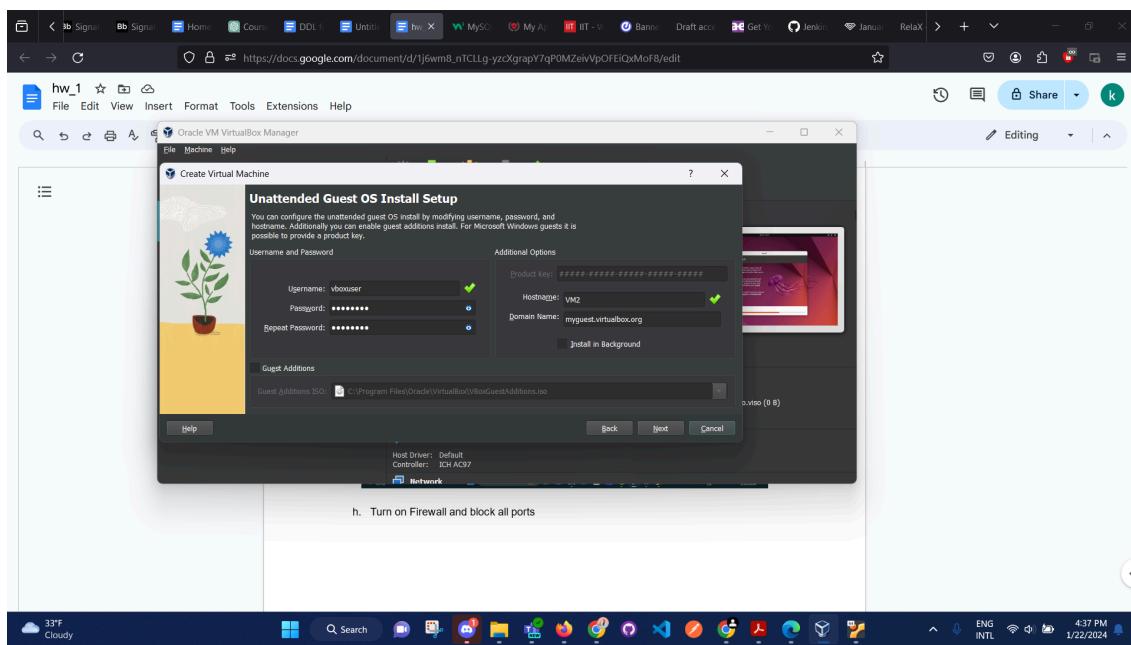


- f. Install Linux from the ISO image

CLOUD COMPUTING ASSIGNMENT 01

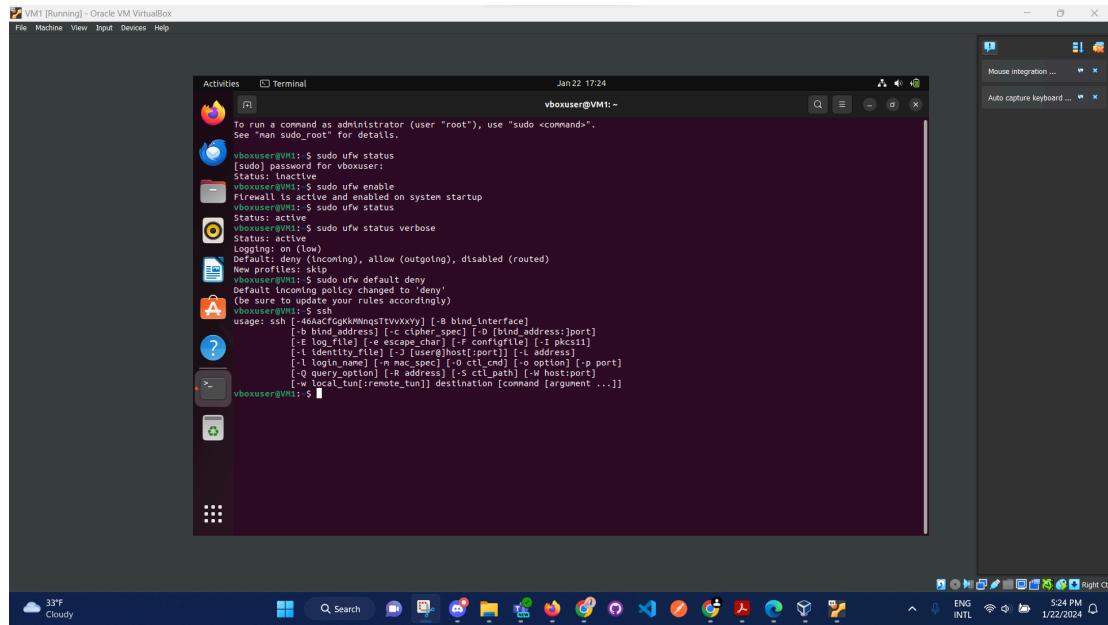


g. Create a user id and password



h. Turn on Firewall and block all ports

CLOUD COMPUTING ASSIGNMENT 01



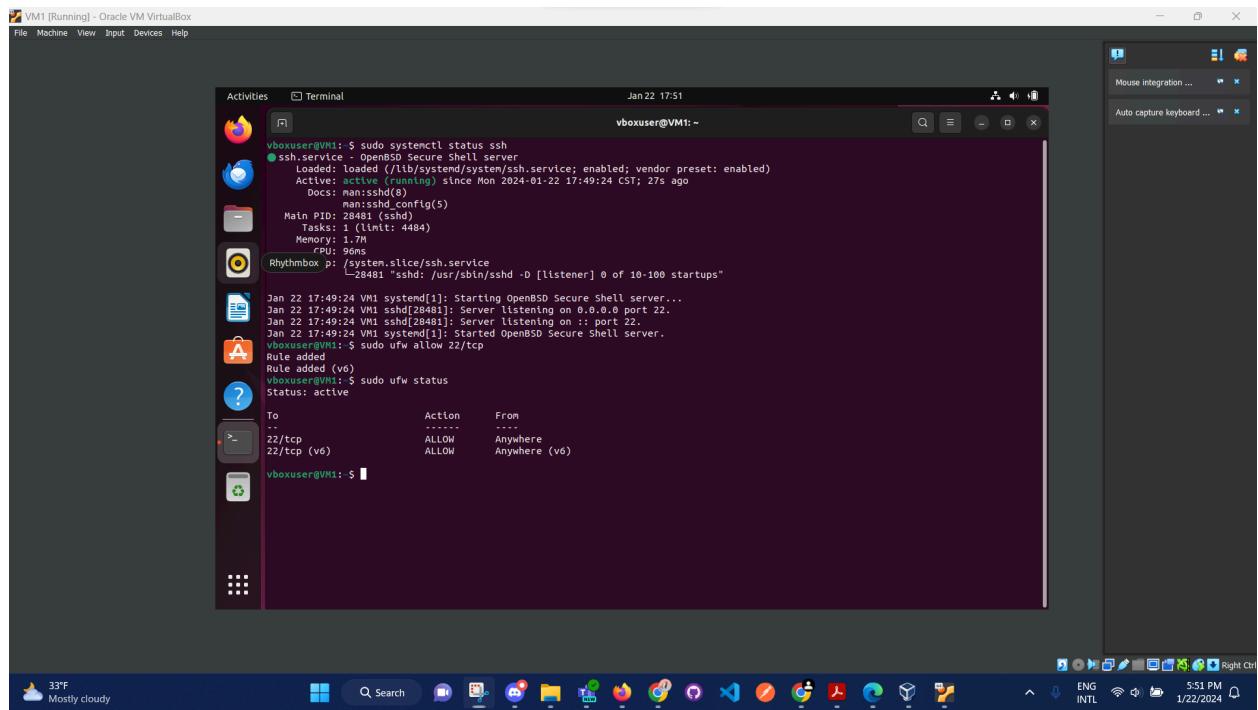
A screenshot of a Linux desktop environment, likely Ubuntu, running in Oracle VM VirtualBox. The desktop has a dark theme. A terminal window is open in the foreground, showing the user's session:

```
vboxuser@VM1: ~
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  

vboxuser@VM1: $ sudo ufw status  
[sudo] password for vboxuser:  
Status: active  
Firewall is active and enabled on system startup  
vboxuser@VM1: $ sudo ufw status  
Status: active  
vboxuser@VM1: $ sudo ufw status verbose  
Status: active  
Logging: low  
Default: deny (incoming), allow (outgoing), disabled (routed)  
New profiles: skip  
vboxuser@VM1: $ sudo ufw default deny  
Default policy successfully changed to "deny"  
(be sure to update your rules accordingly)
vboxuser@VM1: $ ssh
usage: ssh [-AaCcfGgKkNnqTtVvXxYy] [-B bind_interface]
           [-c cipher_spec] [-D [bind_address]:port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-l identity_file] [-o [user@]host[:port]] [-L address]
           [-i login_name] [-m mac_spec] [-O c11_cmd] [-o option] [-P port]
           [-Q query_file] [-R remote_port] [-S socket_path] [-t timeout]
           [-W local_tun|remote_tun]] destination [command [argument ...]]
```

The desktop bar at the bottom shows various application icons and system status indicators.

- Enable SSH access to your new Linux installation; open SSH port in firewall



A screenshot of a Linux desktop environment, likely Ubuntu, running in Oracle VM VirtualBox. The desktop has a dark theme. A terminal window is open in the foreground, showing the user's session:

```
vboxuser@VM1: ~
ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-01-22 17:49:24 CST; 27s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
     Main PID: 1 (sshd)
        Tasks: 1 (limit: 4484)
      Memory: 1.7M
         CPU: 96ms
Rhythmbox p: /system.slice/ssh.service
          └─28481 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Jan 22 17:49:24 VM1 systemd[1]: Starting OpenBSD Secure Shell server...
Jan 22 17:49:24 VM1 sshd[28481]: Server listening on 0.0.0.0 port 22.
Jan 22 17:49:24 VM1 sshd[28481]: Server listening on :: port 22.
Jan 22 17:49:24 VM1 systemd[1]: Started OpenBSD Secure Shell server.
vboxuser@VM1: $ sudo ufw allow 22/tcp
vboxuser@VM1: $ sudo ufw status
Status: active

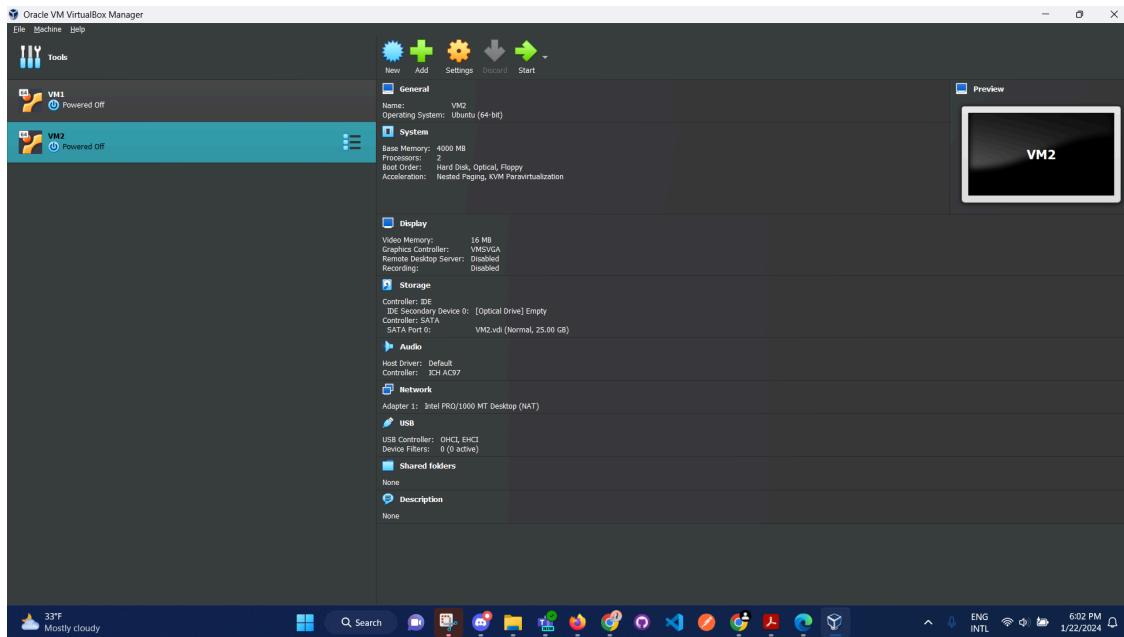
To                         Action      From
                         ...          ...
22/tcp                     ALLOW      Anywhere
22/tcp (v6)                ALLOW      Anywhere (v6)

vboxuser@VM1: $
```

The desktop bar at the bottom shows various application icons and system status indicators.

- Repeat steps 5 through 9, and create another VM with the same specifications as the first one.

CLOUD COMPUTING ASSIGNMENT 01



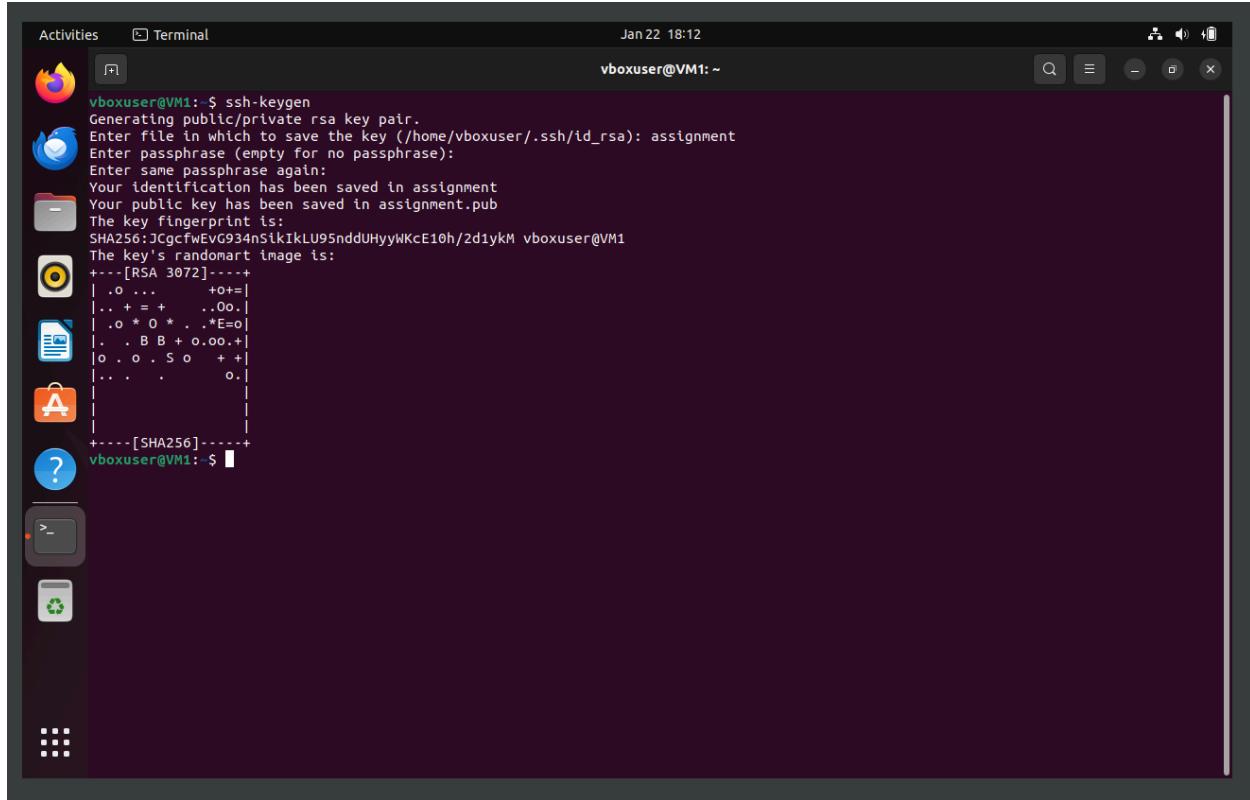
- k. Create private/public keys and install them properly in both of your new VMs

A screenshot of a terminal window titled "vboxuser@VM1: ~" running on VM1. The user has run the command "ssh-keygen" and is generating a RSA key pair. The terminal output shows:

```
vboxuser@VM1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vboxuser/.ssh/id_rsa): assignment
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in assignment
Your public key has been saved in assignment.pub
The key fingerprint is:
SHA256:RWcc9W/G4gg/34PgpPmtDkxPLP4zGVGogcIaecbs33A vboxuser@VM1
The key's randomart image is:
+---[RSA 3072]---+
|   = . . =o. |
| o B . o +. . |
| * . + . . |
| . . . E. . .. |
| . So. o. . =|
| .+.+* o + |
| +=.B o |
| oo=.+ o |
| o+=. o |
+---[SHA256]---+
vboxuser@VM1:~$
```

The terminal window is part of a desktop environment with a dark theme, showing a dock of icons at the bottom.

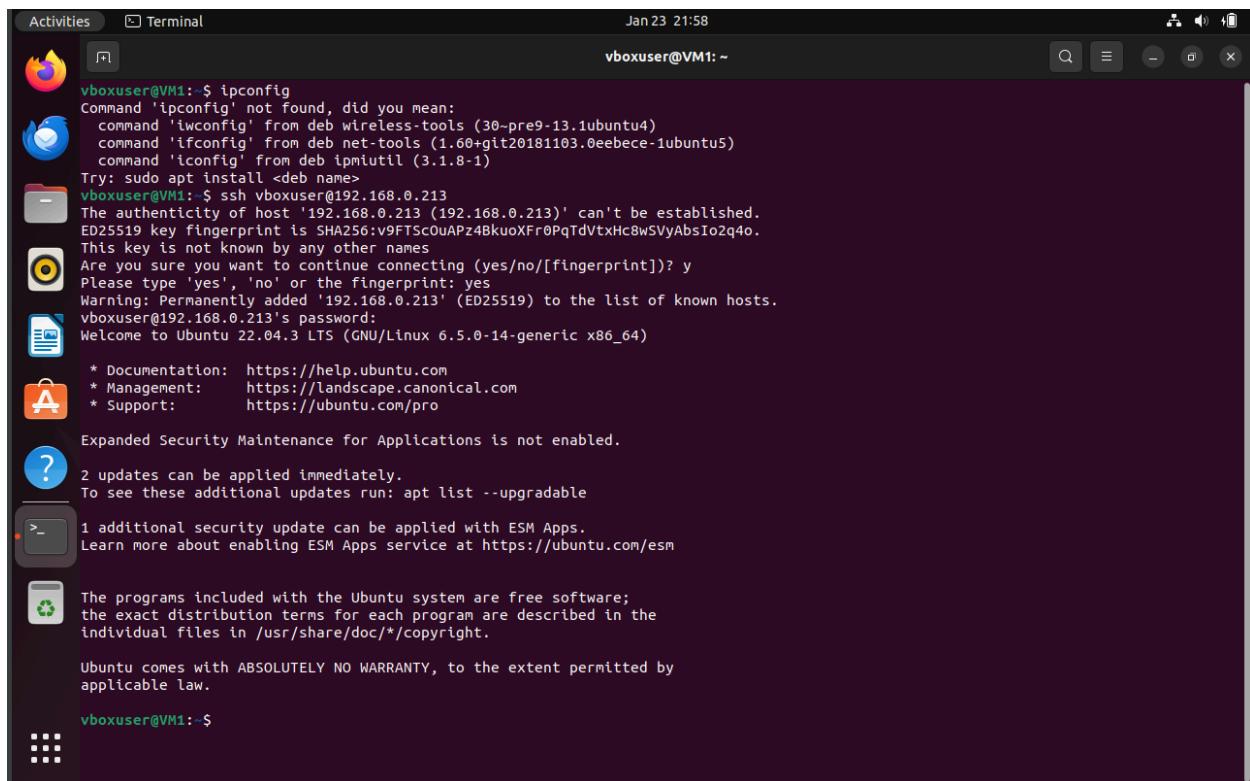
CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Ubuntu desktop environment. The terminal window shows the command `ssh-keygen` being run, generating a RSA key pair. The key fingerprint is displayed as a long string of characters.

```
vboxuser@VM1: $ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vboxuser/.ssh/id_rsa): assignment
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in assignment
Your public key has been saved in assignment.pub
The key fingerprint is:
SHA256:JGcgfwEvg934nStk1kLU95nddUHyyKCe1oh/2d1ykM vboxuser@VM1
The key's randomart image is:
+---[RSA 3072]---+
| . . . . ++= |
| .. + = + ..Oo. |
| .o * O * . .*E=o |
| . . B B + o.oo.+ |
| o . o . S o + + |
| . . . . . o. |
+---[SHA256]---+
vboxuser@VM1: $
```

I. Test that you can connect remotely to your keys, from one VM to the other VM



A screenshot of a Ubuntu desktop environment. The terminal window shows the user attempting to connect to another host via SSH. It lists alternative commands for 'ipconfig' and then prompts for a password. The user then sees a standard Ubuntu welcome message and security information.

```
vboxuser@VM1:~$ ipconfig
Command 'ipconfig' not found, did you mean:
  command 'iwconfig' from deb wireless-tools (30-pre9-13.1ubuntu4)
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntus)
  command 'iconfig' from deb ipmiutil (3.1.8-1)
Try: sudo apt install <deb name>
vboxuser@VM1:~$ ssh vboxuser@192.168.0.213
The authenticity of host '192.168.0.213 (192.168.0.213)' can't be established.
ED25519 key fingerprint is SHA256:v9FTSc0uAPz4BkuoXFr0PqTdVtxHc8wSVyAbsIo2q4o.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.0.213' (ED25519) to the list of known hosts.
vboxuser@192.168.0.213's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-14-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

vboxuser@VM1:~$
```

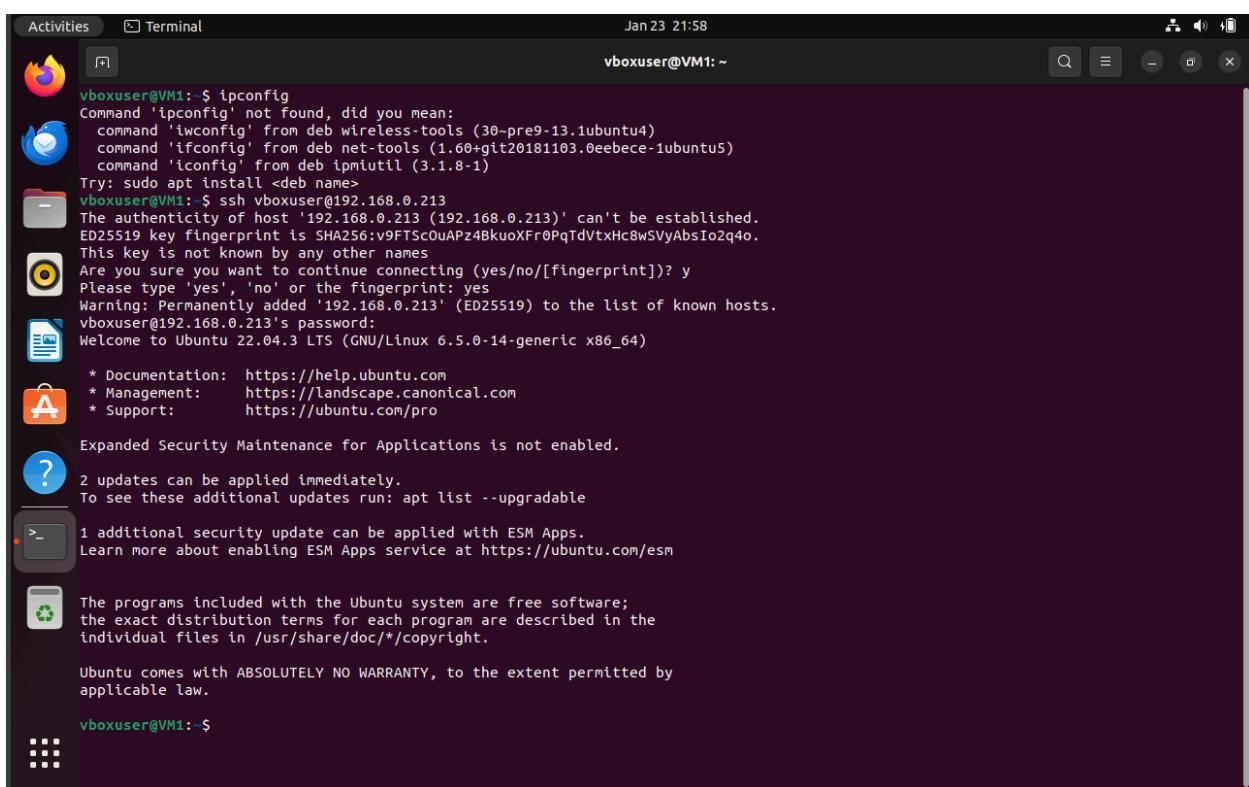
CLOUD COMPUTING ASSIGNMENT 01

2. Show an example of using the following commands; take screen shots of your commands; clear screen between each command; explain in own words what these commands do:

- 1) ssh (Secure Shell):

Description: Connects to a remote server securely over a network.

Example Usage: ssh username@remote_server



A screenshot of a Linux desktop environment showing a terminal window titled "Terminal". The terminal window has a dark background and displays the following text:

```
vboxuser@VM1:~$ ipconfig
Command 'ipconfig' not found, did you mean:
  command 'iwconfig' from deb wireless-tools (30-pre9-13.1ubuntu4)
  command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
  command 'iconfig' from deb ipmiutil (3.1.8-1)
Try: sudo apt install <deb name>
vboxuser@VM1:~$ ssh vboxuser@192.168.0.213
The authenticity of host '192.168.0.213 (192.168.0.213)' can't be established.
ED25519 key fingerprint is SHA256:v9FTScOUAPz4BkuoXFr0PqTdVtxHc8wSVyAbsI02q4o.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.0.213' (ED25519) to the list of known hosts.
vboxuser@192.168.0.213's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.5.0-14-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

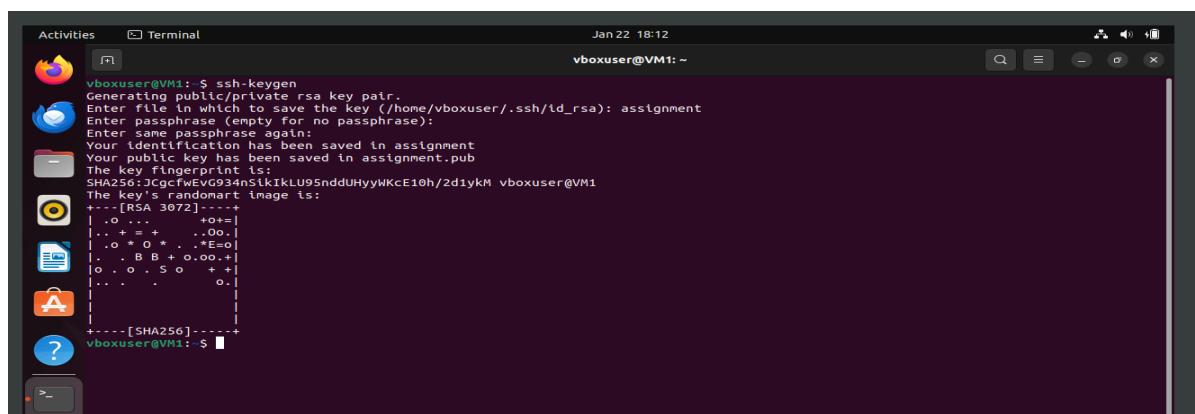
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

vboxuser@VM1:~$
```

- 2) ssh-keygen:

Description: Generates SSH key pairs for secure authentication.

Example Usage: ssh-keygen -t rsa



A screenshot of a Linux desktop environment showing a terminal window titled "Terminal". The terminal window has a dark background and displays the following text:

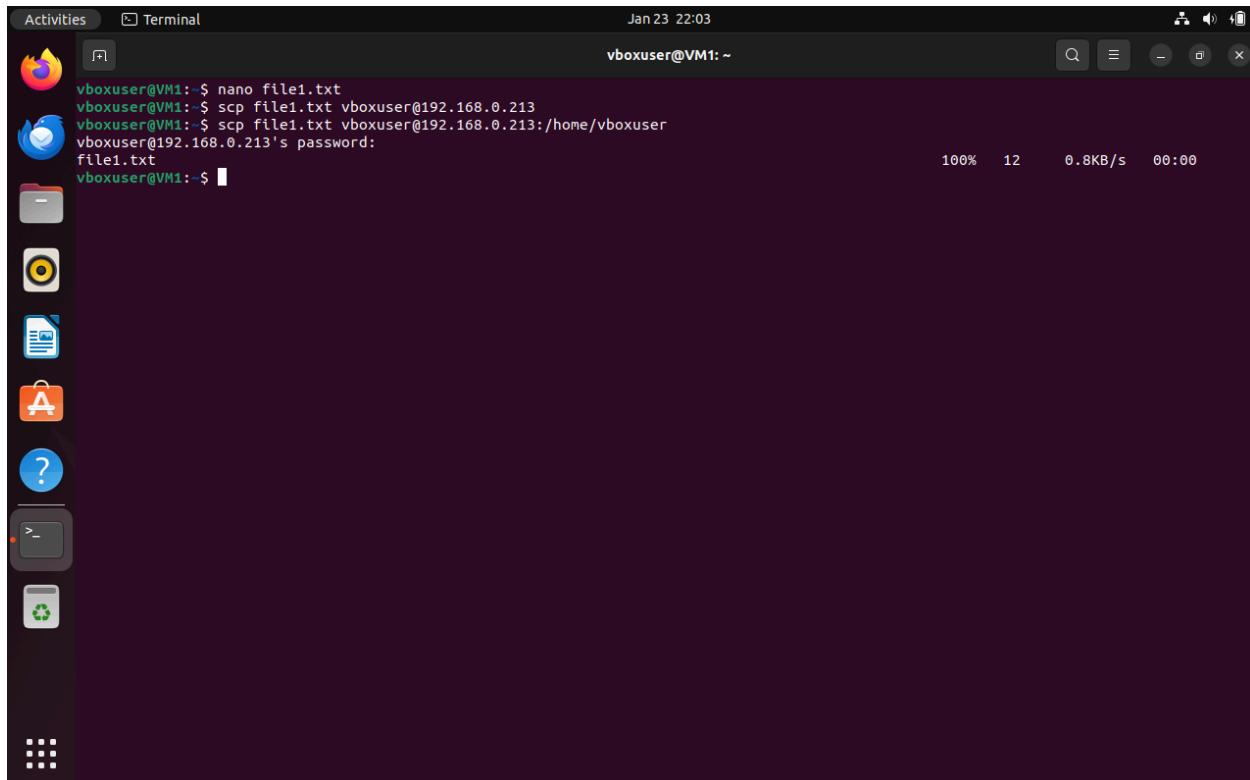
```
vboxuser@VM1:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vboxuser/.ssh/id_rsa): assignment
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in assignment
Your public key has been saved in assignment.pub
The key fingerprint is:
SHA256:3CgcfTwEvG934nStk1kLU95nddUHyyWKcE10h/2diyK M vboxuser@VM1
-----[RSA 3072]-----
-----[SHA256]-----
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

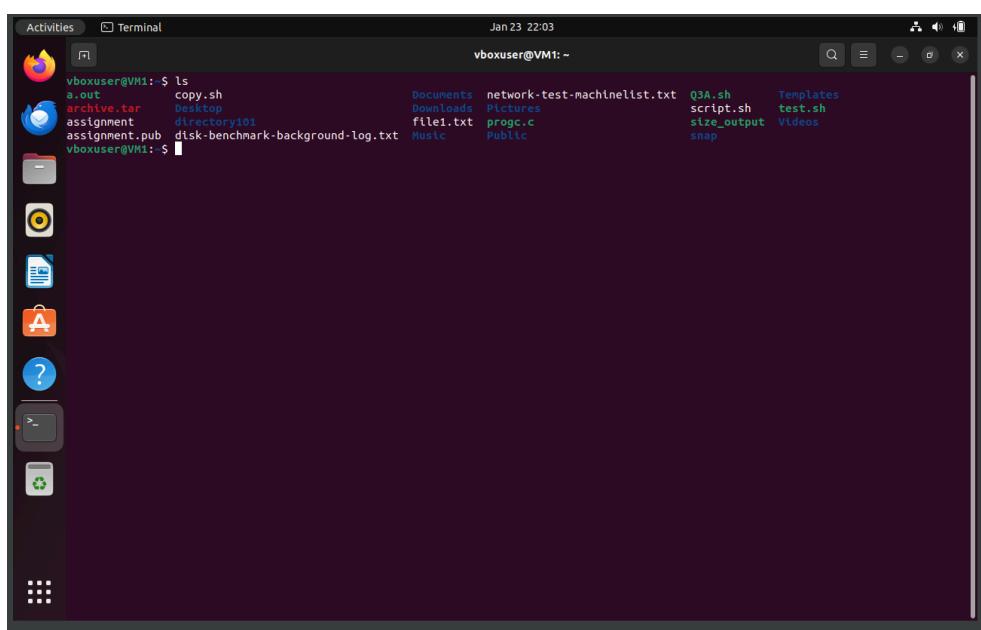
3) scp (Secure Copy):

Description: Copies files securely between hosts on a network.

Example Usage: `scp local_file.txt username@remote_server:/path/to/destination`



```
vboxuser@VM1:~$ nano file1.txt
vboxuser@VM1:~$ scp file1.txt vboxuser@192.168.0.213:
vboxuser@VM1:~$ scp file1.txt vboxuser@192.168.0.213:/home/vboxuser
vboxuser@192.168.0.213's password:
file1.txt
100%   12      0.8KB/s  00:00
vboxuser@VM1:~$
```



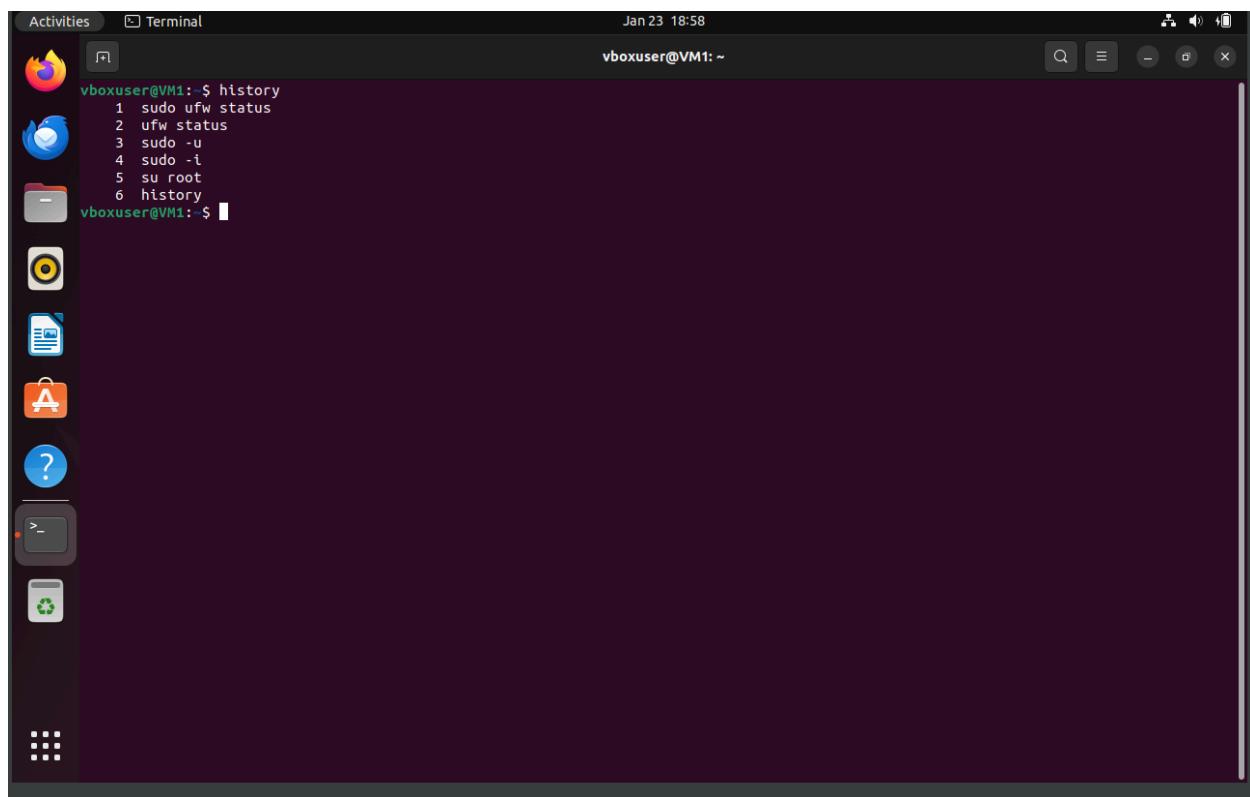
```
vboxuser@VM1:~$ ls
a.out          copy.sh          Documents    network-test-machinelist.txt  Q3A.sh      Templates
archive.tar    Desktop          Downloads    Pictures           proc.c       script.sh  test.sh
assignment     directory101     file1.txt    Public            size_output  Videos
assignment.pub disk-benchmark-background-log.txt Music        snap
```

CLOUD COMPUTING ASSIGNMENT 01

4) history:

Description: Displays a list of previously executed commands.

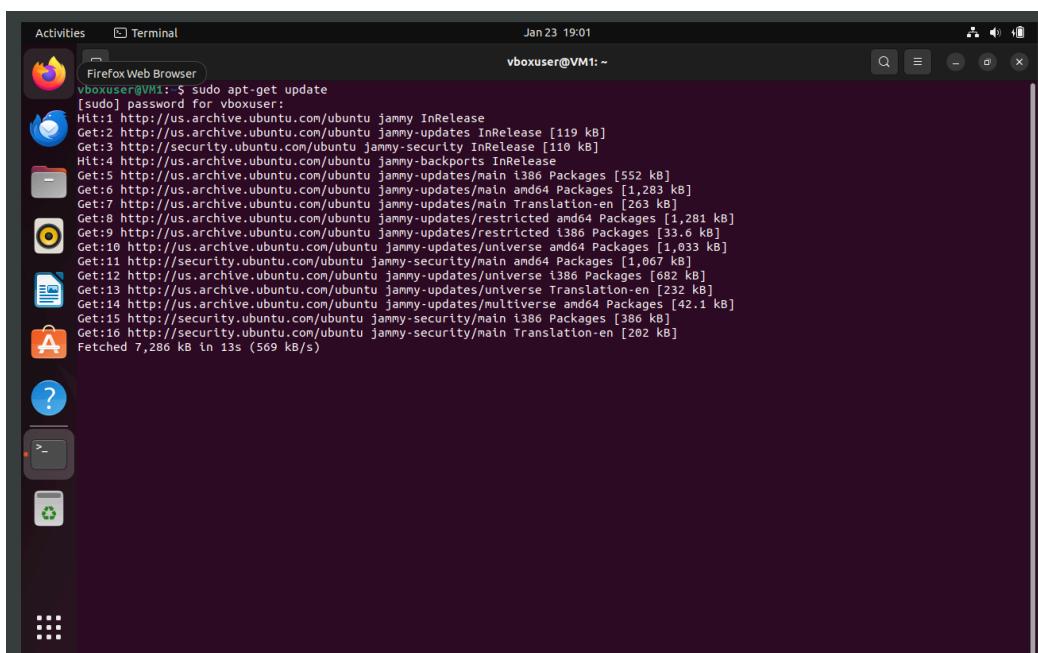
Example Usage: history



5) sudo (Superuser Do):

Description: Executes a command with elevated privileges (as a superuser).

Example Usage: sudo command_to_execute

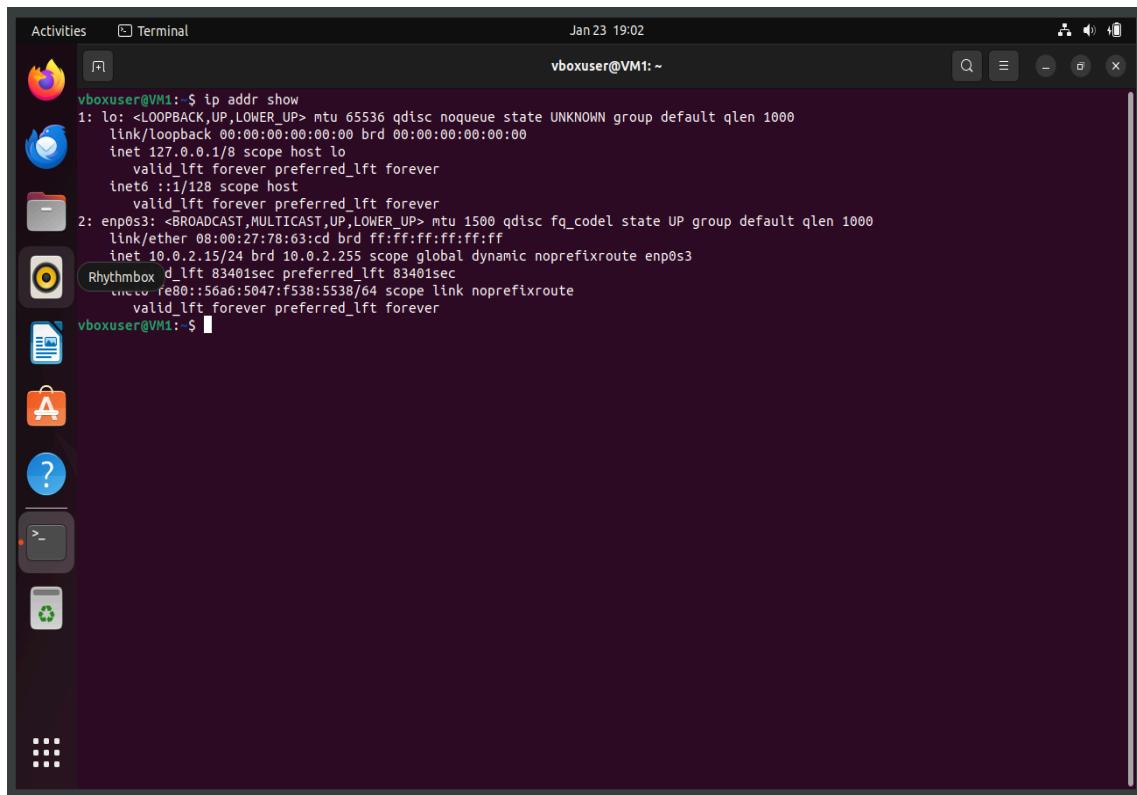


CLOUD COMPUTING ASSIGNMENT 01

6) ip:

Description: Shows and manipulates network interfaces.

Example Usage: ip addr show



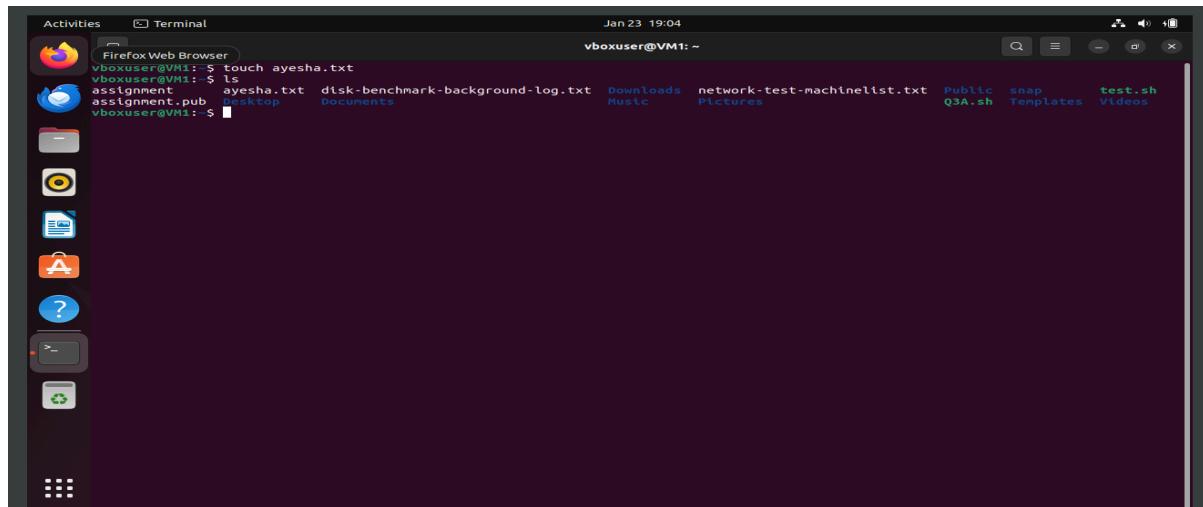
A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and other applications. The main window is a terminal window titled 'Terminal' with the command 'vboxuser@VM1: ~'. The output of the 'ip addr show' command is displayed, listing network interfaces like 'lo', 'eth0', and 'enp0s3' with their respective details. The terminal window has a dark background and light-colored text. The status bar at the top shows 'Activities' and the date 'Jan 23 19:02'.

```
vboxuser@VM1: $ ip addr show
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 08:00:27:78:63:cd brd ff:ff:ff:ff:ff:ff
        inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
            brd 10.0.2.15
            inet6 fe80::800:27ff:fe78:63cd/64 scope link noprefixroute
                valid_lft forever preferred_lft forever
vboxuser@VM1: $
```

7) touch:

Description: Creates an empty file or updates the access and modification timestamps of a file.

Example Usage: touch filename.txt



A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and other applications. The main window is a terminal window titled 'Terminal' with the command 'vboxuser@VM1: ~'. The output of the 'ls' command is shown, listing files like 'ayesha.txt', 'disk-benchmark-background-log.txt', and 'test.sh'. The terminal window has a dark background and light-colored text. The status bar at the top shows 'Activities' and the date 'Jan 23 19:04'.

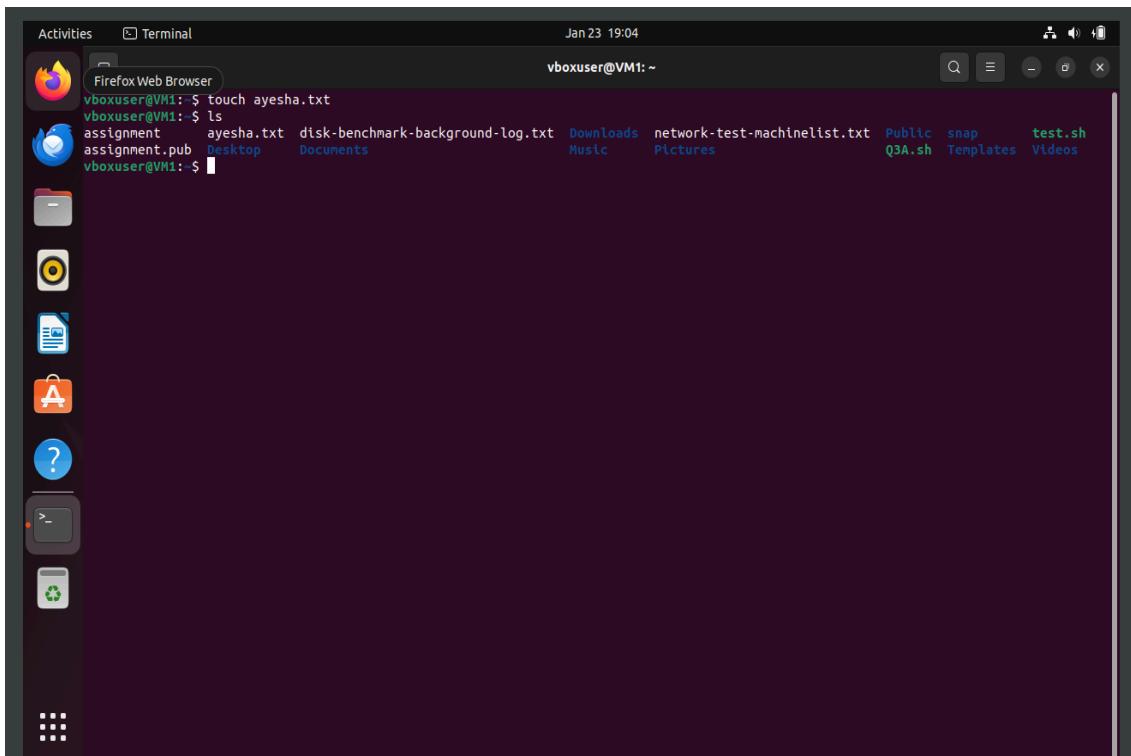
```
vboxuser@VM1: $ touch ayesha.txt
vboxuser@VM1: $ ls
ayesha.txt  disk-benchmark-background-log.txt  Downloads  network-test-machinelist.txt  Public  snap  test.sh
assignment1  assignment1.pub  desktop  Documents  Music  Pictures  QSA.sh  Templates  Videos
vboxuser@VM1: $
```

CLOUD COMPUTING ASSIGNMENT 01

8) ls (List):

Description: Lists files and directories in the current directory.

Example Usage: ls



A screenshot of a Linux desktop environment. On the left is a dock with various icons: Firefox, Dash, Home, Applications, Documents, Activities, Help, and a terminal icon. The main window is a terminal titled 'Terminal' with the command 'ls' being run. The output shows the contents of the current directory, which includes files like 'assignment', 'assignment.pub', 'ayesha.txt', 'disk-benchmark-background-log.txt', 'Downloads', 'network-test-machinelist.txt', 'Public', 'snap', 'test.sh', and 'Q3A.sh'. It also lists standard directory names: Desktop, Documents, Music, Pictures, Templates, and Videos. The terminal window has a dark background and light-colored text. The status bar at the top right shows the date and time as 'Jan 23 19:04' and the user as 'vboxuser@VM1: ~'.

9) mkdir (Make Directory):

Description: Creates a new directory.

Example Usage: mkdir new_directory

CLOUD COMPUTING ASSIGNMENT 01

A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications: a browser (Firefox), a file manager (Nautilus), a terminal (GNOME Terminal), a text editor (gedit), a file viewer (Eye of GNOME), a help center (Ubuntu Help), a system monitor (System Monitor), and a trash can. The main window is a terminal window titled 'Terminal' with the command 'vboxuser@VM1:~\$'. The terminal shows the user has run 'mkdir directory101' and then 'ls' to list the contents of their home directory. The output of 'ls' includes:

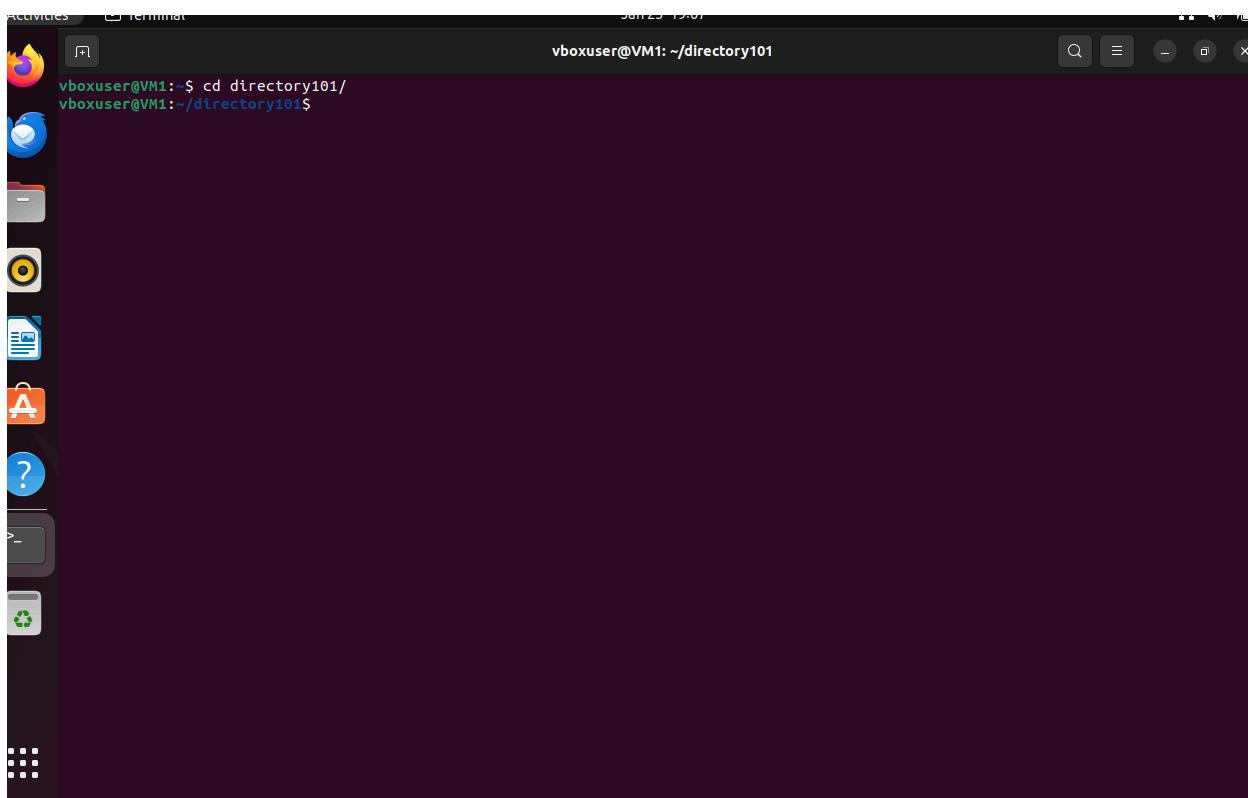
```
vboxuser@VM1:~$ ls
assignment          Desktop           Documents   network-test-machinelist.txt  Q3A.sh      test.sh
assignment.pub       directory101      Downloads    Pictures             snap        Videos
ayesha.txt          disk-benchmark-background-log.txt  Music       Public              Templates
```

10) cd (Change Directory):

Description: Changes the current working directory.

Example Usage: cd /path/to/directory

CLOUD COMPUTING ASSIGNMENT 01



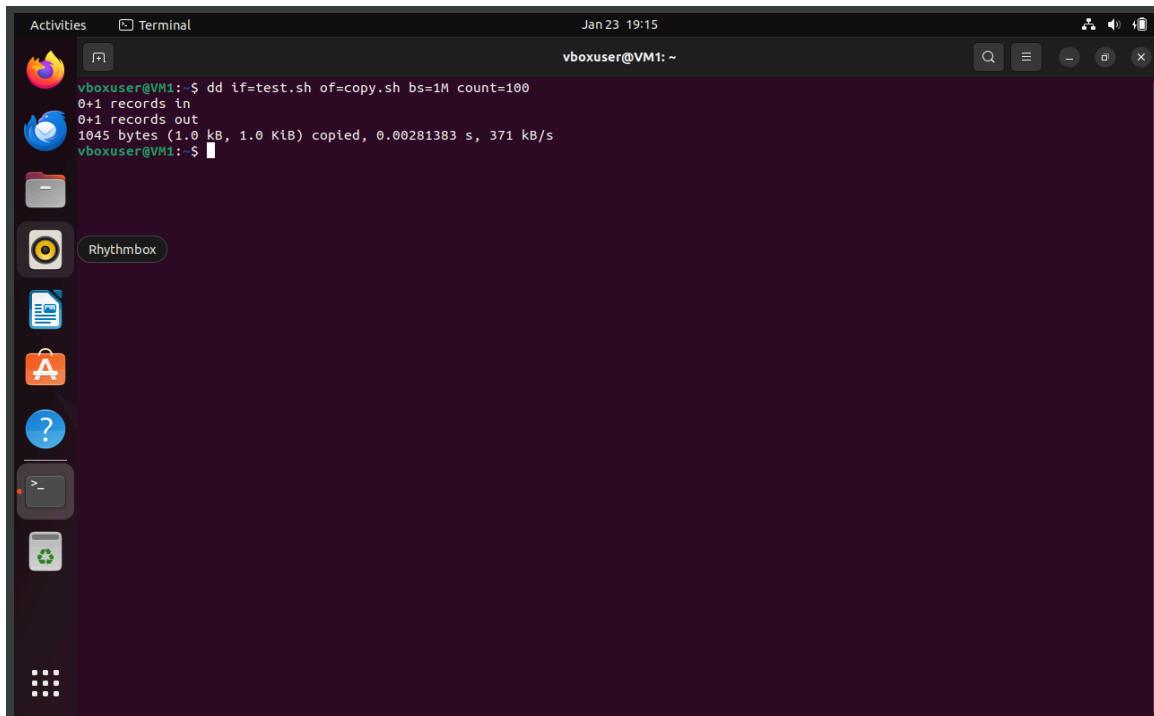
A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window title is "Terminal" and the command prompt is "vboxuser@VM1: ~/directory101". The terminal shows the command "cd directory101/" being run. The desktop interface includes a dock with various icons like Dash, Home, and Applications, and a vertical Activities overview panel on the left.

```
vboxuser@VM1: ~$ cd directory101/  
vboxuser@VM1:~/directory101$
```

11) dd:

Description: Copies and converts data according to specified options.

Example Usage: dd if=input_file of=output_file bs=4k



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window title is "Terminal" and the command prompt is "vboxuser@VM1: ~". The terminal shows the command "dd if=test.sh of=copy.sh bs=1M count=100" being run. The output indicates 0+1 records were copied at a rate of 1045 bytes (1.0 kB, 1.0 KiB) per second. The desktop interface includes a dock with various icons like Dash, Home, and Applications, and a vertical Activities overview panel on the left.

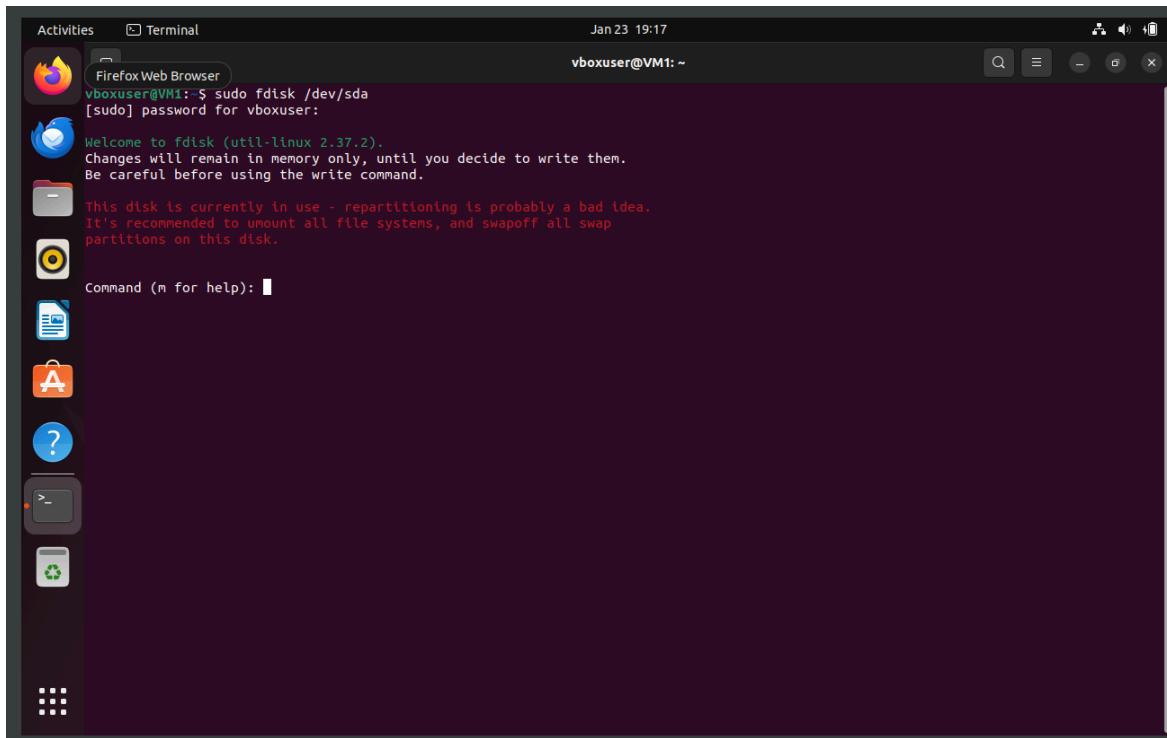
```
vboxuser@VM1: ~$ dd if=test.sh of=copy.sh bs=1M count=100  
0+1 records in  
0+1 records out  
1045 bytes (1.0 kB, 1.0 KiB) copied, 0.00281383 s, 371 kB/s  
vboxuser@VM1: ~$
```

CLOUD COMPUTING ASSIGNMENT 01

12) fdisk:

Description: Manipulates disk partition tables.

Example Usage: fdisk /dev/sdX



A screenshot of a Linux desktop environment. The terminal window shows the command `sudo fdisk /dev/sda` being run. The output includes a warning about the disk being in use and a prompt for a password. The desktop interface features a dock with icons for various applications like a web browser, file manager, and terminal.

```
vboxuser@VM1: ~$ sudo fdisk /dev/sda
[sudo] password for vboxuser:
Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

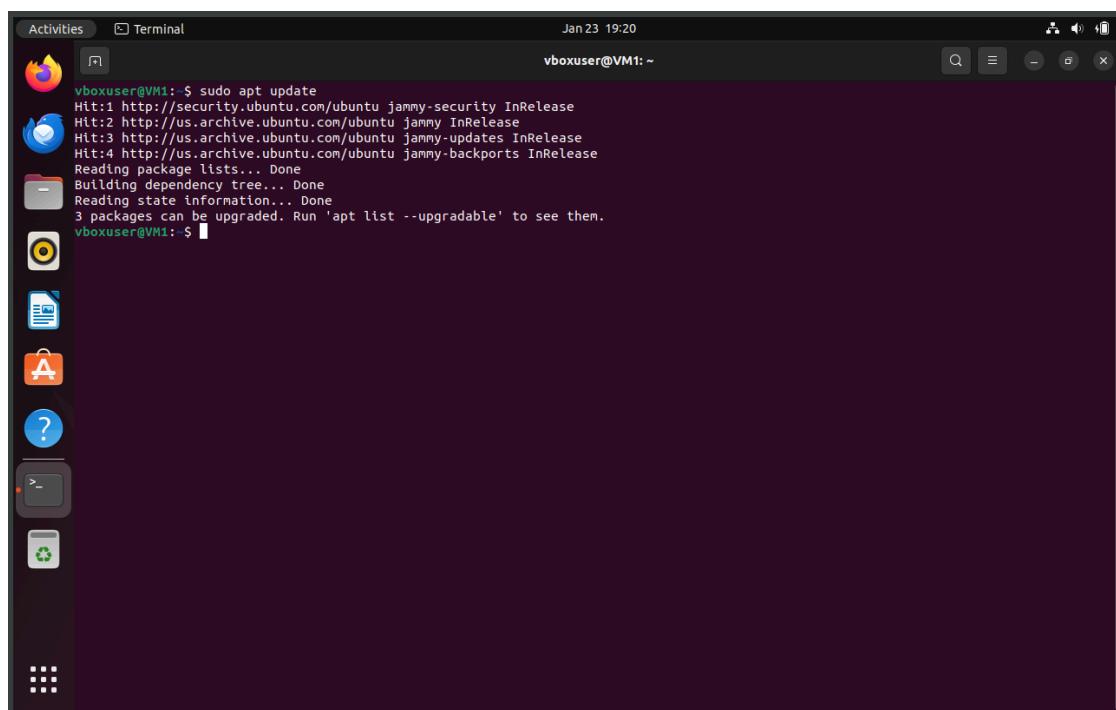
This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to unmount all file systems, and swapoff all swap
partitions on this disk.

Command (m for help):
```

13) apt (Advanced Package Tool):

Description: Package management tool for Debian-based systems.

Example Usage: sudo apt update && sudo apt install package_name



A screenshot of a Linux desktop environment. The terminal window shows the command `sudo apt update` being run. The output displays the progress of fetching package lists from various repositories. The desktop interface features a dock with icons for various applications like a web browser, file manager, and terminal.

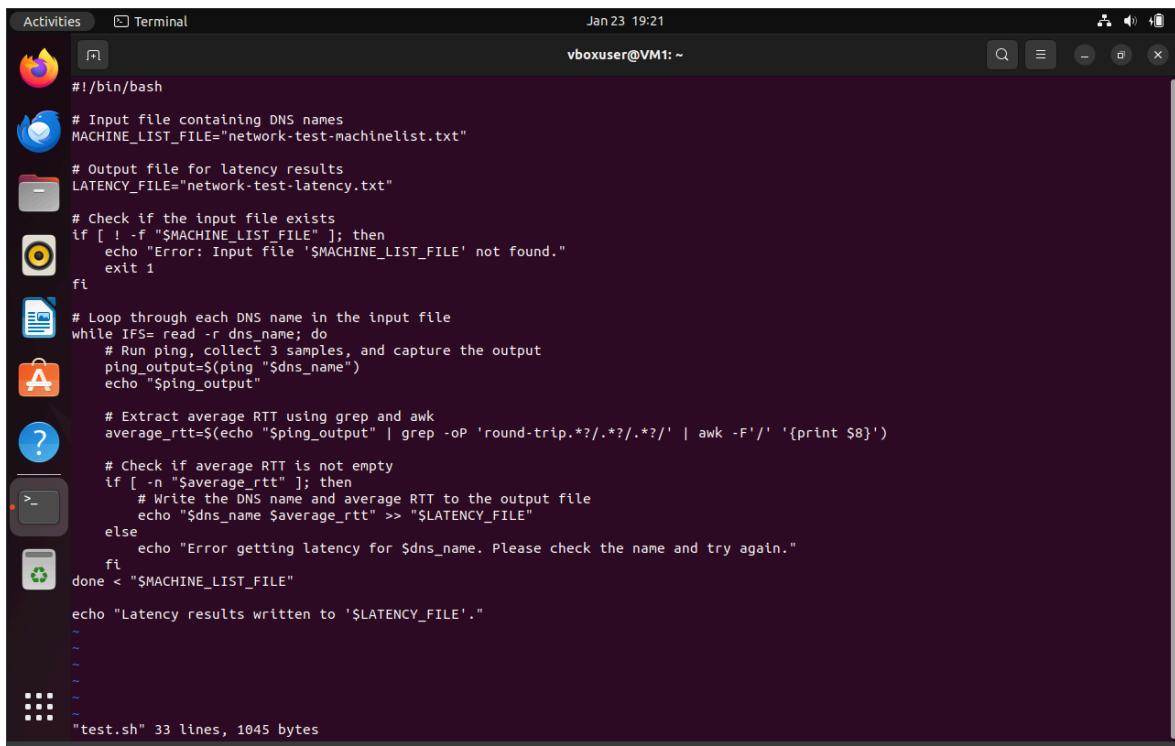
```
vboxuser@VM1: ~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
vboxuser@VM1: ~$
```

CLOUD COMPUTING ASSIGNMENT 01

14) vi:

Description: Text editor with modes for inserting, navigating, and editing text.

Example Usage: vi filename.txt



```
#!/bin/bash

# Input file containing DNS names
MACHINE_LIST_FILE="network-test-machinelist.txt"

# Output file for latency results
LATENCY_FILE="network-test-latency.txt"

# Check if the input file exists
if [ ! -f "$MACHINE_LIST_FILE" ]; then
    echo "Error: Input file '$MACHINE_LIST_FILE' not found."
    exit 1
fi

# Loop through each DNS name in the input file
while IFS= read -r dns_name; do
    # Run ping, collect 3 samples, and capture the output
    ping_output=$(ping "$dns_name")
    echo "$ping_output"

    # Extract average RTT using grep and awk
    average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/.*?/' | awk -F'/' '{print $8}')

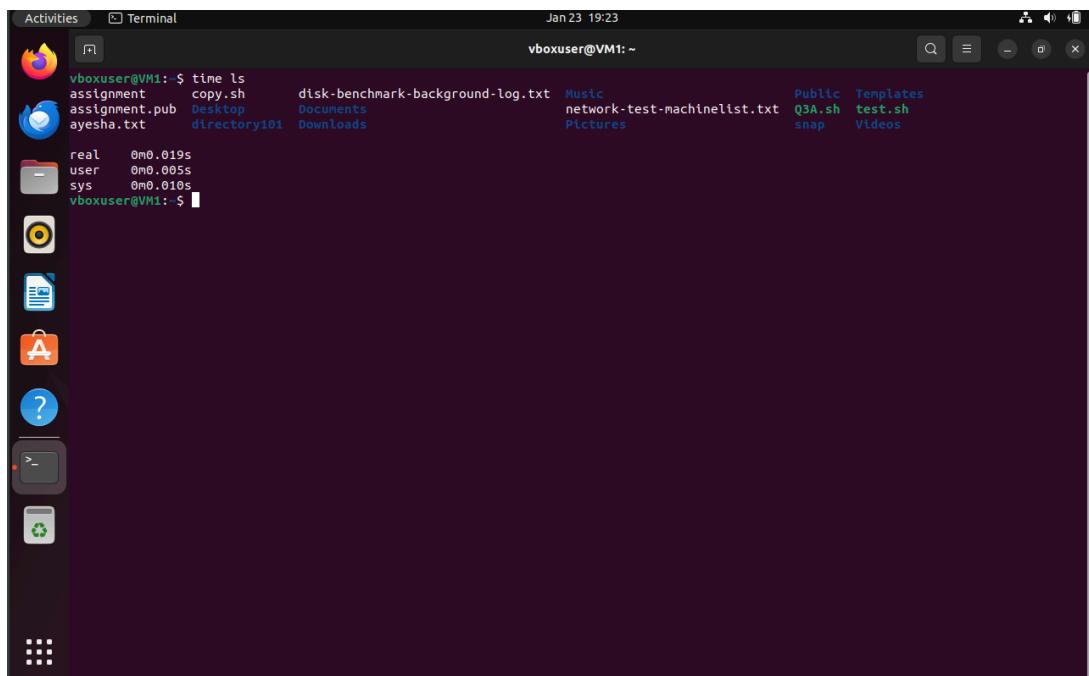
    # Check if average RTT is not empty
    if [ -n "$average_rtt" ]; then
        # Write the DNS name and average RTT to the output file
        echo "$dns_name $average_rtt" >> "$LATENCY_FILE"
    else
        echo "Error getting latency for $dns_name. Please check the name and try again."
    fi
done < "$MACHINE_LIST_FILE"

echo "Latency results written to '$LATENCY_FILE'."
```

15) time:

Description: Measures the execution time of a command.

Example Usage: time command_to_measure



```
vboxuser@VM1:~$ time ls
assignment      copy.sh      disk-benchmark-background-log.txt  Music          Public   Templates
assignment.pub  Desktop     Documents                  network-test-machinelist.txt  Q3A.sh  test.sh
ayesha.txt      directory101 Downloads                Pictures          snap    Videos
real    0m0.019s
user    0m0.005s
sys     0m0.010s
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

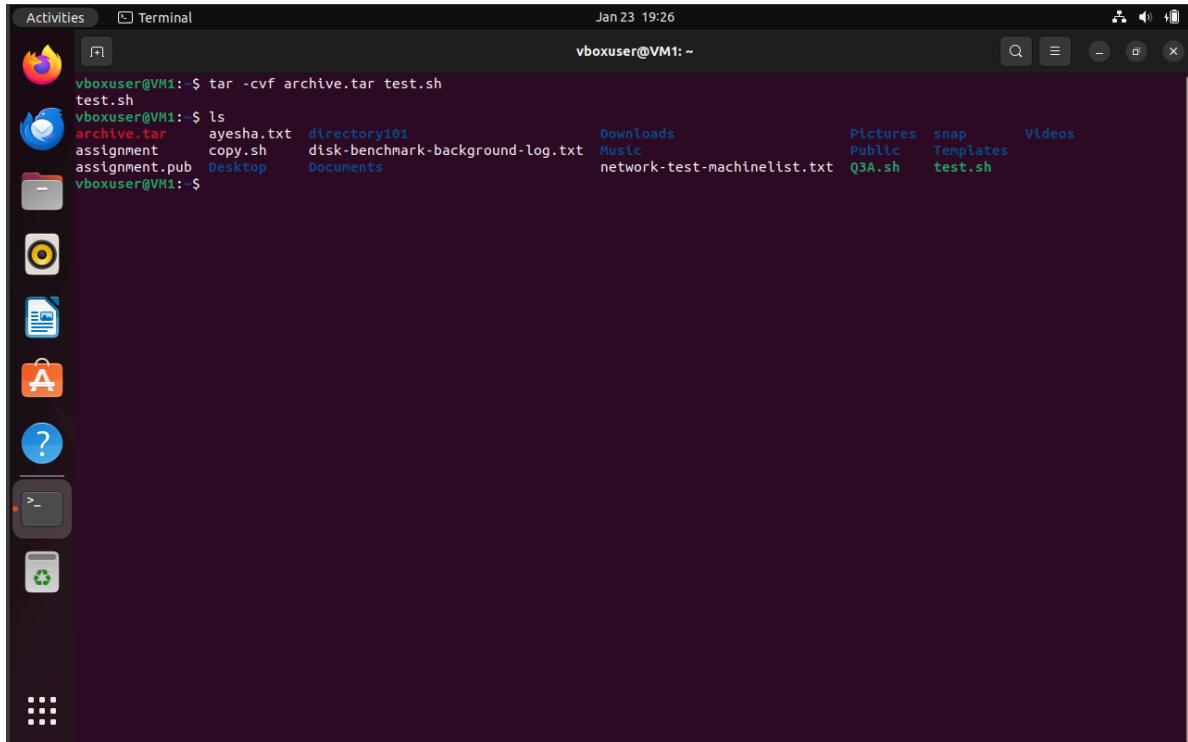
16) tar:

Description: Creates or extracts files from an archive file.

Example Usage:

To create a tar archive: tar -cvf archive.tar file1 file2.

To extract from a tar archive: tar -xvf archive.tar



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications like a browser, file manager, and system tools. The main area shows a terminal window titled 'Terminal' with the command 'vboxuser@VM1: ~'. The terminal output shows the creation of a tar archive and its contents:

```
vboxuser@VM1: $ tar -cvf archive.tar test.sh
test.sh
vboxuser@VM1: $ ls
archive.tar    ayesha.txt  directory101      Downloads          Pictures   snap      Videos
assignment     copy.sh     disk-benchmark-background-log.txt  Music       Public    Templates
assignment.pub  Desktop    Documents        network-test-machinelist.txt  Q3A.sh    test.sh
```

17) rm (Remove):

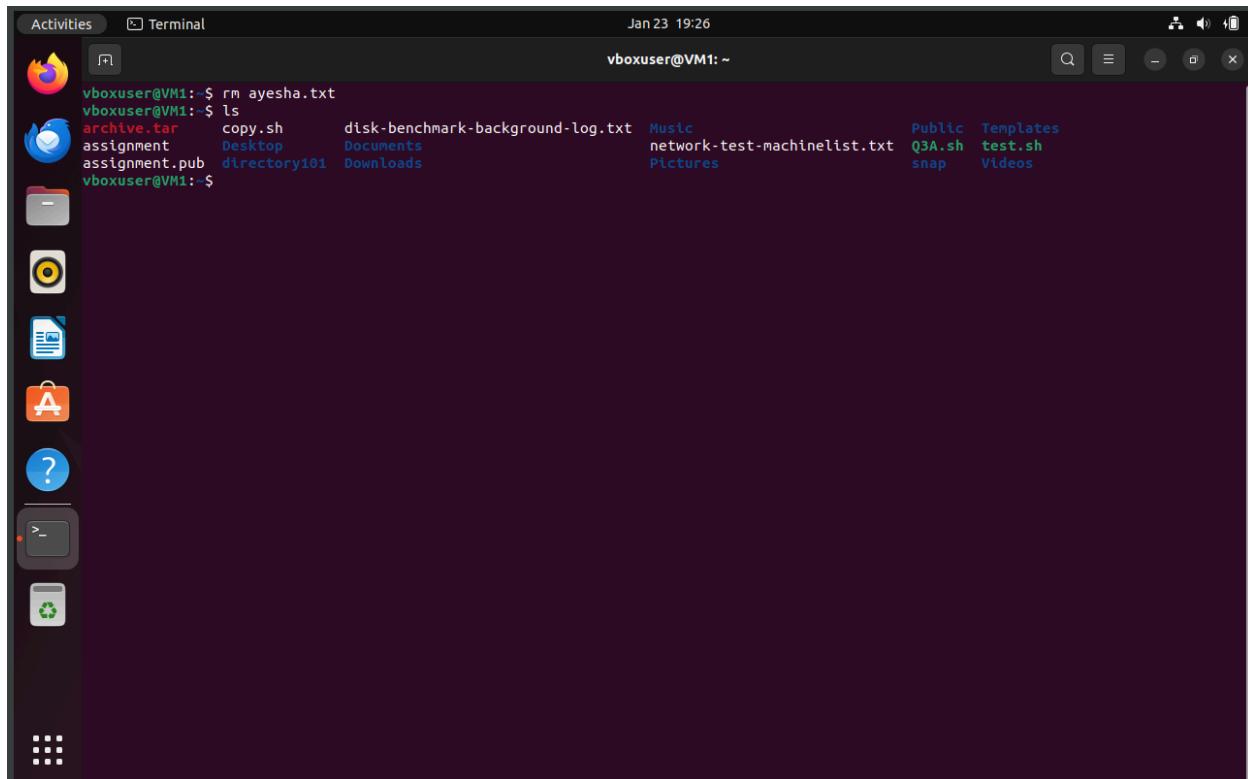
Description: Removes (deletes) files or directories.

Example Usage:

To remove a file: rm filename.txt

To remove a directory and its contents: rm -r directory_name

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal window has a dark purple background and displays the following command and its output:

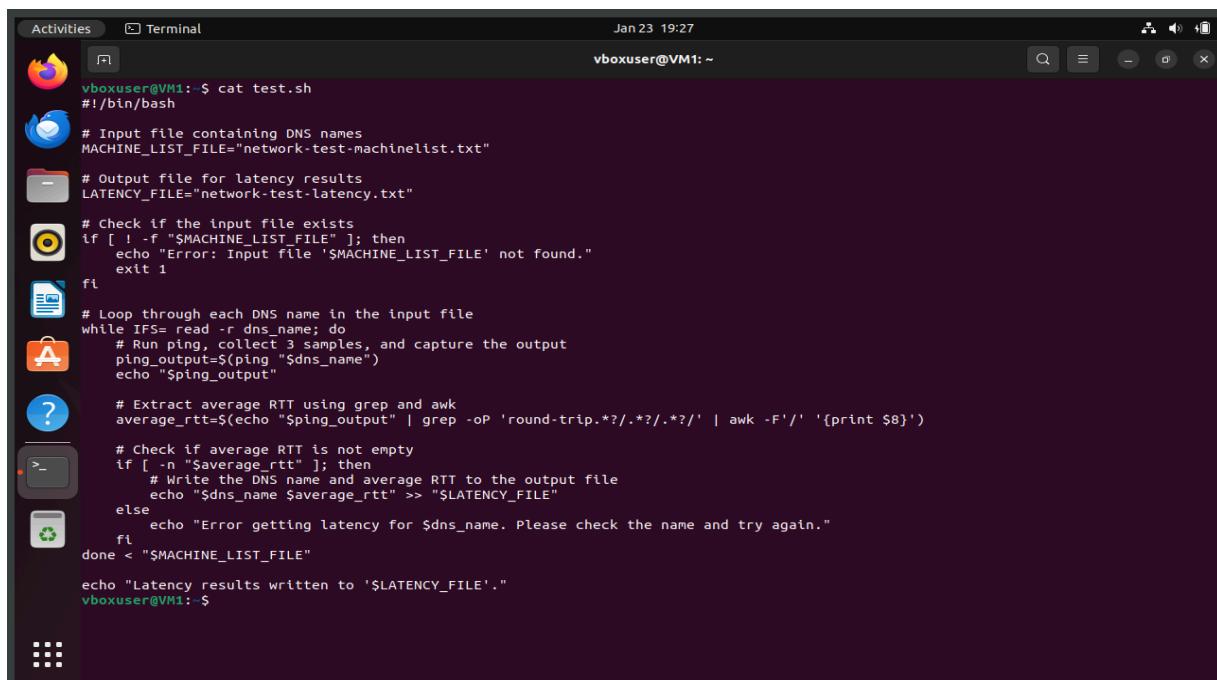
```
vboxuser@VM1:~$ rm ayesha.txt
vboxuser@VM1:~$ ls
archive.tar      copy.sh      disk-benchmark-background-log.txt  Music          Public   Templates
assignment       Desktop     Documents           network-test-machinelist.txt  Q3A.sh  test.sh
assignment.pub   directory101 Downloads    Pictures          snap     Videos
vboxuser@VM1:~$
```

The terminal window is part of a desktop interface with a dock on the left containing icons for various applications like Dash, Home, Dash to Dock, and others.

18) cat (Concatenate):

Description: Displays the content of a file or concatenates files.

Example Usage: cat filename.txt



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window titled "Terminal". The terminal window has a dark purple background and displays the content of a script named "test.sh". The script uses "#!" syntax to indicate it's a bash script. It defines variables for input and output files, checks if the input file exists, loops through DNS names, runs ping commands, and calculates average RTT using awk. Finally, it writes the results to an output file.

```
vboxuser@VM1:~$ cat test.sh
#!/bin/bash

# Input file containing DNS names
MACHINE_LIST_FILE="network-test-machinelist.txt"

# Output file for latency results
LATENCY_FILE="network-test-latency.txt"

# Check if the input file exists
if [ ! -f "$MACHINE_LIST_FILE" ]; then
    echo "Error: Input file '$MACHINE_LIST_FILE' not found."
    exit 1
fi

# Loop through each DNS name in the input file
while IFS= read -r dns_name; do
    # Run ping, collect 3 samples, and capture the output
    ping_output=$(ping "$dns_name")
    echo "$ping_output"

    # Extract average RTT using grep and awk
    average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/.*?/' | awk -F'/' '{print $8}')

    # Check if average RTT is not empty
    if [ -n "$average_rtt" ]; then
        # Write the DNS name and average RTT to the output file
        echo "$dns_name $average_rtt" >> "$LATENCY_FILE"
    else
        echo "Error getting latency for $dns_name. Please check the name and try again."
    fi
done < "$MACHINE_LIST_FILE"

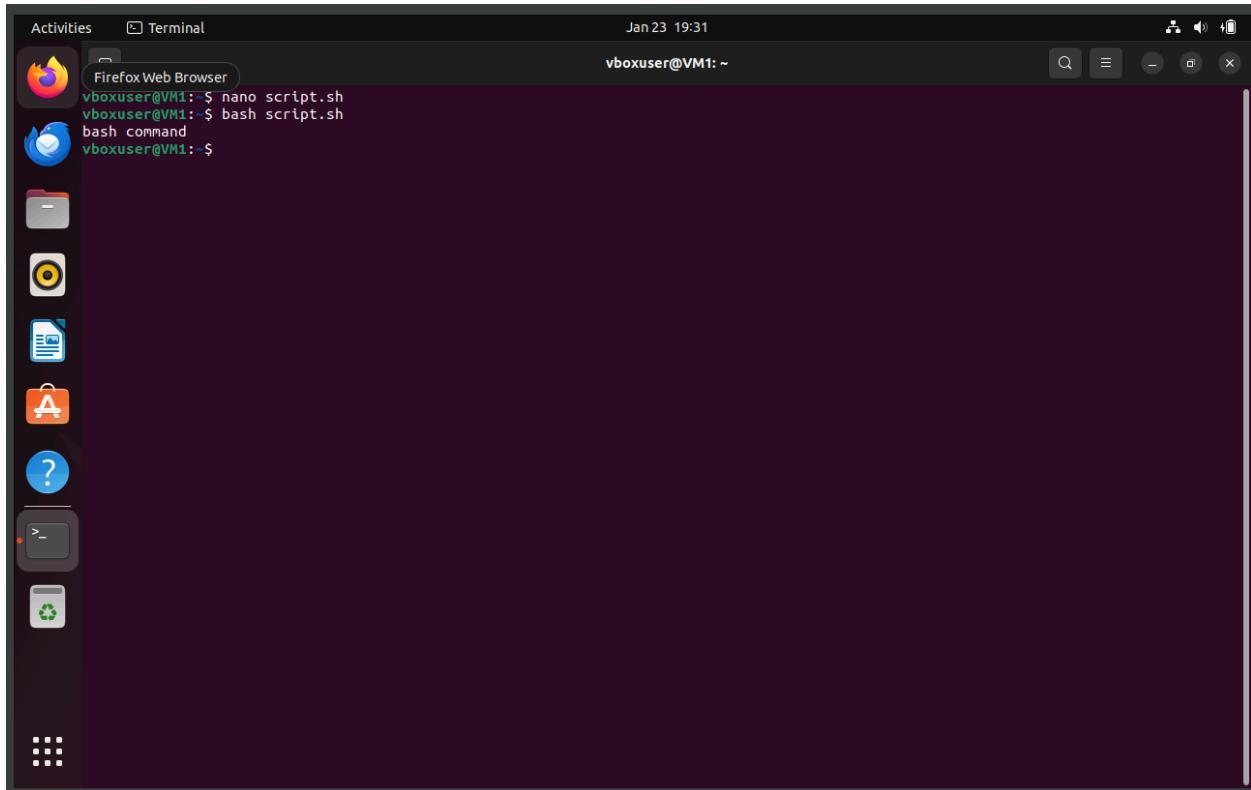
echo "Latency results written to '$LATENCY_FILE'."
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

19) bash:

Description: The GNU Bourne-Again SHell, a command processor that typically runs in a text window.

Example Usage: Running a script: bash script.sh

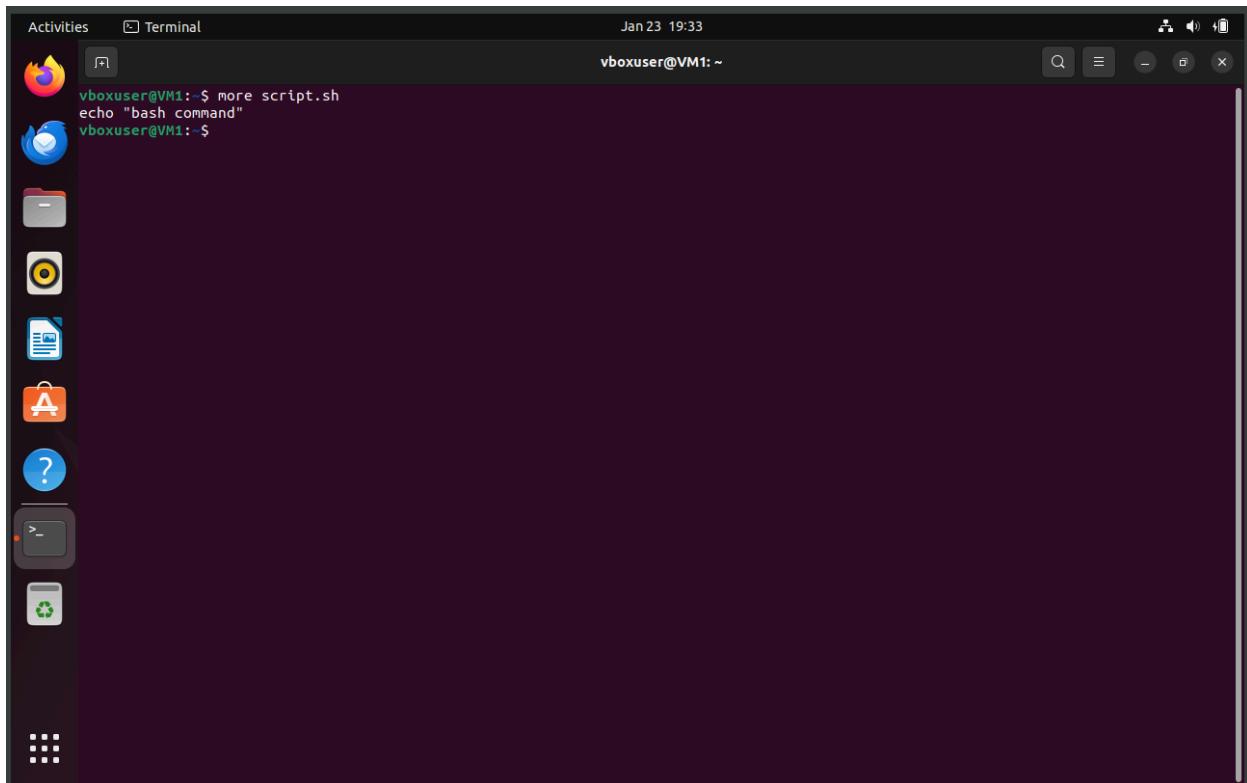


20) more:

Description: Displays the contents of a file one screen at a time.

Example Usage: more filename.txt

CLOUD COMPUTING ASSIGNMENT 01



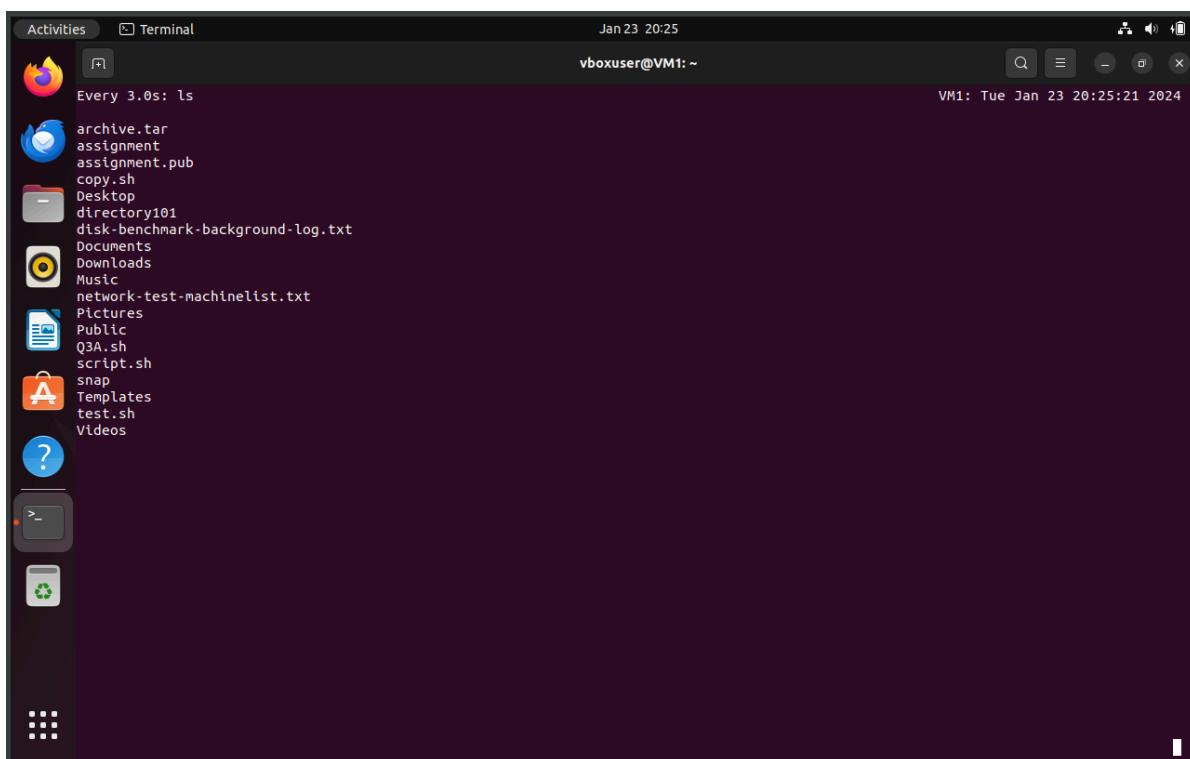
A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. A terminal window is open at the top right, showing the command:

```
vboxuser@VM1:~$ more script.sh  
echo "bash command"  
vboxuser@VM1:~$
```

21) watch:

Description: Periodically executes a command and displays the output.

Example Usage: `watch -n 5 command_to_watch`



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. A terminal window is open at the top right, showing the command:

```
Every 3.0s: ls  
archive.tar  
assignment  
assignment.pub  
copy.sh  
Desktop  
directory101  
disk-benchmark-background-log.txt  
Documents  
Downloads  
Music  
network-test-machinelist.txt  
Pictures  
Public  
Q3A.sh  
script.sh  
snap  
Templates  
test.sh  
Videos
```

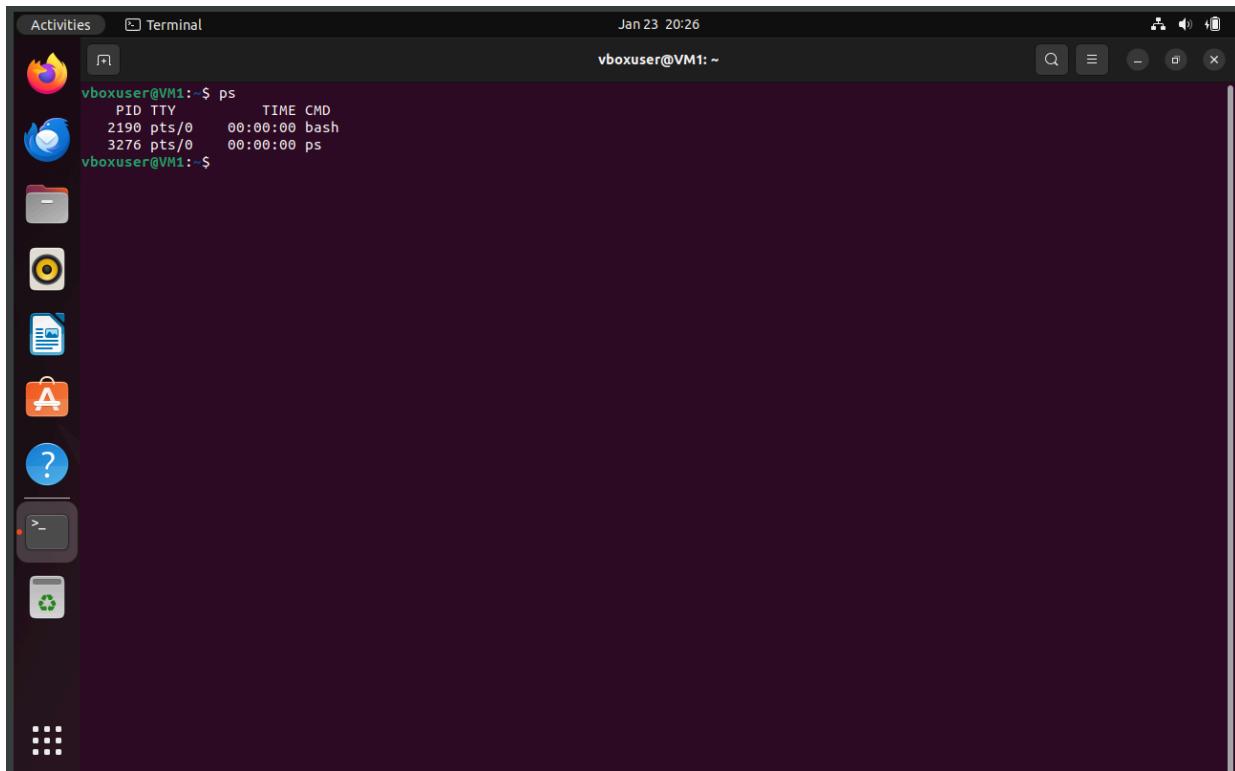
The terminal window also shows the date and time: `VM1: Tue Jan 23 20:25:21 2024`.

CLOUD COMPUTING ASSIGNMENT 01

22) ps (Process Status):

Description: Displays information about processes running on the system.

Example Usage: ps aux



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for a file manager, terminal, browser, and other applications. The main window is a terminal window titled 'Terminal' with the command 'ps' run, showing two processes: bash and ps. The terminal window has a dark background and light text. The desktop background is also dark.

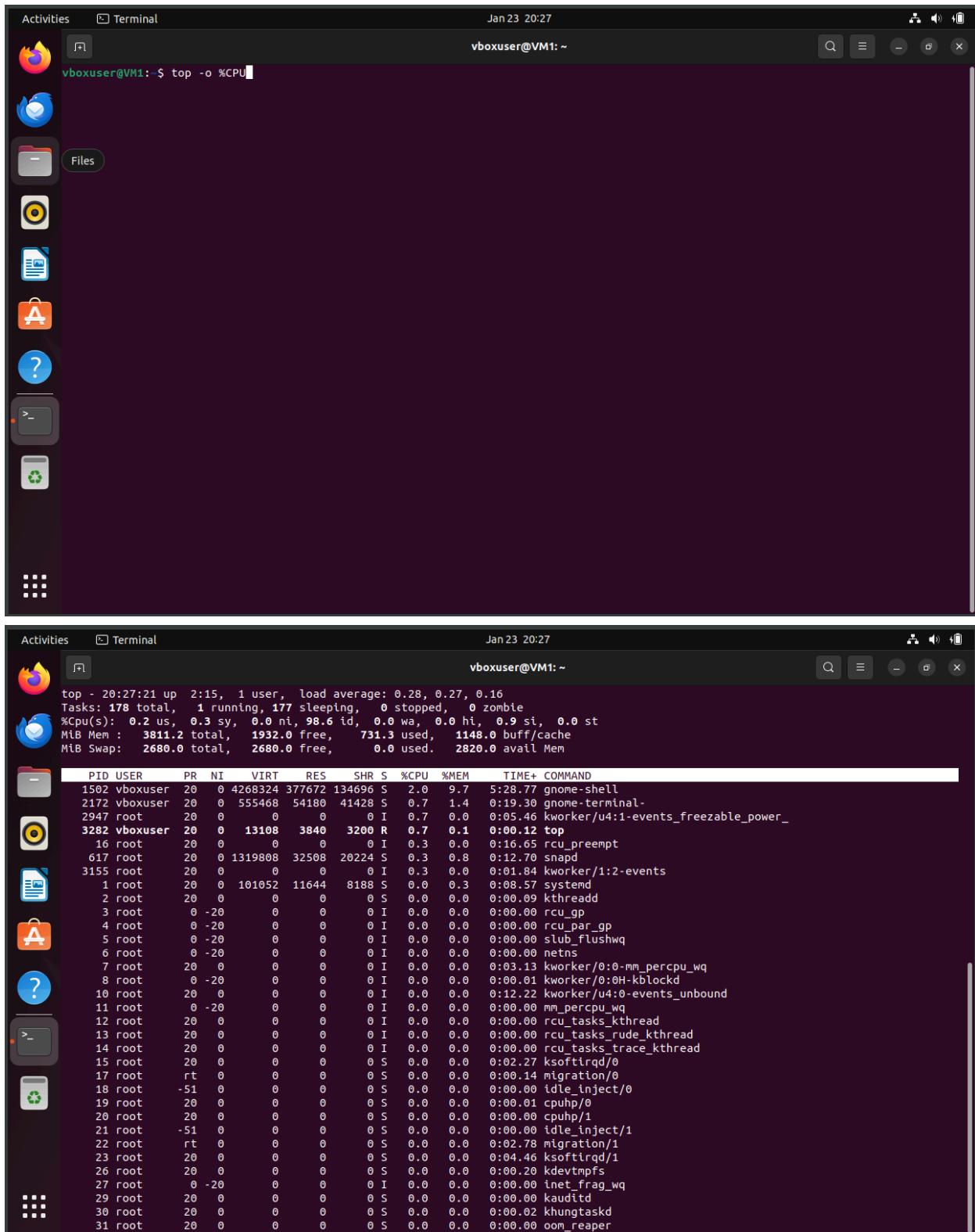
```
vboxuser@VM1:~$ ps
  PID TTY      TIME CMD
 2190 pts/0    00:00:00 bash
 3276 pts/0    00:00:00 ps
vboxuser@VM1:~$
```

23) top:

Description: Displays real-time system information, including a list of processes.

Example Usage: top

CLOUD COMPUTING ASSIGNMENT 01



```
vboxuser@VM1:~$ top -o %CPU
```

```
top - 20:27:21 up 2:15, 1 user, load average: 0.28, 0.27, 0.16
Tasks: 178 total, 1 running, 177 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.3 sy, 0.0 ni, 98.6 id, 0.0 wa, 0.0 hi, 0.9 si, 0.0 st
MiB Mem : 3811.2 total, 1932.0 free, 731.3 used, 1149.0 buff/cache
MiB Swap: 2680.0 total, 2680.0 free, 0.0 used. 2820.0 avail Mem
```

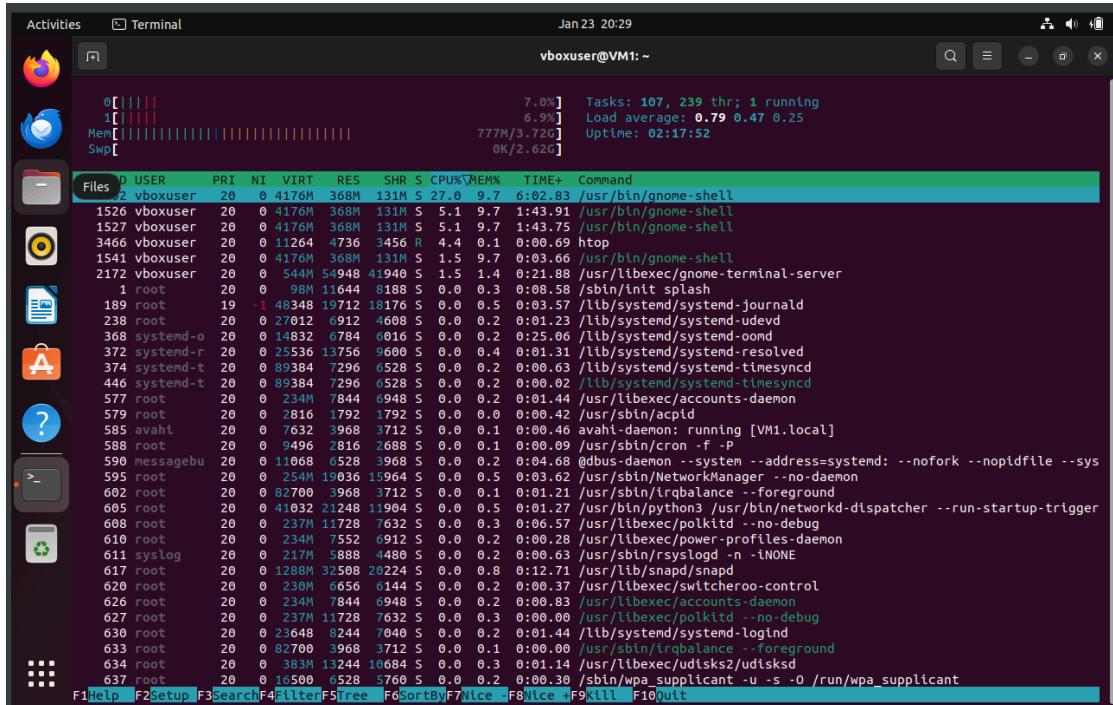
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1502	vboxuser	20	0	4268324	377672	134696	S	2.0	9.7	5:28.77	gnome-shell
2172	vboxuser	20	0	555468	54180	41428	S	0.7	1.4	0:19.30	gnome-terminal-
2947	root	20	0	0	0	0	I	0.7	0.0	0:05.46	worker/u4:1-events_freezable_power_-
3282	vboxuser	20	0	13108	3840	3200	R	0.7	0.1	0:00.12	top
16	root	20	0	0	0	0	I	0.3	0.0	0:16.65	rcu_prempt
617	root	20	0	1319808	32508	20224	S	0.3	0.8	0:12.70	snapd
3155	root	20	0	0	0	0	I	0.3	0.0	0:01.84	worker/1:2-events
1	root	20	0	101052	11644	8188	S	0.0	0.3	0:08.57	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.09	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_flushwq
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	20	0	0	0	0	I	0.0	0.0	0:03.13	worker/0:0-mm_percpu_wq
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.01	worker/0:0H-kblockd
10	root	20	0	0	0	0	I	0.0	0.0	0:12.22	worker/u4:0-events_unbound
11	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
13	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
15	root	20	0	0	0	0	S	0.0	0.0	0:02.27	ksoftirqd/0
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.14	migration/0
18	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
19	root	20	0	0	0	0	S	0.0	0.0	0:00.01	cpuhp/0
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
21	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
22	root	rt	0	0	0	0	S	0.0	0.0	0:02.78	migration/1
23	root	20	0	0	0	0	S	0.0	0.0	0:04.46	ksoftirqd/1
26	root	20	0	0	0	0	S	0.0	0.0	0:00.20	kdevtmpfs
27	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	inet_frag_wq
29	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kauditfd
30	root	20	0	0	0	0	S	0.0	0.0	0:00.02	khungtaskd
31	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reaper

CLOUD COMPUTING ASSIGNMENT 01

24) htop:

Description: An interactive process viewer and system monitor.

Example Usage: htop



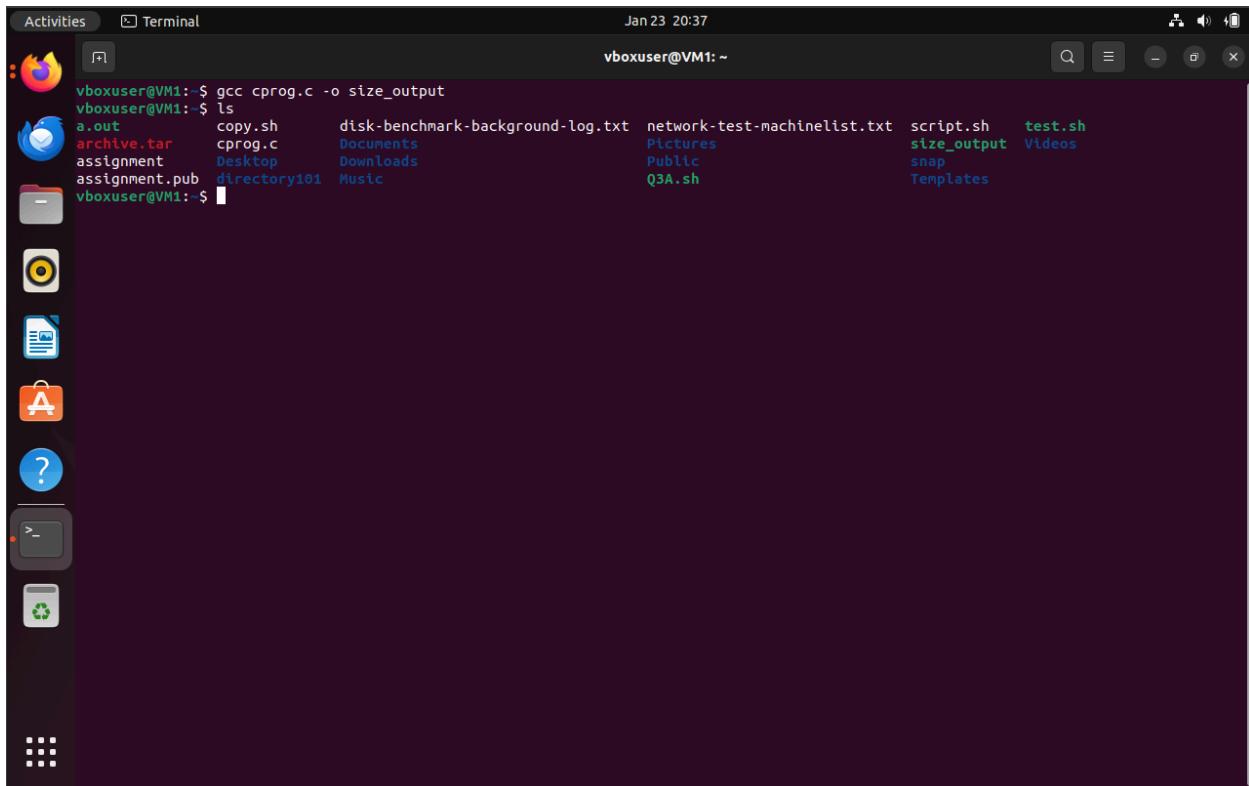
```
Activities Terminal Jan 23 20:30
vboxuser@VM1: ~
vboxuser@VM1: $ htop
Command 'htop' not found, but can be installed with:
  sudo snap install htop # version 3.3.0, or
  sudo apt install htop # version 3.0.5-7build2
See 'snap info htop' for additional versions.
vboxuser@VM1: $ sudo apt install htop
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  lm-sensors
The following NEW packages will be installed:
  htop
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 128 kB of archives.
After this operation, 342 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy amd64 htop amd64 3.0.5-7build2 [128 kB]
Fetched 128 kB in 0s (323 kB/s)
Selecting previously unselected package htop.
(Reading database ... 208380 files and directories currently installed.)
Preparing to unpack .../htop_3.0.5-7build2_amd64.deb ...
Unpacking htop (3.0.5-7build2) ...
Setting up htop (3.0.5-7build2) ...
Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
Processing triggers for man-db (2.10.2-1) ...
vboxuser@VM1: $ htop
vboxuser@VM1: $
```

CLOUD COMPUTING ASSIGNMENT 01

25) gcc (GNU Compiler Collection):

Description: Compiler for various programming languages, primarily C, C++, and Fortran.

Example Usage: gcc -o output_program source_code.c



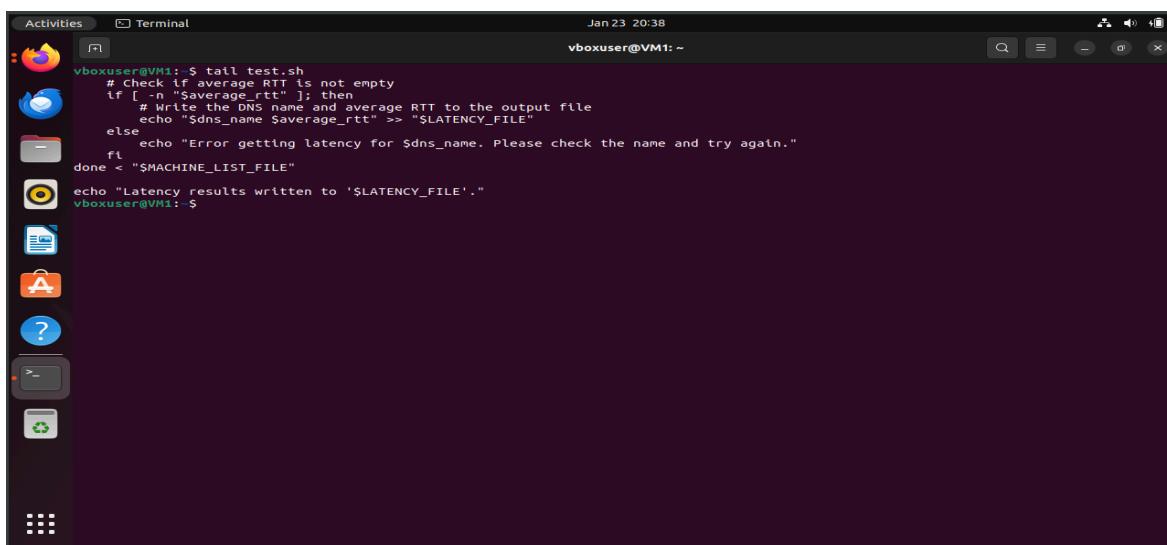
A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. The main window is a terminal titled "Terminal" with the command "vboxuser@VM1: ~". The terminal shows the following output:

```
vboxuser@VM1: $ gcc cprog.c -o size_output
vboxuser@VM1: $ ls
a.out      copy.sh      disk-benchmark-background-log.txt  network-test-machinelist.txt  script.sh    test.sh
archive.tar  cprog.c      Documents      Pictures        size_output  Videos
assignment   Desktop     Downloads      Public         snap
assignment.pub directory101 Music          Q3A.sh       Templates
```

26) tail:

Description: Displays the last part of a file (by default, the last 10 lines).

Example Usage: tail filename.txt



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. The main window is a terminal titled "Terminal" with the command "vboxuser@VM1: ~". The terminal shows the following output:

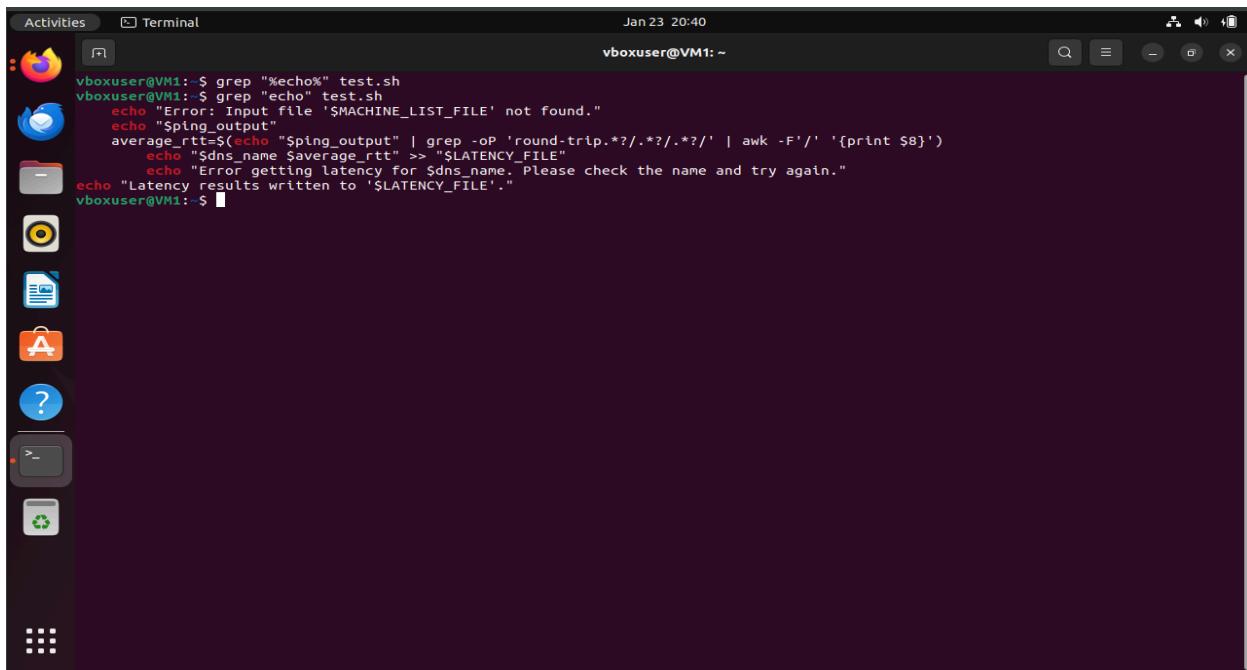
```
vboxuser@VM1: $ tail test.sh
# Check if average RTT is not empty
if [ -n "$average_rtt" ]; then
    # Write the DNS name and average RTT to the output file
    echo "$dns_name $average_rtt" >> "$LATENCY_FILE"
else
    echo "Error getting latency for $dns_name. Please check the name and try again."
fi
done < "$MACHINE_LIST_FILE"
echo "Latency results written to '$LATENCY_FILE'."
vboxuser@VM1: $
```

CLOUD COMPUTING ASSIGNMENT 01

27) grep:

Description: Searches for a specific pattern in files.

Example Usage: grep "pattern" filename.txt



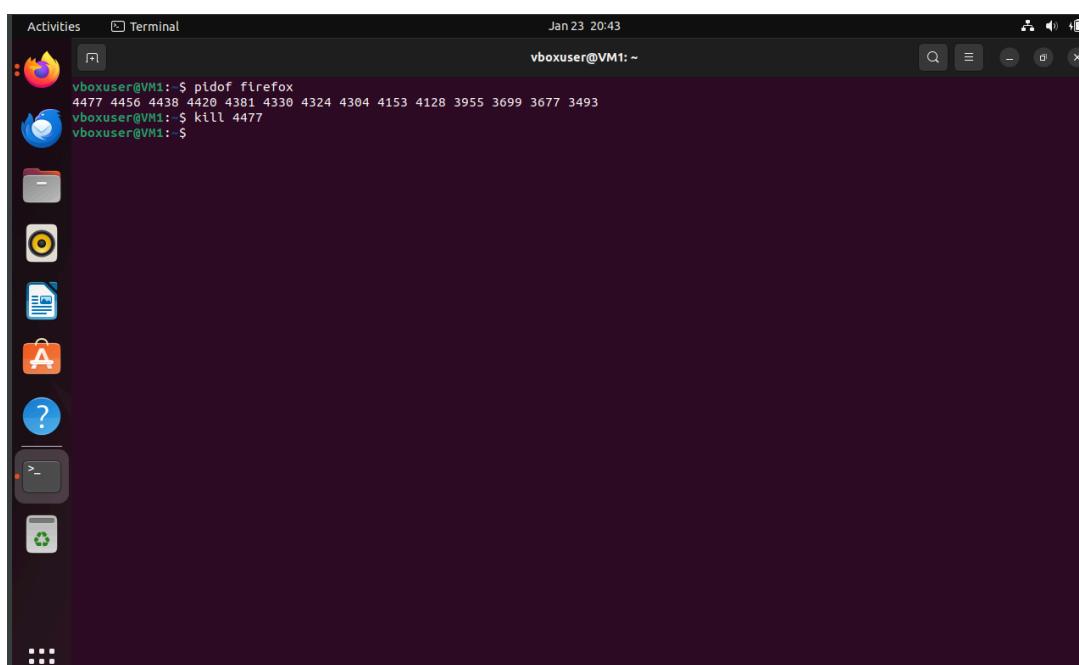
A screenshot of a Linux desktop environment. On the left is a dock with icons for a terminal, file manager, browser, and other applications. The main area shows a terminal window titled 'Terminal' with the command 'grep "%echo%" test.sh' entered. The output of the command is displayed, showing an error message about a missing input file and a warning about a syntax error in the script. The terminal window has a dark theme with white text and a light gray background. The desktop background is also dark.

```
vboxuser@VM1: $ grep "%echo%" test.sh
vboxuser@VM1: $ grep "echo" test.sh
    echo "Error: Input file '$MACHINE_LIST_FILE' not found."
    echo "$ping_output"
    average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/.*?' | awk -F'/' '{print $8}')
    echo "Sdns_name $average_rtt" >> "$LATENCY_FILE"
    echo "Error getting latency for $dns_name. Please check the name and try again."
echo "Latency results written to '$LATENCY_FILE'."
vboxuser@VM1: $
```

28) kill:

Description: Sends a signal to terminate a process.

Example Usage: kill process_id



A screenshot of a Linux desktop environment, similar to the one above. It shows a terminal window with the command 'pidof firefox' run, which outputs the process ID 4477. Then, the command 'kill 4477' is run to terminate the process. The terminal window has a dark theme with white text and a light gray background. The desktop background is dark.

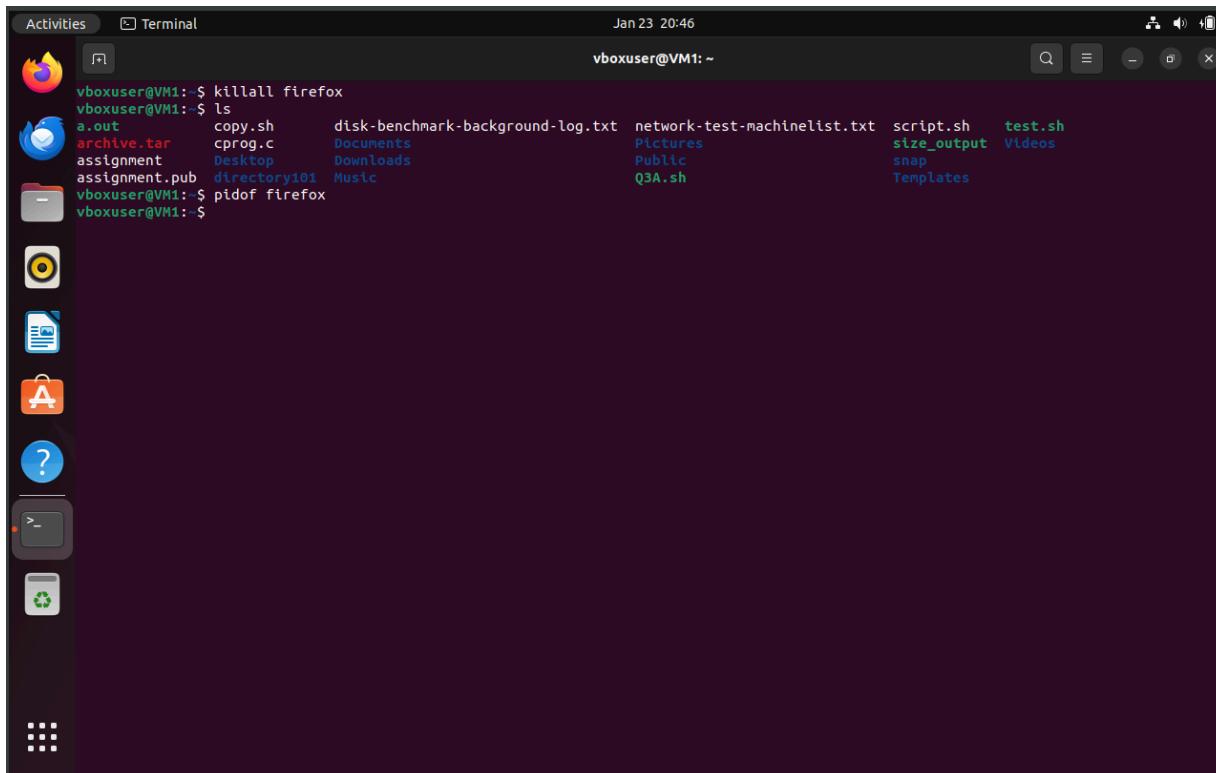
```
vboxuser@VM1: $ pidof firefox
4477 4456 4438 4420 4381 4330 4324 4304 4153 4128 3955 3699 3677 3493
vboxuser@VM1: $ kill 4477
vboxuser@VM1: $
```

CLOUD COMPUTING ASSIGNMENT 01

29) killall:

Description: Kills processes by name.

Example Usage: killall process_name



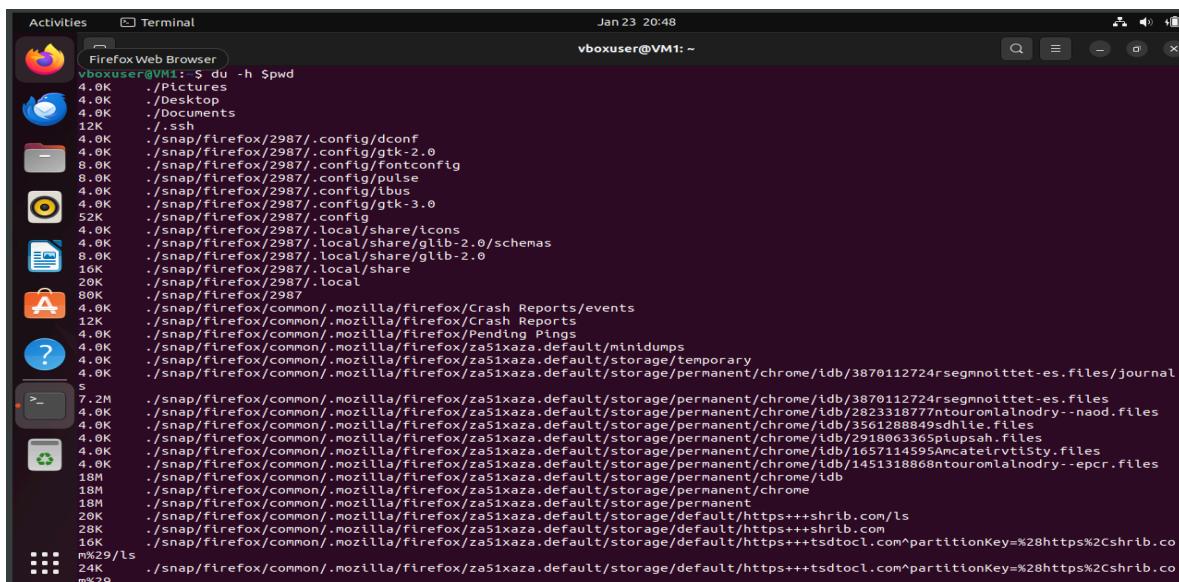
A screenshot of a Linux desktop environment. On the left is a dock with icons for a file manager, terminal, browser, and other applications. The main window is a terminal titled "Terminal" with the command "vboxuser@VM1: ~". The terminal shows the following session:

```
vboxuser@VM1: $ killall firefox
vboxuser@VM1: $ ls
a.out      copy.sh    disk-benchmark-background-log.txt  network-test-machinelist.txt  script.sh  test.sh
archive.tar  cprog.c   Documents          Pictures           size_output  Videos
assignment   Desktop   Downloads         Public            snap
assignment.pub directory01 Music          Q3A.sh           Templates
vboxuser@VM1: $ pidof firefox
vboxuser@VM1: $
```

30) du: (Disk Usage)

Description: Shows the disk space used by files and directories.

Example Usage: du -h directory_name



A screenshot of a Linux desktop environment. On the left is a dock with icons for a file manager, terminal, browser, and other applications. The main window is a terminal titled "Terminal" with the command "vboxuser@VM1: ~". The terminal shows the following session:

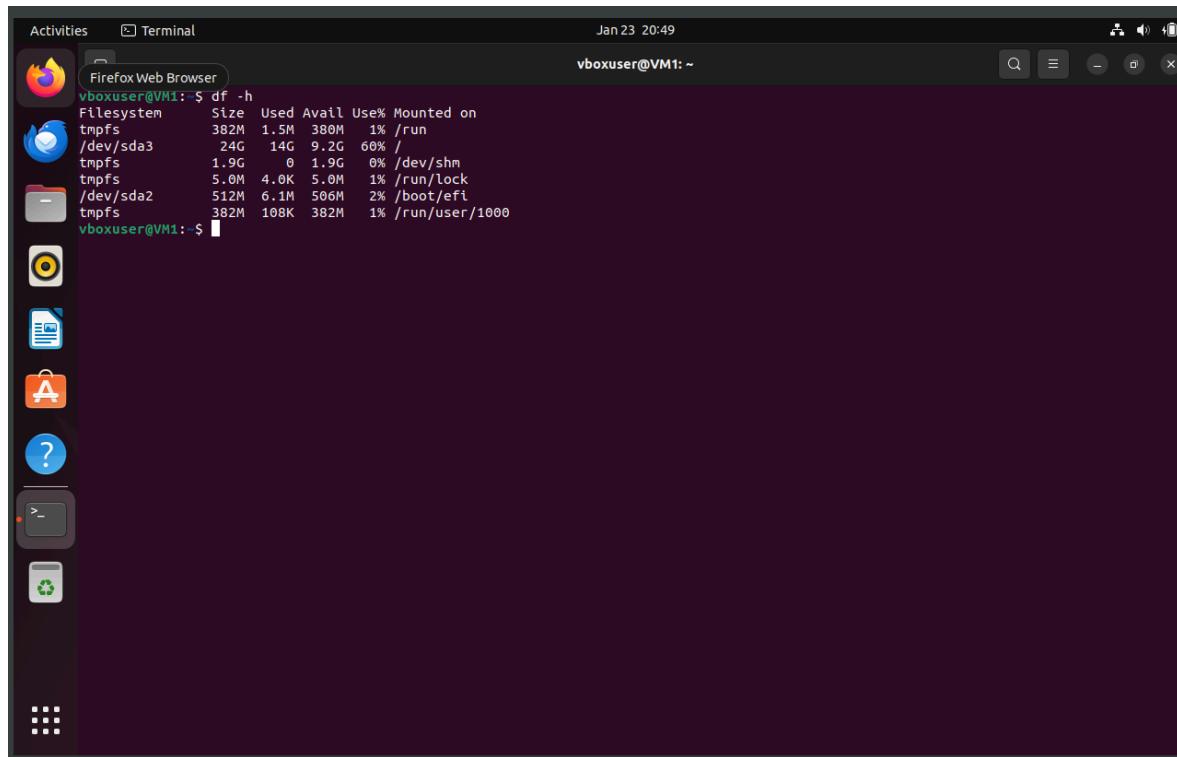
```
vboxuser@VM1: $ du -h Spwd
4.0K  ./Pictures
4.0K  ./Desktop
4.0K  ./Documents
12K   ./ssh
4.0K  ./snap/firefox/2987/.config/dconf
4.0K  ./snap/firefox/2987/.config/gtk-2.0
8.0K   ./snap/firefox/2987/.config/fontconfig
8.0K   ./snap/firefox/2987/.config/pulse
4.0K  ./snap/firefox/2987/.config/dbus
4.0K  ./snap/firefox/2987/.config/gtk-3.0
52K   ./snap/firefox/2987/.config
4.0K  ./snap/firefox/2987/.local/share/icons
4.0K  ./snap/firefox/2987/.local/share/glib-2.0/schemas
8.0K   ./snap/firefox/2987/.local/share/glib-2.0
16K   ./snap/firefox/2987/.local/share
20K   ./snap/firefox/2987/.local
80K   ./snap/firefox/2987
4.0K  ./snap/firefox/common/.mozilla/firefox/Crash Reports/events
12K   ./snap/firefox/common/.mozilla/firefox/Crash Reports
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/minidumps
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/temporary
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/3870112724rsegmnottet-es.files/journal
S     ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/3870112724rsegmnottet-es.files
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/2823318777ntouromlalnodry--naod.files
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/3561288849sdhlie.files
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/2918063365piupsah.files
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/1657114595AmcateIrvtisty.files
4.0K  ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome/idb/1451318868ntouromlalnodry--epcr.files
18M   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent/chrome
18M   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/permanent
20K   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/default/https+++shrlb.com/ls
20K   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/default/https+++shrlb.com
16K   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/default/https+++tsdtocl.com^partitionKey=%28https%2Cshrib.co
m%29
24K   ./snap/firefox/common/.mozilla/firefox/za5ixaza.default/storage/default/https+++tsdtocl.com^partitionKey=%28https%2Cshrib.co
m%29
```

CLOUD COMPUTING ASSIGNMENT 01

31) df (Disk Free):

Description: Displays the amount of disk space available on the file system.

Example Usage: df -h



A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and other applications. In the center is a terminal window titled 'Terminal' with the command 'df -h' running. The output shows disk usage for various file systems:

```
vboxuser@VM1: ~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           382M   1.5M  380M  1% /run
/dev/sda3        24G   14G  9.2G  60% /
tmpfs           1.9G    0  1.9G  0% /dev/shm
tmpfs           5.0M  4.0K  5.0M  1% /run/lock
/dev/sda2        512M  6.1M  506M  2% /boot/efi
tmpfs           382M  108K  382M  1% /run/user/1000
vboxuser@VM1: ~$
```

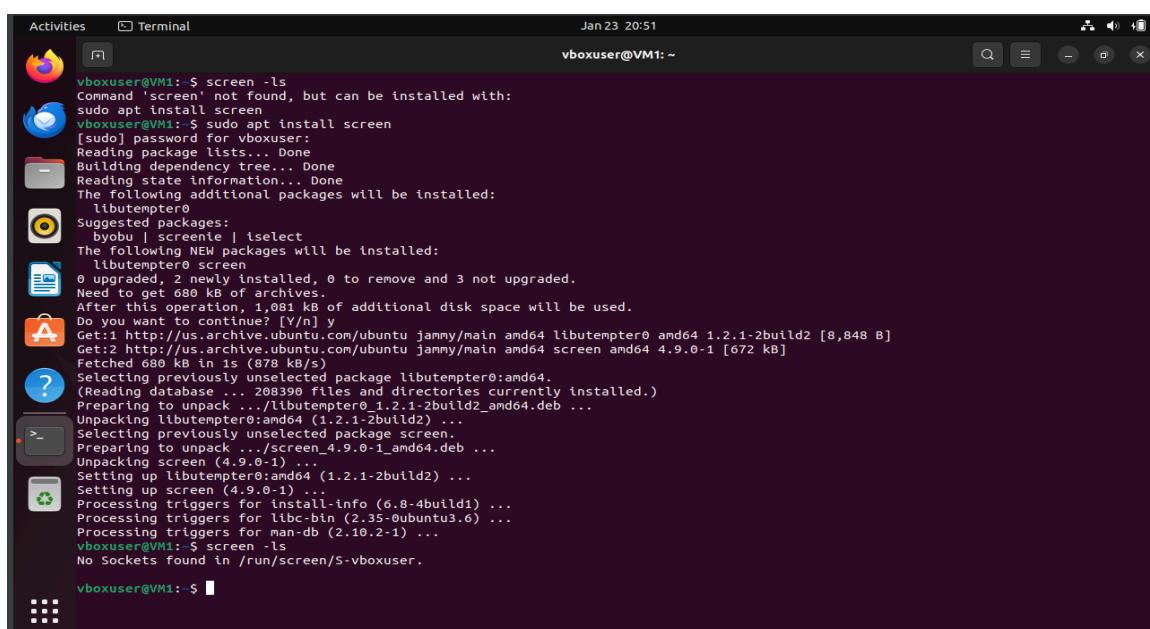
32) screen:

Description: Multiplexes a physical terminal between several processes.

Example Usage:

To start a new screen session: screen

To detach from a screen session: Press Ctrl-a followed by d



A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and other applications. In the center is a terminal window titled 'Terminal' with the command 'screen -ls' running. The output shows that the 'screen' command was not found, but it can be installed via apt. It then proceeds to install the 'screen' package:

```
vboxuser@VM1: ~$ screen -ls
Command 'screen' not found, but can be installed with:
sudo apt install screen
vboxuser@VM1: ~$ sudo apt install screen
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
libutempter0
Suggested packages:
  bوبو | screenie | iselect
The following NEW packages will be installed:
  libutempter0 screen
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Need to get 680 kB of archives.
After this operation, 1,081 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libutempter0 amd64 1.2.1-2build2 [8,848 B]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 screen amd64 4.9.0-1 [672 kB]
Fetched 680 kB in 1s (875 kB/s)
Selecting previously unselected package libutempter0:amd64.
(Reading database ... 208390 files and directories currently installed.)
Preparing to unpack .../libutempter0_1.2.1-2build2_amd64.deb ...
Unpacking libutempter0:amd64 (1.2.1-2build2) ...
Selecting previously unselected package screen.
Preparing to unpack .../screen_4.9.0-1_amd64.deb ...
Unpacking screen (4.9.0-1) ...
Setting up libutempter0:amd64 (1.2.1-2build2) ...
Setting up screen (4.9.0-1) ...
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
Processing triggers for man-db (2.10.2-1) ...
vboxuser@VM1: ~$ screen -ls
No Sockets found in /run/screen/S-vboxuser.

vboxuser@VM1: ~$
```

CLOUD COMPUTING ASSIGNMENT 01

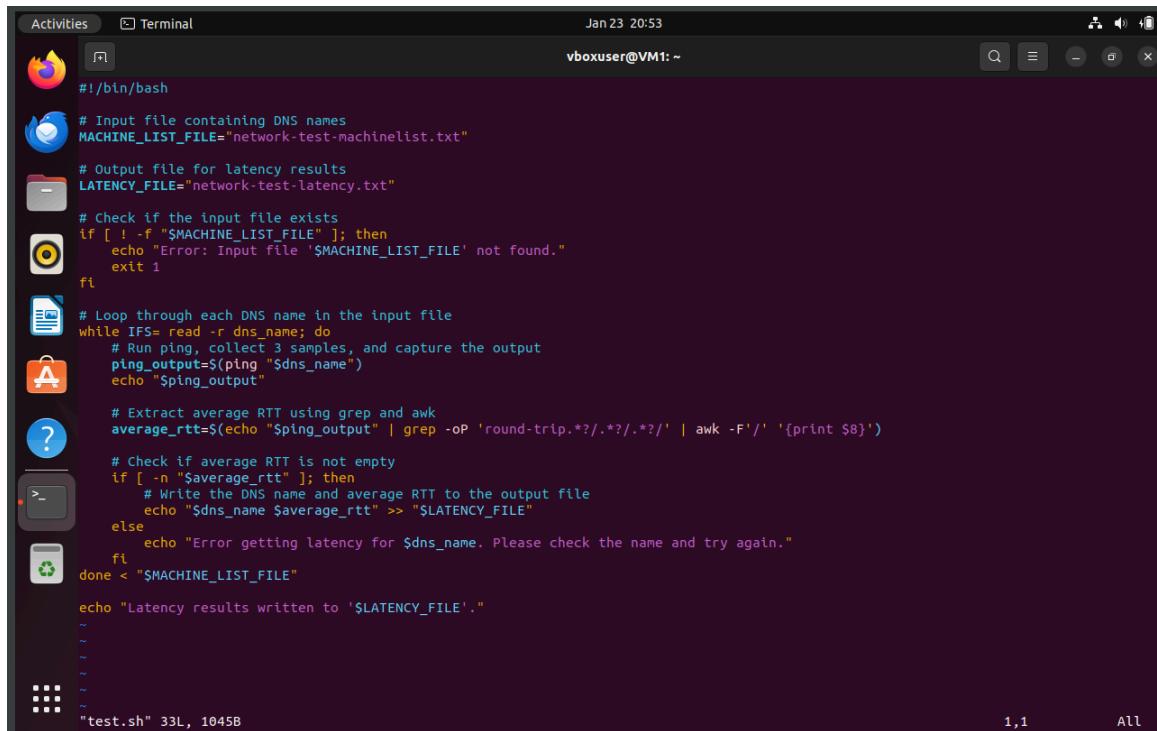
33) vim:

Description: A highly configurable text editor.

Example Usage: vim filename.txt

To enter editing mode, press i.

To save and exit, press Esc, then type :wq and press Enter.



The screenshot shows a terminal window titled "Terminal" with the command "vboxuser@VM1: ~". The window displays the contents of a file named "test.sh". The script performs several tasks: it reads a list of DNS names from a file ("network-test-machinelist.txt"), pings each name, extracts the average round-trip time (RTT) using grep and awk, and writes the results to a file ("network-test-latency.txt"). The terminal window has a dark background and light-colored text. The status bar at the bottom right shows "1,1 All".

```
#!/bin/bash

# Input file containing DNS names
MACHINE_LIST_FILE="network-test-machinelist.txt"

# Output file for latency results
LATENCY_FILE="network-test-latency.txt"

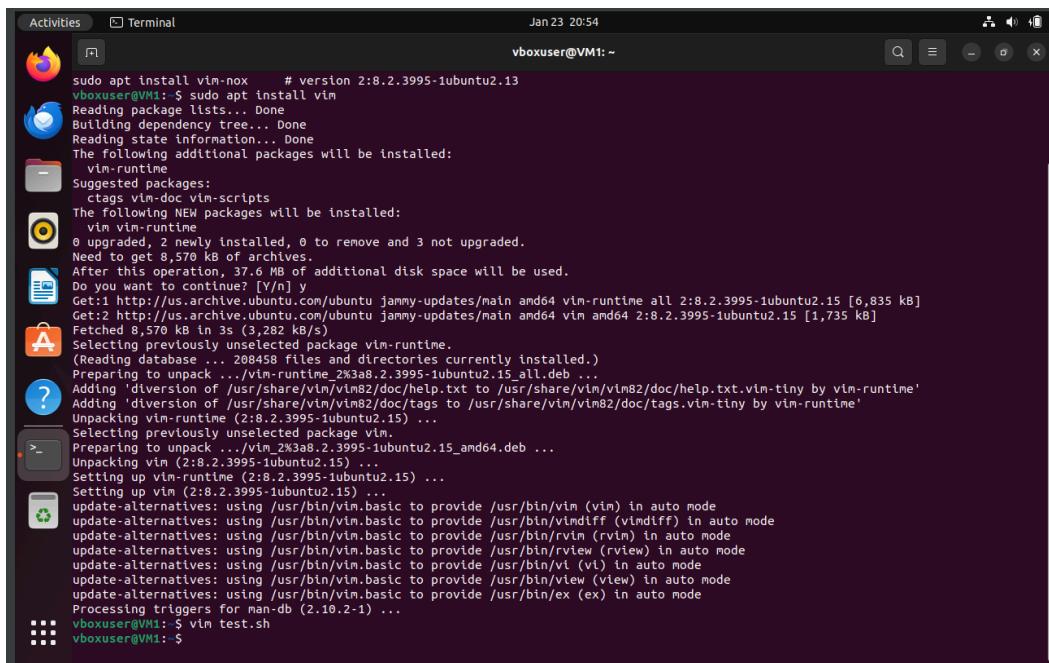
# Check if the input file exists
if [ ! -f "$MACHINE_LIST_FILE" ]; then
    echo "Error: Input file '$MACHINE_LIST_FILE' not found."
    exit 1
fi

# Loop through each DNS name in the input file
while IFS= read -r dns_name; do
    # Run ping, collect 3 samples, and capture the output
    ping_output=$(ping "$dns_name")
    echo "$ping_output"

    # Extract average RTT using grep and awk
    average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/.*?' | awk -F'/' '{print $8}')

    # Check if average RTT is not empty
    if [ -n "$average_rtt" ]; then
        # Write the DNS name and average RTT to the output file
        echo "$dns_name $average_rtt" >> "$LATENCY_FILE"
    else
        echo "Error getting latency for $dns_name. Please check the name and try again."
    fi
done < "$MACHINE_LIST_FILE"

echo "Latency results written to '$LATENCY_FILE'."
```



The screenshot shows a terminal window titled "Terminal" with the command "vboxuser@VM1: ~". The user runs the command "sudo apt install vim-nox" to install the vim editor. The terminal displays the package manager's progress, including reading package lists, building dependency trees, and installing vim-runtime and vim packages. It also shows the download of vim runtime files from the Ubuntu archive. The terminal window has a dark background and light-colored text. The status bar at the bottom right shows "1,1 All".

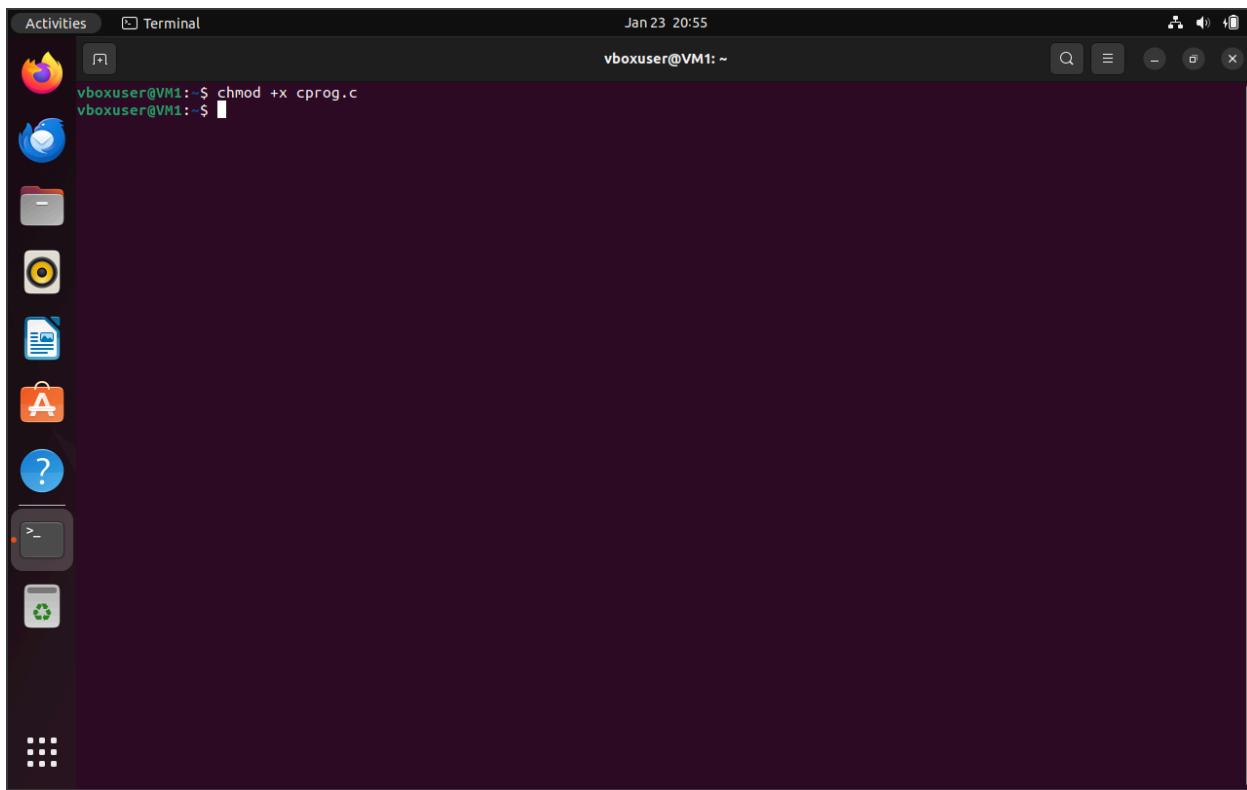
```
sudo apt install vim-nox      # version 2:8.2.3995-1ubuntu2.13
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.
Need to get 8,570 kB of archives.
After this operation, 37.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 vim-runtime all 2:8.2.3995-1ubuntu2.15 [6,835 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Vim amd64 2:8.2.3995-1ubuntu2.15 [1,735 kB]
Fetched 8,570 kB in 3s (3,282 kB/s)
Selecting previously unselected package vim-runtime.
(Reading database ... 208458 files and directories currently installed.)
Preparing to unpack .../vim-runtime_2k3a8.2.3995-1ubuntu2.15_all.deb ...
Adding 'diversion of /usr/share/vim/vim82/doc/help.txt to /usr/share/vim/vim82/doc/help.txt.vim-tiny by vim-runtime'
Adding 'diversion of /usr/share/vim/vim82/doc/tags to /usr/share/vim/vim82/doc/tags.vim-tiny by vim-runtime'
Unpacking vim-runtime (2:8.2.3995-1ubuntu2.15) ...
Selecting previously unselected package vim.
Preparing to unpack .../vim_2k3a8.2.3995-1ubuntu2.15_amd64.deb ...
Unpacking vim (2:8.2.3995-1ubuntu2.15) ...
Setting up vim-runtime (2:8.2.3995-1ubuntu2.15) ...
Setting up vim (2:8.2.3995-1ubuntu2.15) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vl (vl) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode
Processing triggers for man-db (2.10.2-1) ...
vboxuser@VM1: ~ vim test.sh
vboxuser@VM1: ~
```

CLOUD COMPUTING ASSIGNMENT 01

34) chmod(Change Mode):

Description: Changes the permissions of a file or directory.

Example Usage: chmod +x script.sh (grants execute permission to a script)

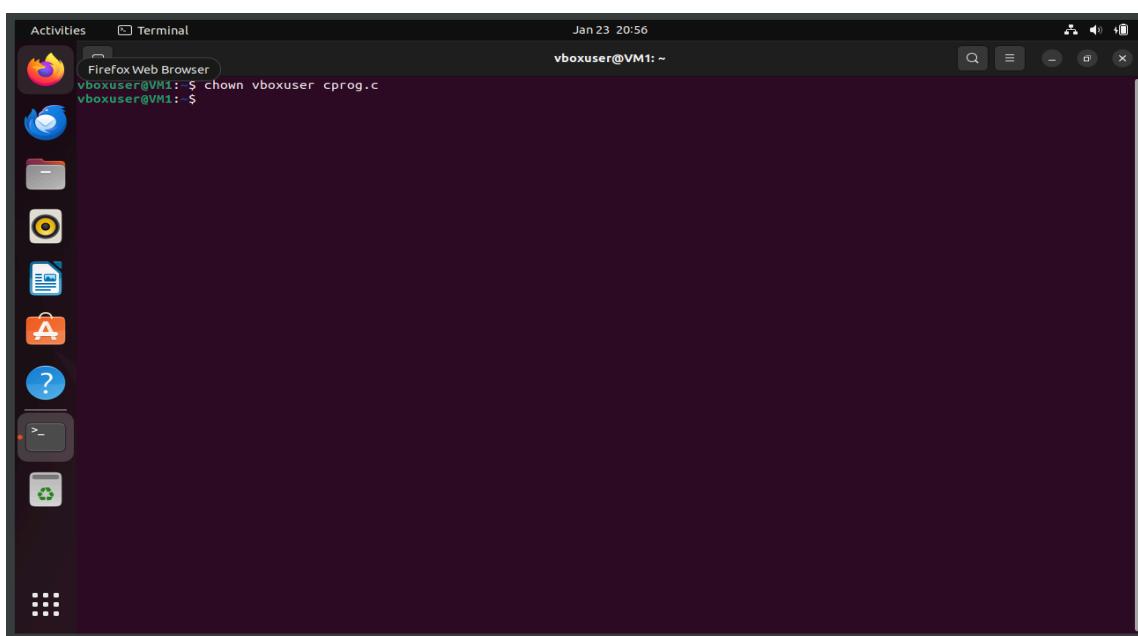


A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications: Activities, Terminal, Home, Dash, Applications, Help, and a terminal icon. The main window is a terminal window titled "Terminal". The title bar shows "Activities" and "Terminal". The status bar at the top right says "Jan 23 20:55" and "vboxuser@VM1: ~". The terminal window contains the command "chmod +x cprog.c" followed by a new line. The desktop background is dark purple.

35) chown(Change Owner):

Description: Changes the owner of a file or directory.

Example Usage: chown new_owner:new_group filename.txt



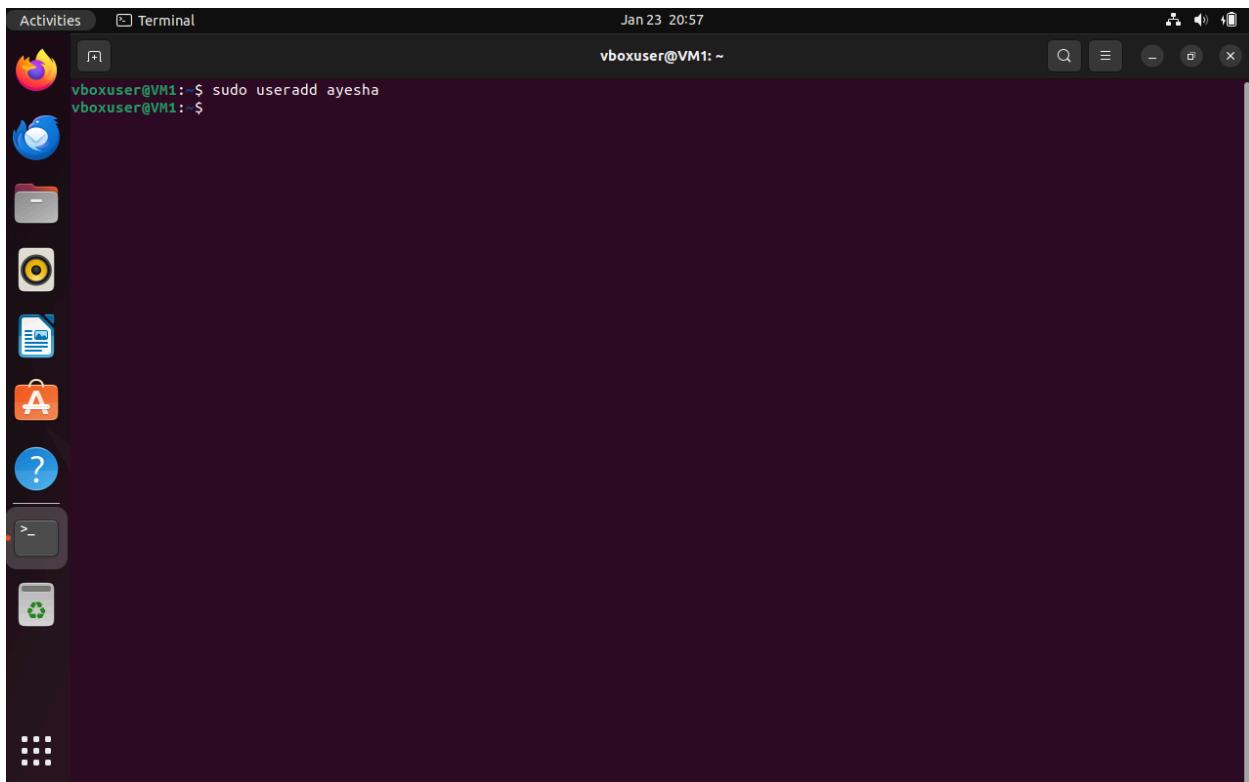
A screenshot of a Linux desktop environment, identical to the one above. The terminal window shows the command "chown vboxuser cprog.c" followed by a new line. The desktop background is dark purple.

CLOUD COMPUTING ASSIGNMENT 01

36) useradd:

Description: Adds a new user to the system.

Example Usage: sudo useradd new_username



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications: Dash (Activities), Terminal, Home, Applications, Help, Dash (Activities), and Recycle Bin. The main window is a terminal window titled 'Terminal' with the title bar 'vboxuser@VM1: ~'. The terminal shows the command 'sudo useradd ayesha' being run, followed by a prompt '\$'. The desktop background is dark purple.

```
vboxuser@VM1: $ sudo useradd ayesha
```

37) mv (Move):

Description: Moves or renames files and directories.

Example Usage: mv old_filename new_filename

CLOUD COMPUTING ASSIGNMENT 01

A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications: a browser (Firefox), a file manager (Nautilus), a terminal (Terminal), a file viewer (Document Viewer), a file manager (File Manager), a help icon (Help), and a system tray (Indicator). The main window is a terminal window titled 'Terminal' with the command 'mv cprog.c prog.c' entered. Below it is another terminal window titled 'vboxuser@VM1: ~' showing the output of the 'ls' command. The desktop background is dark.

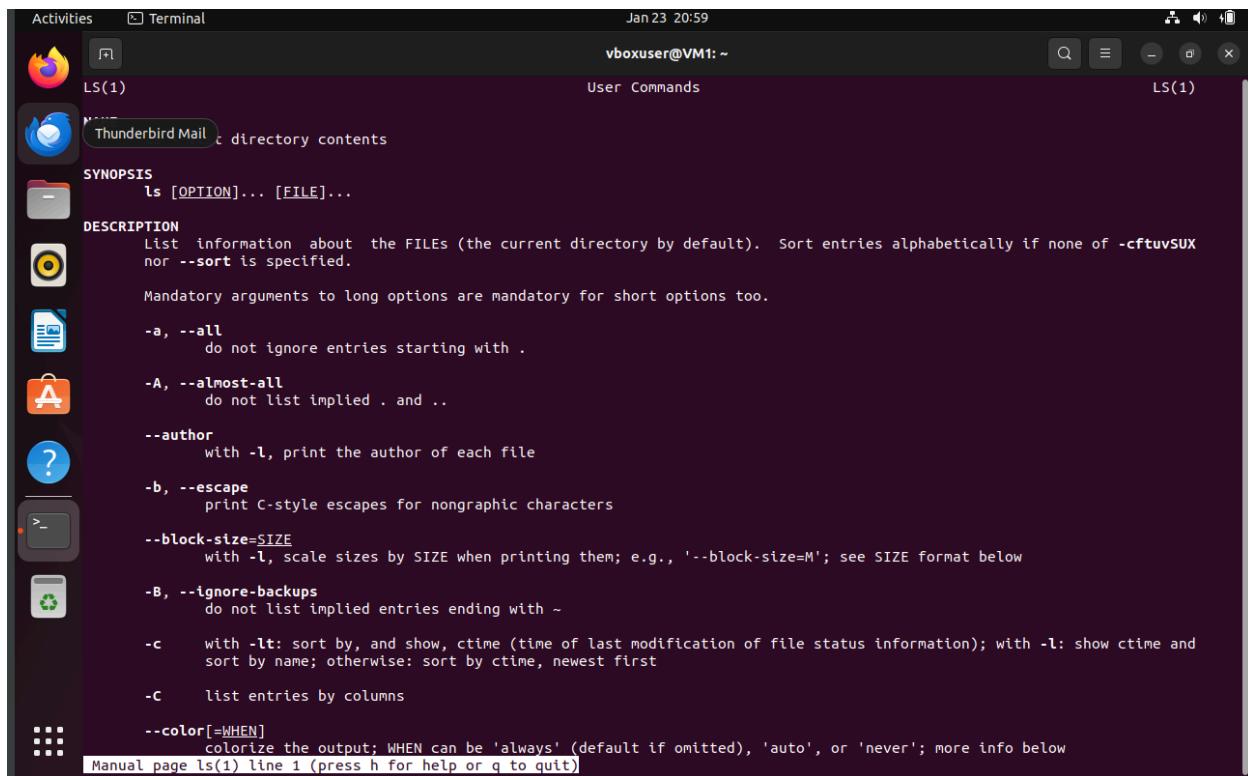
```
vboxuser@VM1:~$ mv cprog.c prog.c
vboxuser@VM1:~$ ls
a.out          copy.sh           Documents        Pictures      script.sh    test.sh
archive.tar    Desktop           Downloads       Public       size_output  Videos
assignment     directory101      Music          snap        Templates
assignment.pub disk-benchmark-background-log.txt network-test-machinelist.txt Q3A.sh
```

38) man (Manual):

Description: Displays the manual page for a command.

Example Usage: man ls

CLOUD COMPUTING ASSIGNMENT 01



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and the subtitle is "Jan 23 20:59". The command entered is "ls(1)". The output is the man page for the ls command, which includes sections for SYNOPSIS, DESCRIPTION, and various options like -a, -A, -b, -c, -l, -t, and --color.

```
SYNOPSIS
ls [OPTION]... [FILE]...

DESCRIPTION
List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all
      do not ignore entries starting with .

-A, --almost-all
      do not list implied . and ..

--author
      with -l, print the author of each file

-b, --escape
      print C-style escapes for nongraphic characters

--block-size=SIZE
      with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

-B, --ignore-backups
      do not list implied entries ending with ~

-c      with -lt: sort by, and show, ctime (time of last modification of file status information); with -l: show ctime and
      sort by name; otherwise: sort by ctime, newest first

-c      list entries by columns

--color[=WHEN]
      colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never'; more info below

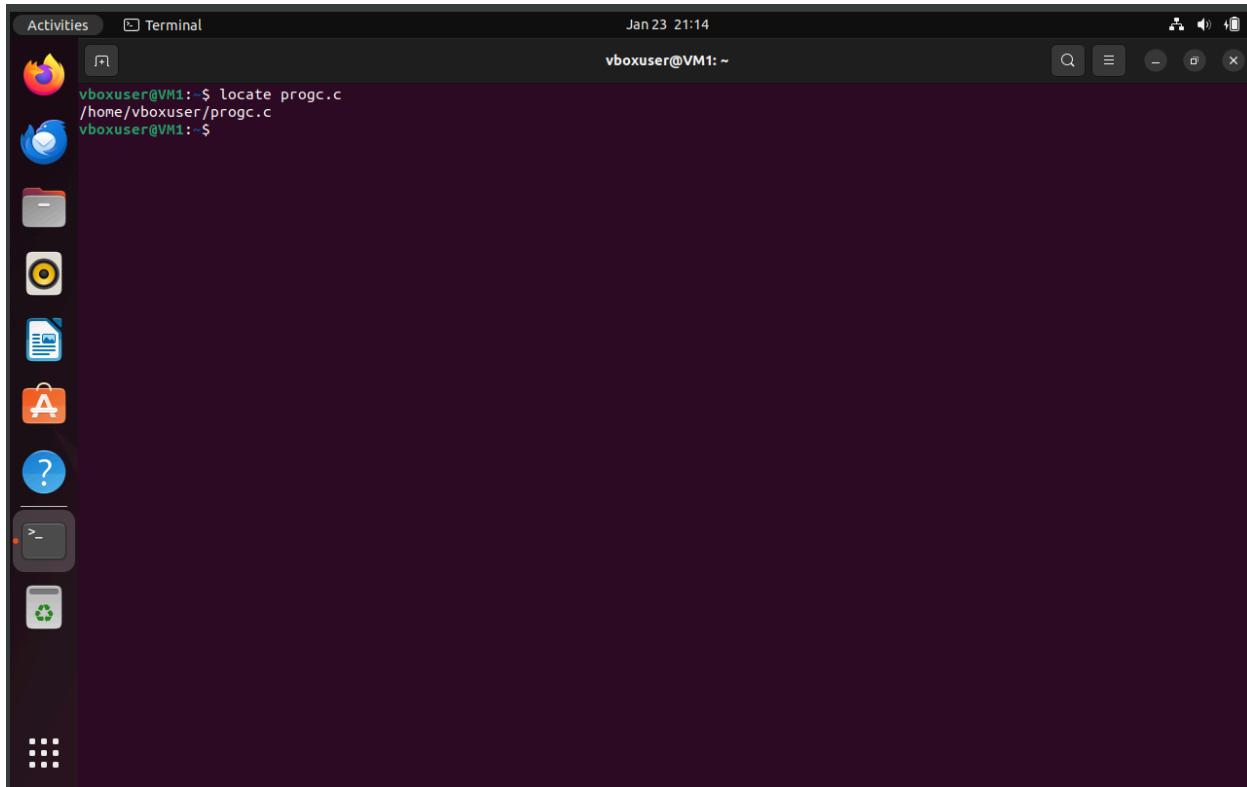
Manual page ls(1) line 1 (press h for help or q to quit)
```

39) locate:

Description: Finds the location of files and directories.E

Example Usage: locate filename

CLOUD COMPUTING ASSIGNMENT 01

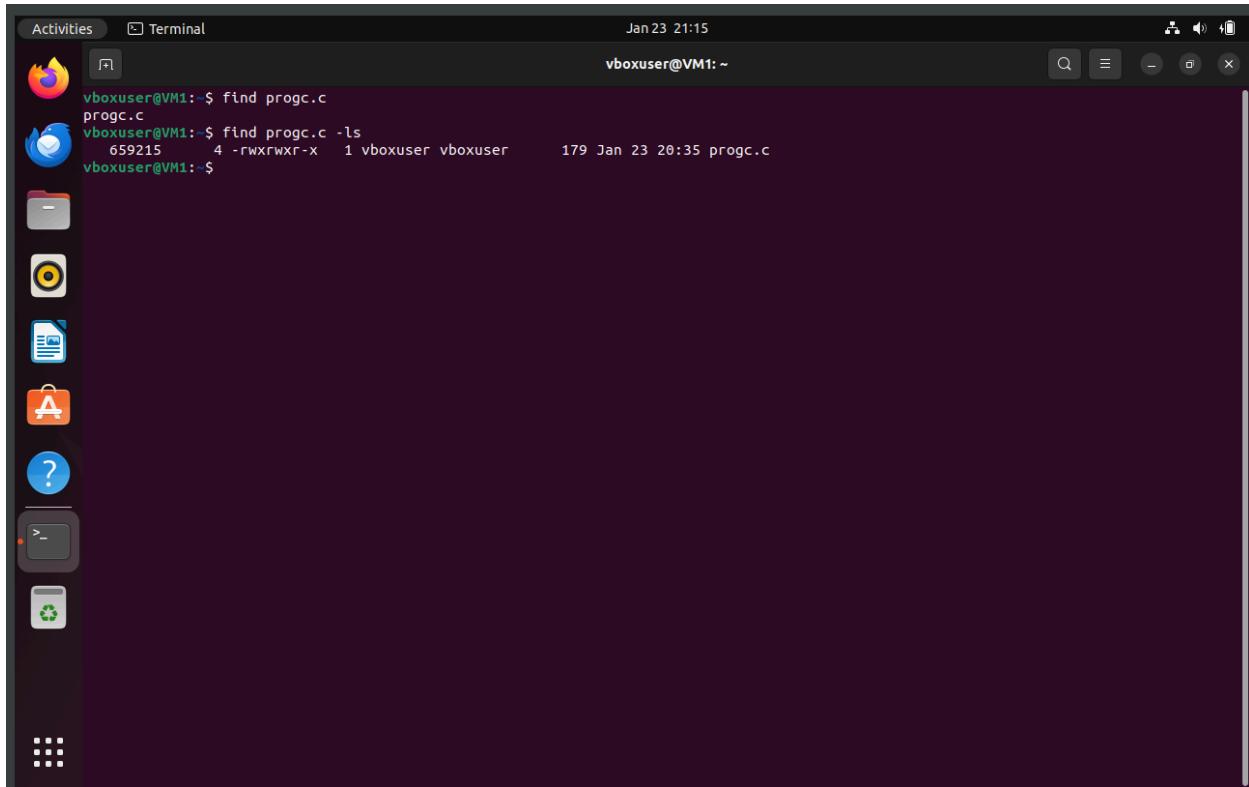


40) find:

Description: Searches for files and directories in a directory hierarchy.

Example Usage: find /path/to/search -name "file_pattern"

CLOUD COMPUTING ASSIGNMENT 01

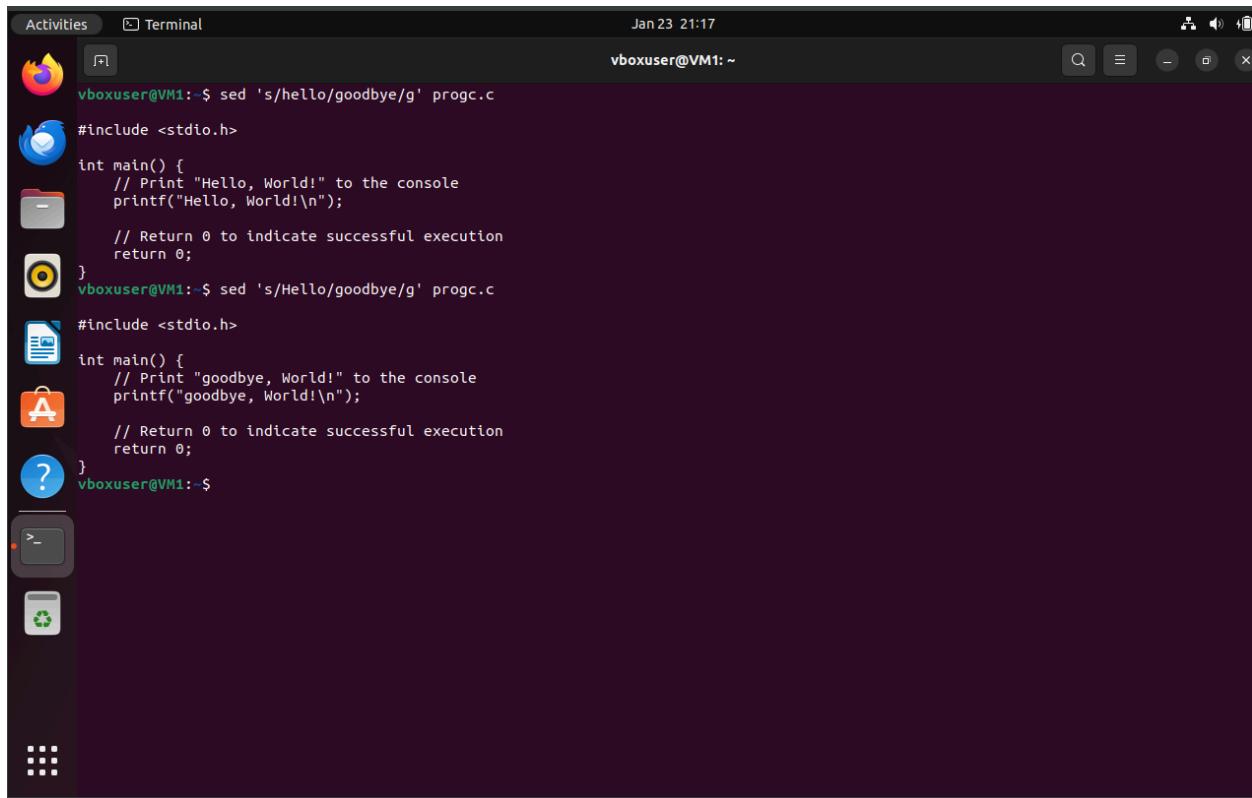


41) sed (Stream Editor):

Description: Edits text by scriptable commands.

Example Usage: `sed 's/old_string/new_string/g' filename.txt`

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and other applications. The main area shows a terminal window titled 'Terminal' with the command 'vboxuser@VM1:~\$'. The terminal displays two code snippets. The first snippet is a C program that prints 'Hello, World!' to the console. The second snippet is a modified version where 'Hello' is replaced by 'goodbye'. Both snippets include comments explaining the purpose of each part.

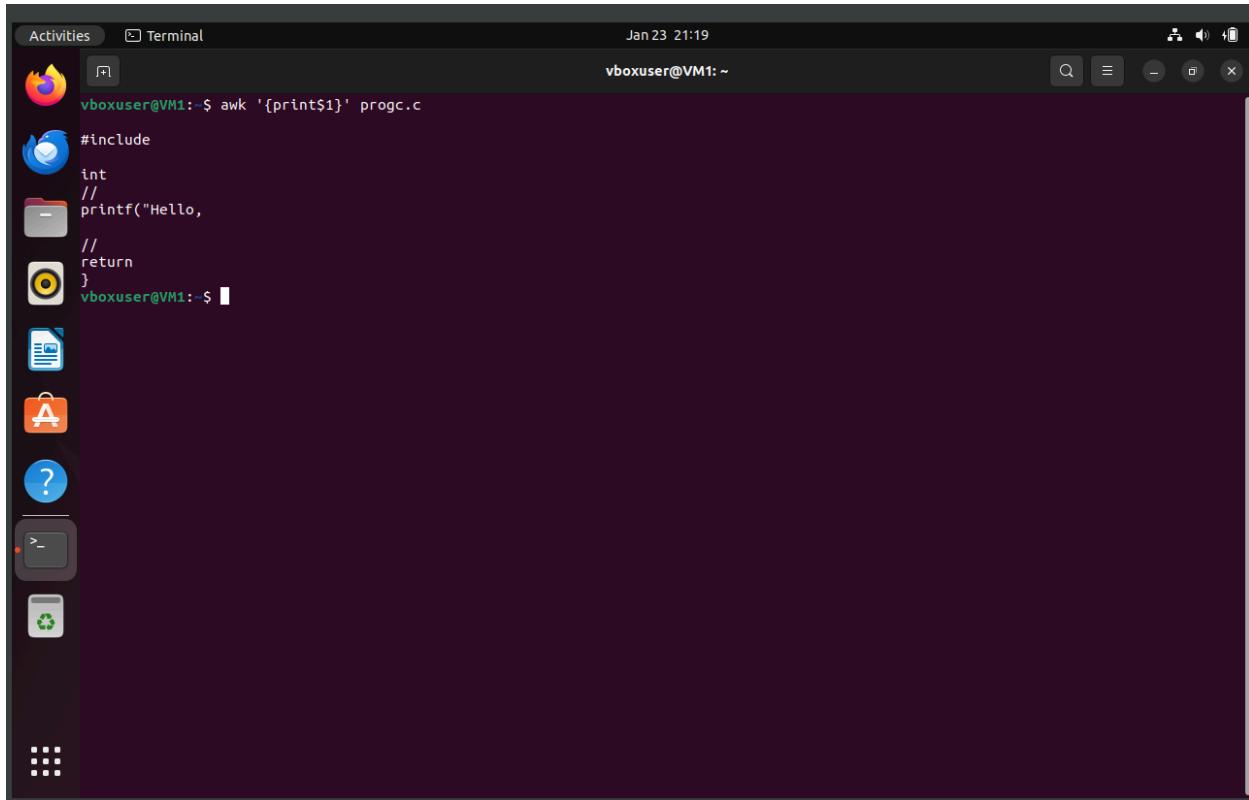
```
vboxuser@VM1:~$ sed 's/hello/goodbye/g' prog.c
#include <stdio.h>
int main() {
    // Print "Hello, World!" to the console
    printf("Hello, World!\n");
    // Return 0 to indicate successful execution
    return 0;
}
vboxuser@VM1:~$ sed 's/Hello/goodbye/g' prog.c
#include <stdio.h>
int main() {
    // Print "goodbye, World!" to the console
    printf("goodbye, World!\n");
    // Return 0 to indicate successful execution
    return 0;
}
```

42) awk:

Description: A text processing tool for pattern scanning and processing.

Example Usage: awk '{print \$1}' filename.txt (prints the first column of a file)

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for a browser (Firefox), file manager, terminal, and other applications. The main area shows a terminal window titled "Terminal" with the command "awk '{print\$1}' progc.c" and its output:

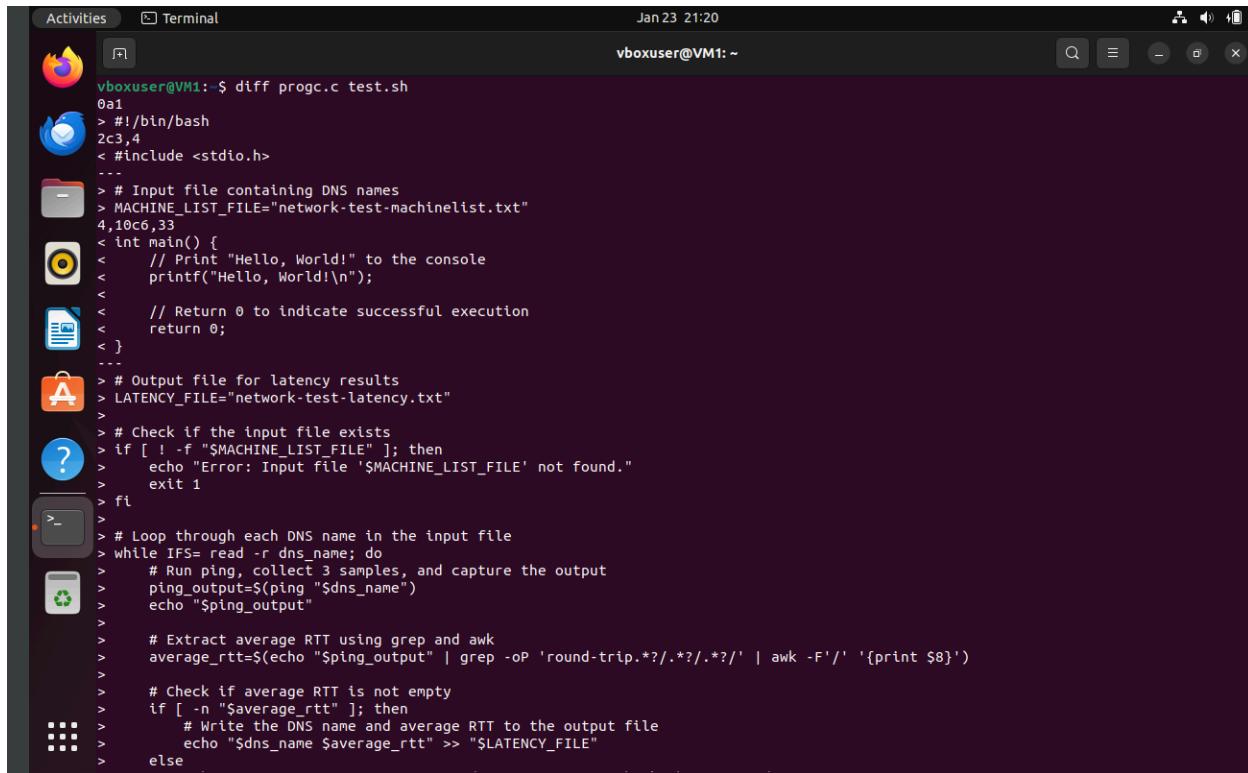
```
vboxuser@VM1:~$ awk '{print$1}' progc.c
#include
int
//
printf("Hello,
//
return
}
vboxuser@VM1:~$
```

43) diff:

Description: Compares files line by line.

Example Usage: diff file1.txt file2.txt

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment showing a terminal window. The terminal title is "Terminal" and the date and time are "Jan 23 21:20". The command run is "diff prog.c test.sh". The output shows the differences between the two files, with lines starting with '>' indicating additions and lines starting with '<' indicating deletions. The code itself is a simple C program that reads a list of DNS names from a file, performs pings, and writes the average round-trip time to another file.

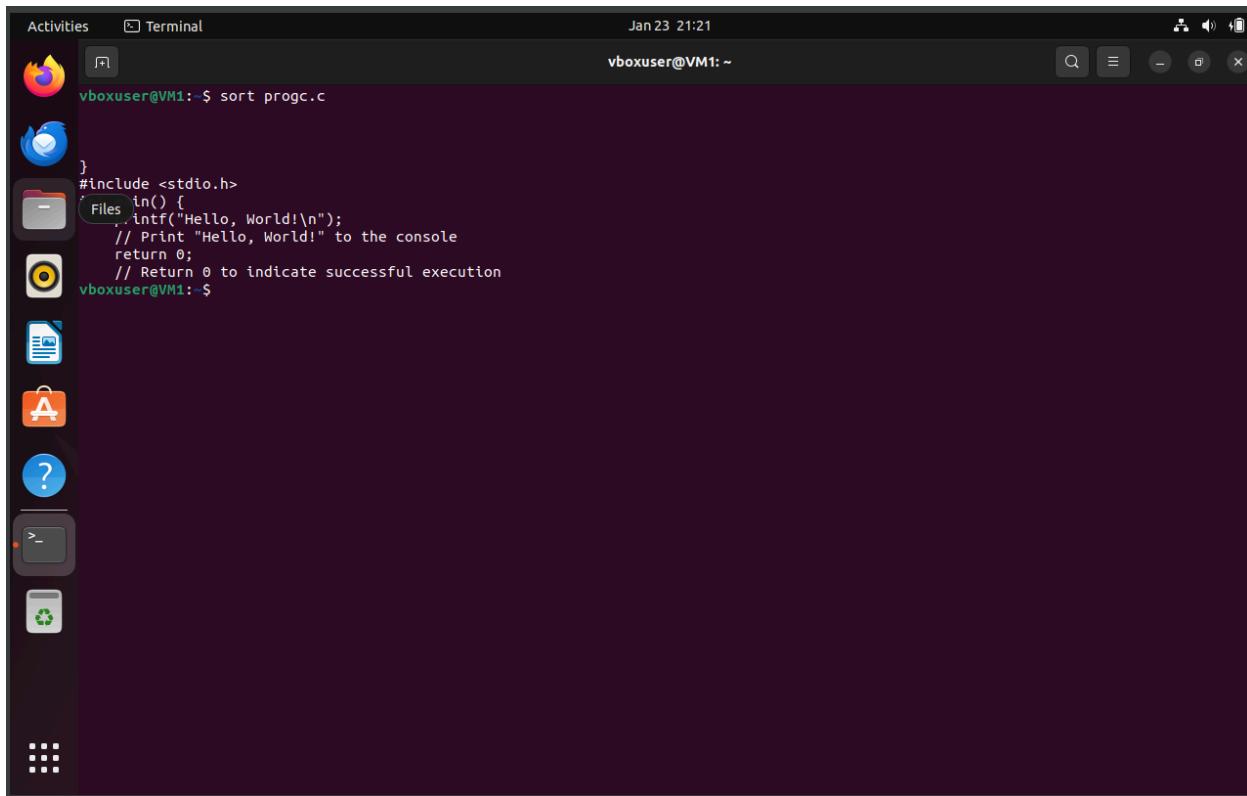
```
vboxuser@VM1:~$ diff prog.c test.sh
0a1
> #!/bin/bash
2c3,4
< #include <stdio.h>
---
> # Input file containing DNS names
> MACHINE_LIST_FILE="network-test-machinelist.txt"
4,10c6,33
< int main() {
<     // Print "Hello, World!" to the console
<     printf("Hello, World!\n");
<
<     // Return 0 to indicate successful execution
<     return 0;
< }
---
> # Output file for latency results
> LATENCY_FILE="network-test-latency.txt"
>
> # Check if the input file exists
> if [ ! -f "$MACHINE_LIST_FILE" ]; then
>     echo "Error: Input file '$MACHINE_LIST_FILE' not found."
>     exit 1
> fi
>
> # Loop through each DNS name in the input file
> while IFS= read -r dns_name; do
>     # Run ping, collect 3 samples, and capture the output
>     ping_output=$(ping "$dns_name")
>     echo "$ping_output"
>
>     # Extract average RTT using grep and awk
>     average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/.*?/' | awk -F'/' '{print $8}')
>
>     # Check if average RTT is not empty
>     if [ -n "$average_rtt" ]; then
>         # Write the DNS name and average RTT to the output file
>         echo "$dns_name $average_rtt" >> "$LATENCY_FILE"
>     else
>
```

44) sort:

Description: Sorts lines of text files.

Example Usage: sort filename.txt

CLOUD COMPUTING ASSIGNMENT 01



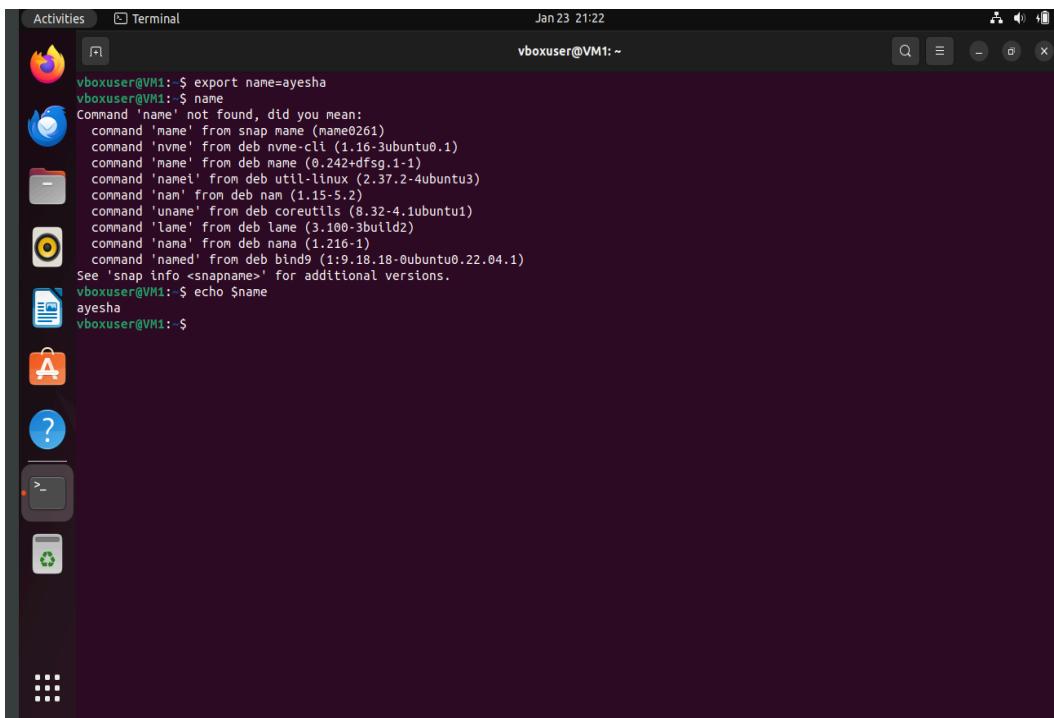
A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Activities, Terminal, Files, App Center, Help, and Dash. The main window is a terminal window titled 'Terminal' with the status bar showing 'Jan 23 21:21' and 'vboxuser@VM1: ~'. The terminal content is:

```
vboxuser@VM1:~$ sort prog.c
}
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    // Print "Hello, World!" to the console
    return 0;
    // Return 0 to indicate successful execution
vboxuser@VM1:~$
```

45) export:

Description: Sets environment variables.

Example Usage: `export VARIABLE_NAME=value`



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Activities, Terminal, Files, App Center, Help, and Dash. The main window is a terminal window titled 'Terminal' with the status bar showing 'Jan 23 21:22' and 'vboxuser@VM1: ~'. The terminal content is:

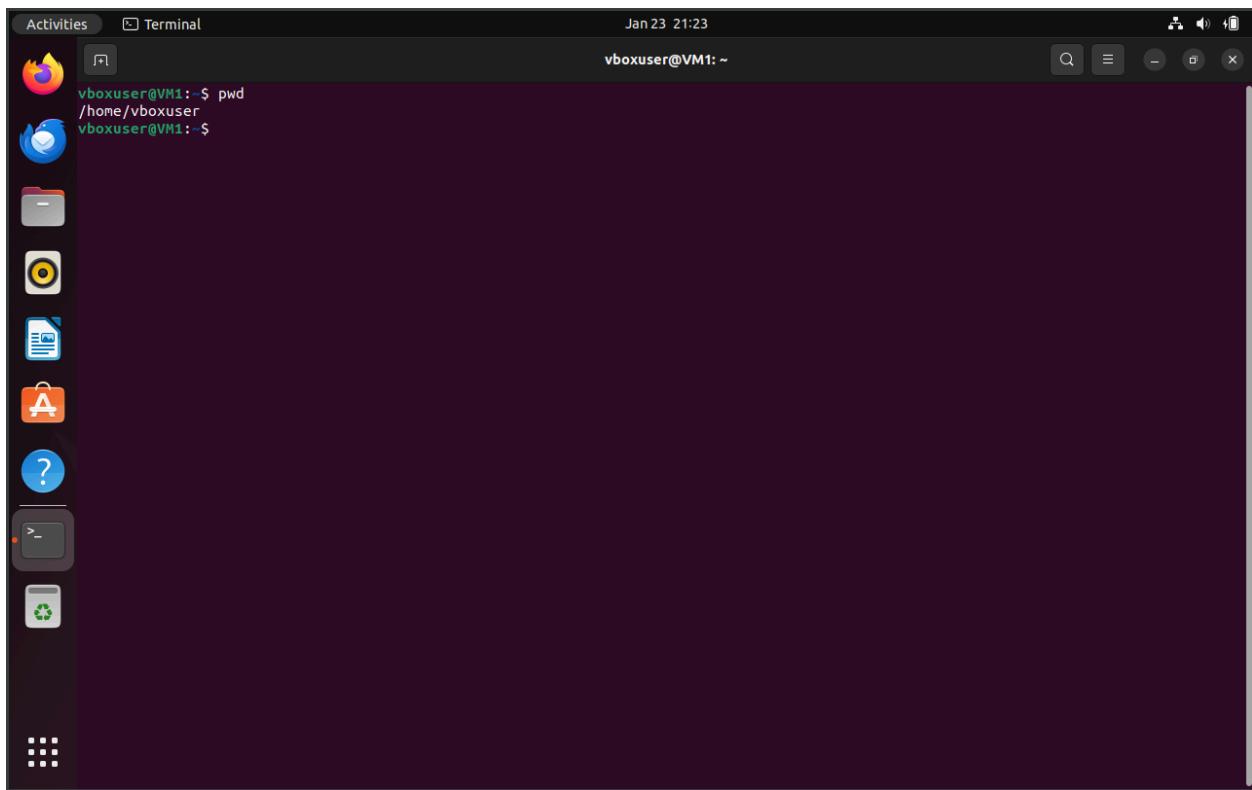
```
vboxuser@VM1:~$ export name=ayesha
vboxuser@VM1:~$ name
Command 'name' not found, did you mean:
  command 'name' from snap mane (name0261)
  command 'nvme' from deb nvme-cli (1.16-3ubuntu0.1)
  command 'name' from deb name (0.242+dfsg.1-1)
  command 'namel' from deb util-linux (2.37.2-4ubuntu3)
  command 'nam' from deb nam (1.15-5.2)
  command 'uname' from deb coreutils (8.32-4.1ubuntu1)
  command 'lame' from deb lame (3.100-3build2)
  command 'nama' from deb nama (1.216-1)
  command 'named' from deb bind9 (1:9.18.18-0ubuntu0.22.04.1)
See 'snap info <snapname>' for additional versions.
vboxuser@VM1:~$ echo $name
ayesha
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

46) `pwd`:

Description: Prints the current working directory.

Example Usage: `pwd`



A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for various applications: a browser (Firefox), file manager, terminal, terminal (selected), system settings, help, and others. The main window is a terminal window titled "Terminal". The title bar shows "Activities" and "Terminal", the date "Jan 23 21:23", and the user "vboxuser@VM1: ~". The terminal window contains the command `pwd` and its output `/home/vboxuser`. The terminal window has a dark background and light-colored text. The desktop background is also dark.

47) `crontab`

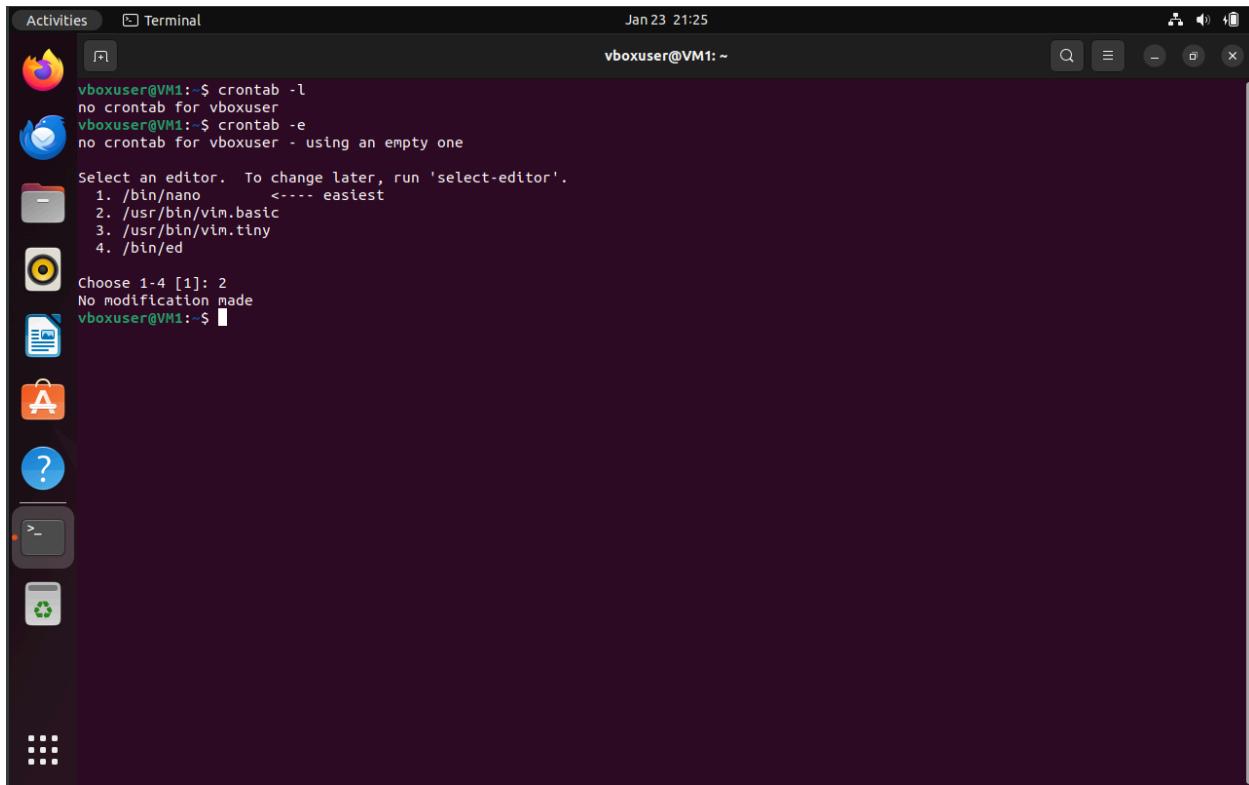
Description: Schedules tasks to run periodically.

Example Usage:

To edit the crontab for the current user: `crontab -e`

To view the crontab for the current user: `crontab -l`

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window and a docked application menu.

The terminal window shows the following command history:

```
vboxuser@VM1:~$ crontab -l
no crontab for vboxuser
vboxuser@VM1:~$ crontab -e
no crontab for vboxuser - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

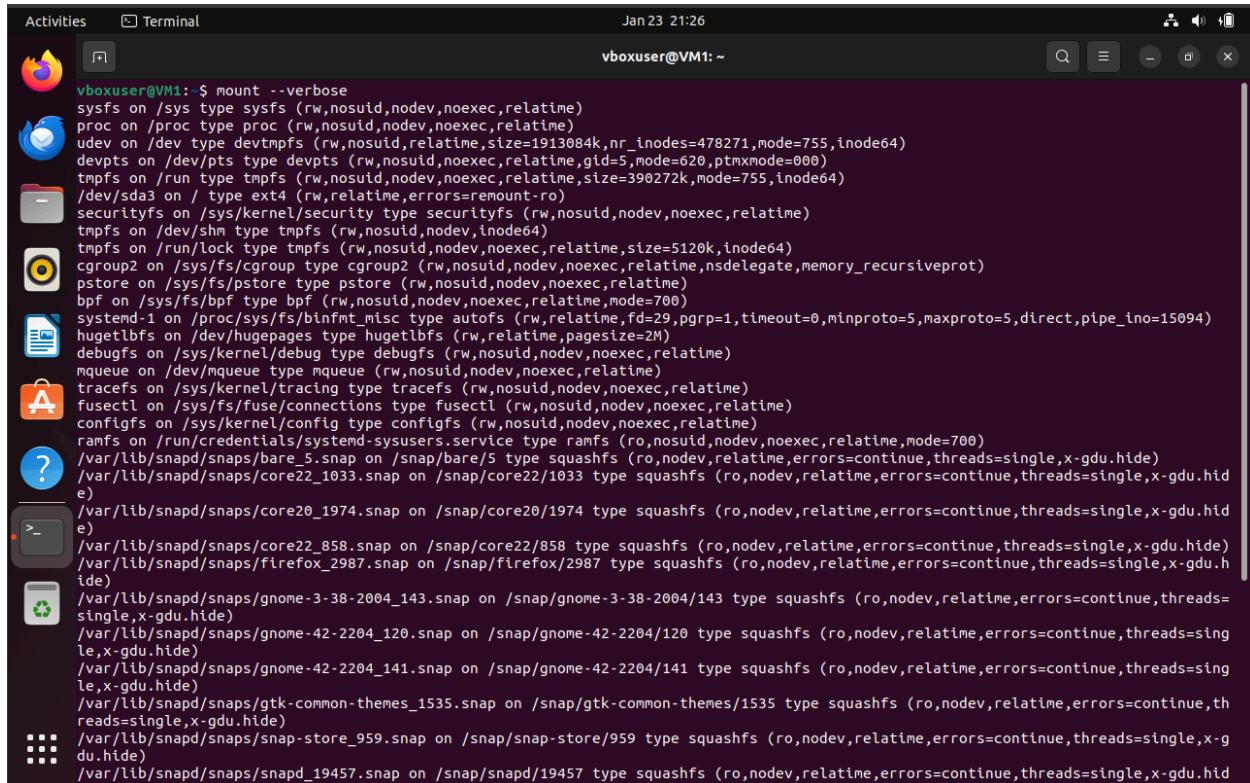
Choose 1-4 [1]: 2
No modification made
vboxuser@VM1:~$
```

48) mount:

Description: Mounts a file system to the directory tree.

Example Usage: sudo mount /dev/sdX1 /mnt

CLOUD COMPUTING ASSIGNMENT 01



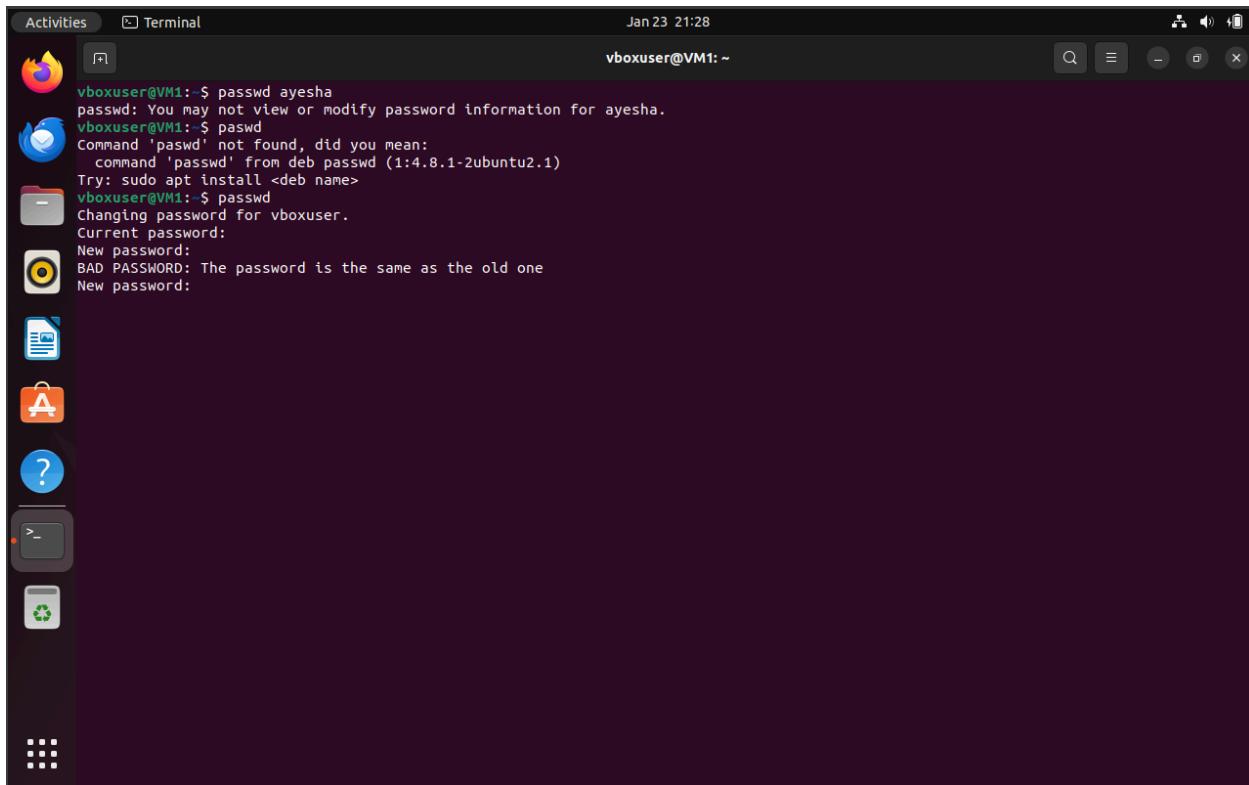
```
vboxuser@VM1: ~$ mount -v
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1913084k,nr_inodes=478271,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,node=620,ptmxnode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=390272k,mode=755,inode64)
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=29,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=15094)
hugepages on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
mqqueue on /dev/mqueue type mqqueue (rw,nosuid,nodev,noexec,relatime)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,nosuid,nodev,noexec,relatime)
configfs on /sys/kernel/config type configfs (rw,nosuid,nodev,noexec,relatime)
ramfs on /run/credentials/systemd-sysusers.service type ramfs (ro,nosuid,nodev,noexec,relatime,mode=700)
/var/lib/snapd/snaps/bare_5.snap on /snap/bare/5 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/core22_1033.snap on /snap/core22/1033 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/core20_1974.snap on /snap/core20/1974 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/core22_858.snap on /snap/core22/858 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/firefox_2987.snap on /snap/firefox/2987 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/gnome-3-38-2004_143.snap on /snap/gnome-3-38-2004/143 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/gnome-42-2204_120.snap on /snap/gnome-42-2204/120 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/gnome-42-2204_141.snap on /snap/gnome-42-2204/141 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/gtk-common-themes_1535.snap on /snap/gtk-common-themes/1535 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/snap-store_959.snap on /snap/snap-store/959 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
/var/lib/snapd/snaps/snapd_19457.snap on /snap/snapd/19457 type squashfs (ro,nodev,relatime,errors=continue,threads=single,x-gdu.hide)
```

49) passwd:

Description: Changes the password for a user.

Example Usage: passwd

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window and a docked application menu.

The terminal window (Activities Terminal) shows the following command-line session:

```
vboxuser@VM1:~$ passwd ayesha
passwd: You may not view or modify password information for ayesha.
vboxuser@VM1:~$ passwd
Command 'passwd' not found, did you mean:
  command 'passwd' from deb passwd (1:4.8.1-2ubuntu2.1)
Try: sudo apt install <deb name>
vboxuser@VM1:~$ passwd
Changing password for vboxuser.
Current password:
New password:
BAD PASSWORD: The password is the same as the old one
New password:
```

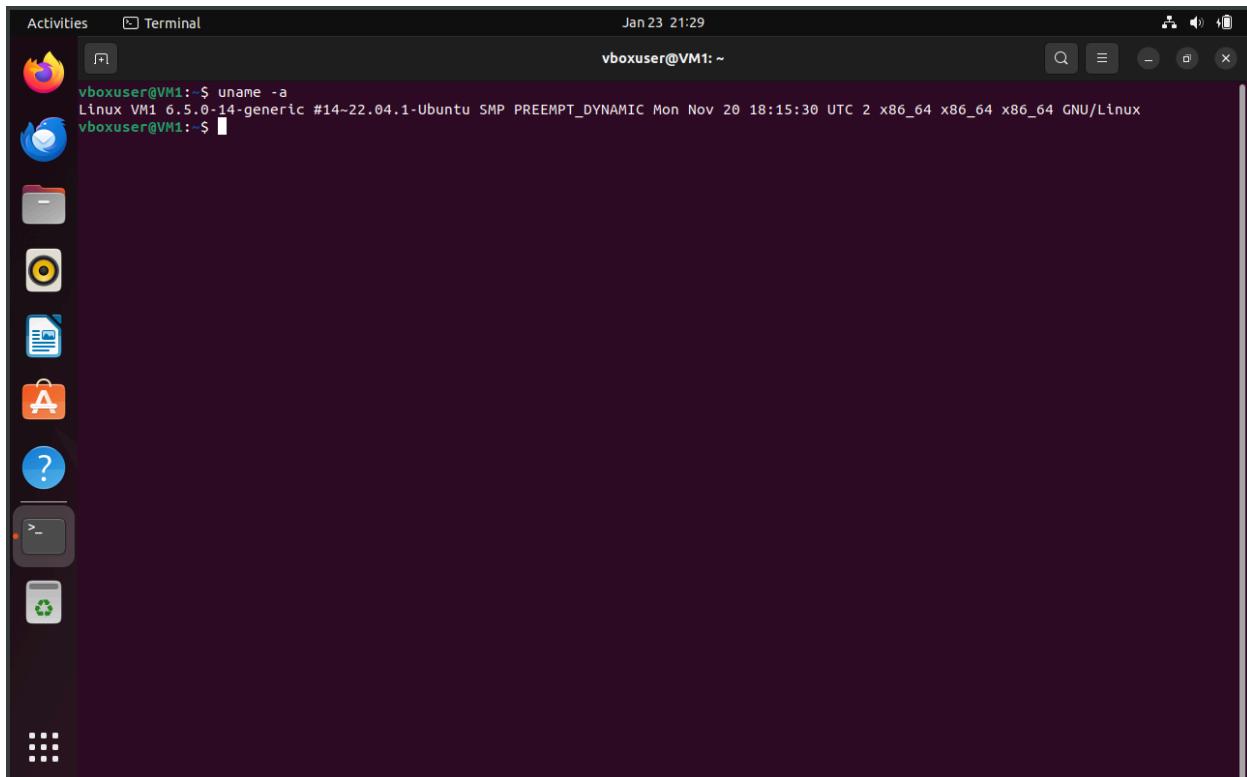
The application menu dock on the left contains icons for various applications, including a browser, file manager, terminal, and system settings.

50) uname (Unix Name):

Description: Displays system information.

Example Usage: `uname -a`

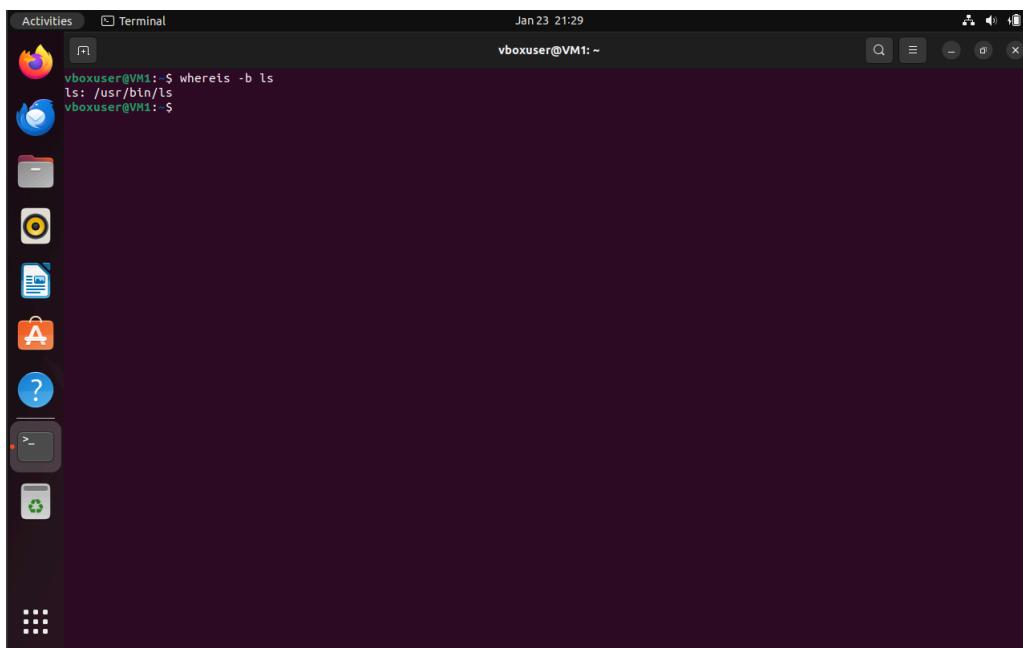
CLOUD COMPUTING ASSIGNMENT 01



51) whereis:

Description: Locates the binary, source, and manual page files for a command.

Example Usage: whereis command

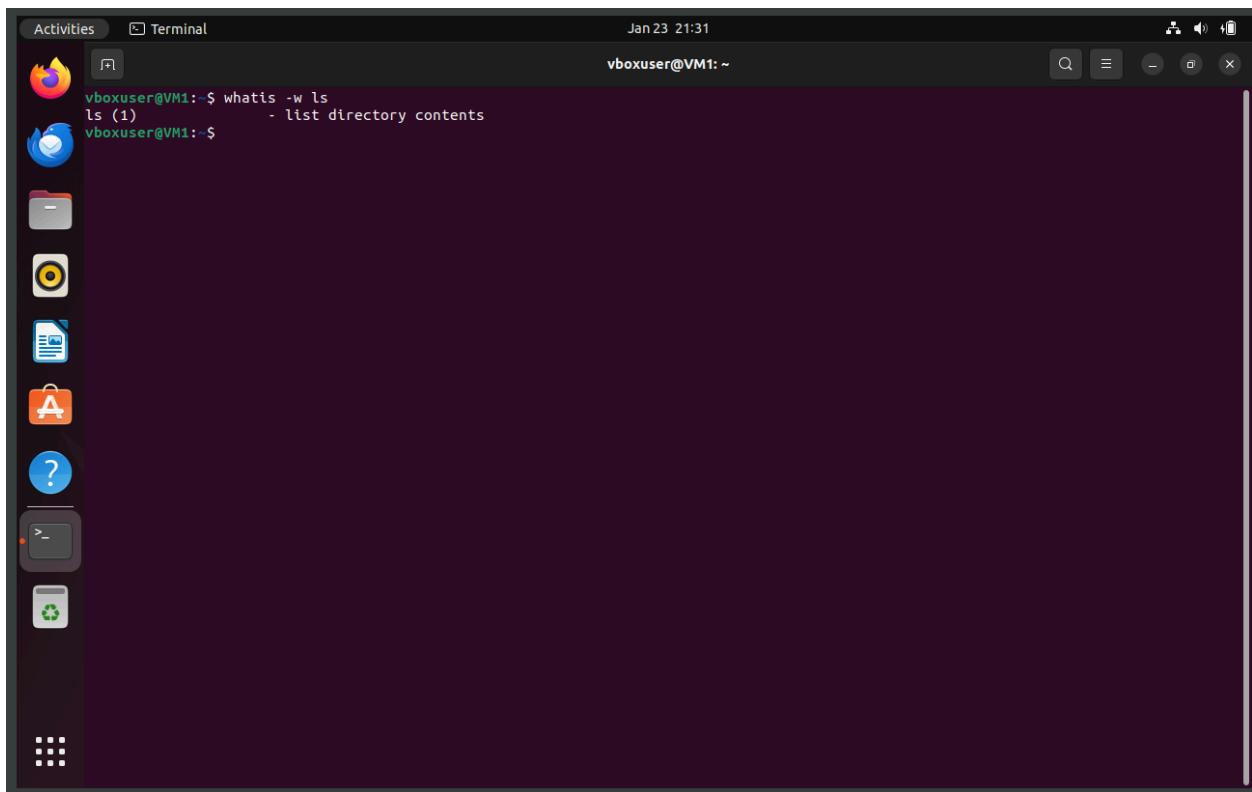


CLOUD COMPUTING ASSIGNMENT 01

52) what is:

Description: Displays a one-line description for a command.

Example Usage: whatis command

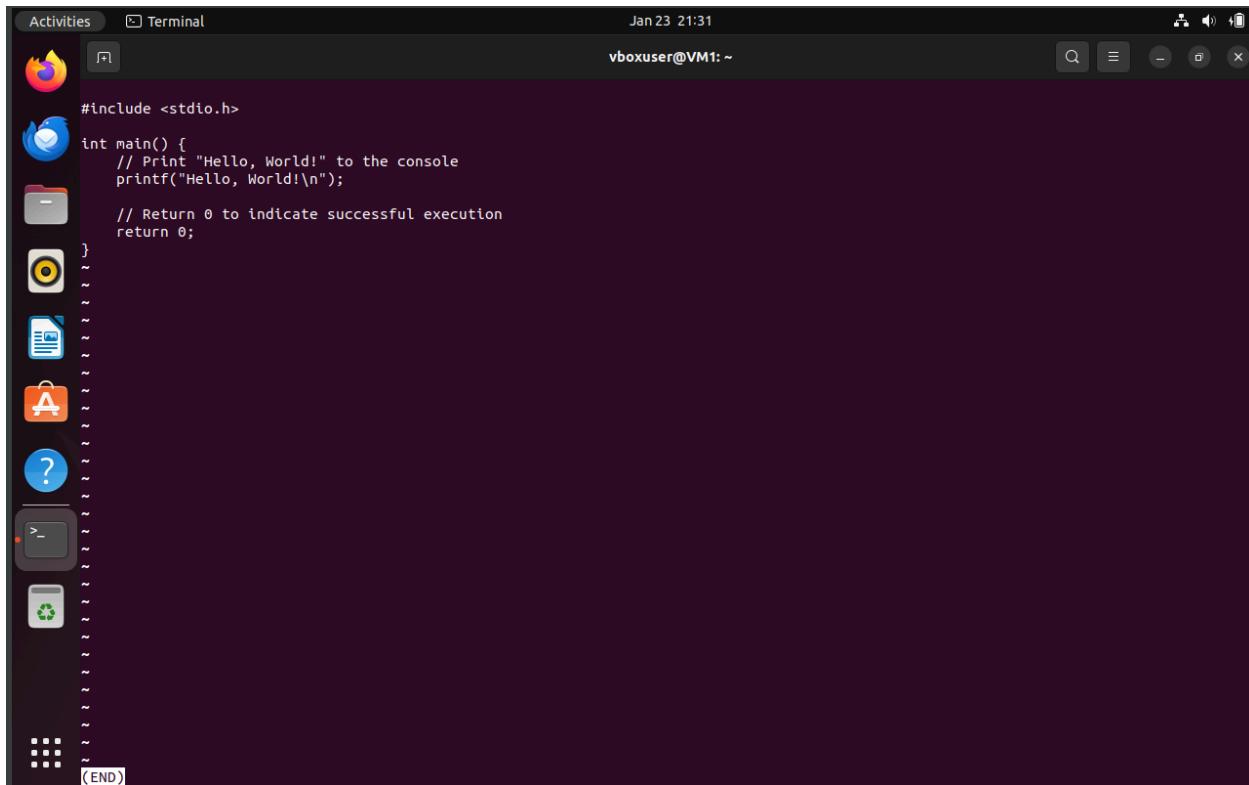


53) Less:

Description: A paginator for viewing text files

Example Usage: less filename.txt

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment. The terminal window is open and shows the following code:

```
#include <stdio.h>
int main() {
    // Print "Hello, World!" to the console
    printf("Hello, World!\n");
    // Return 0 to indicate successful execution
    return 0;
}
```

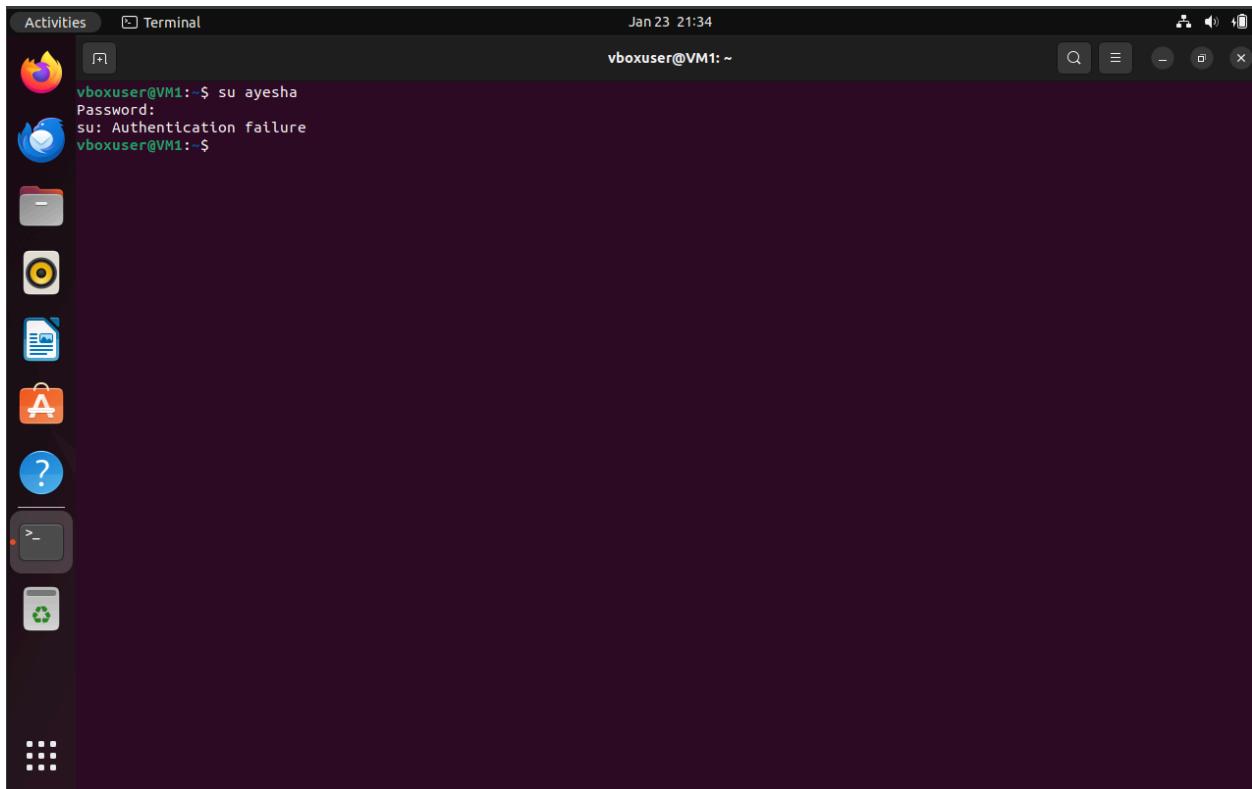
The terminal window title is "Terminal" and the prompt is "vboxuser@VM1: ~". The desktop interface includes a dock with various icons for applications like a web browser, file manager, terminal, and system tools.

54) bsu (Switch User):

Description: Changes user identity to another user.

Example Usage: su username

CLOUD COMPUTING ASSIGNMENT 01

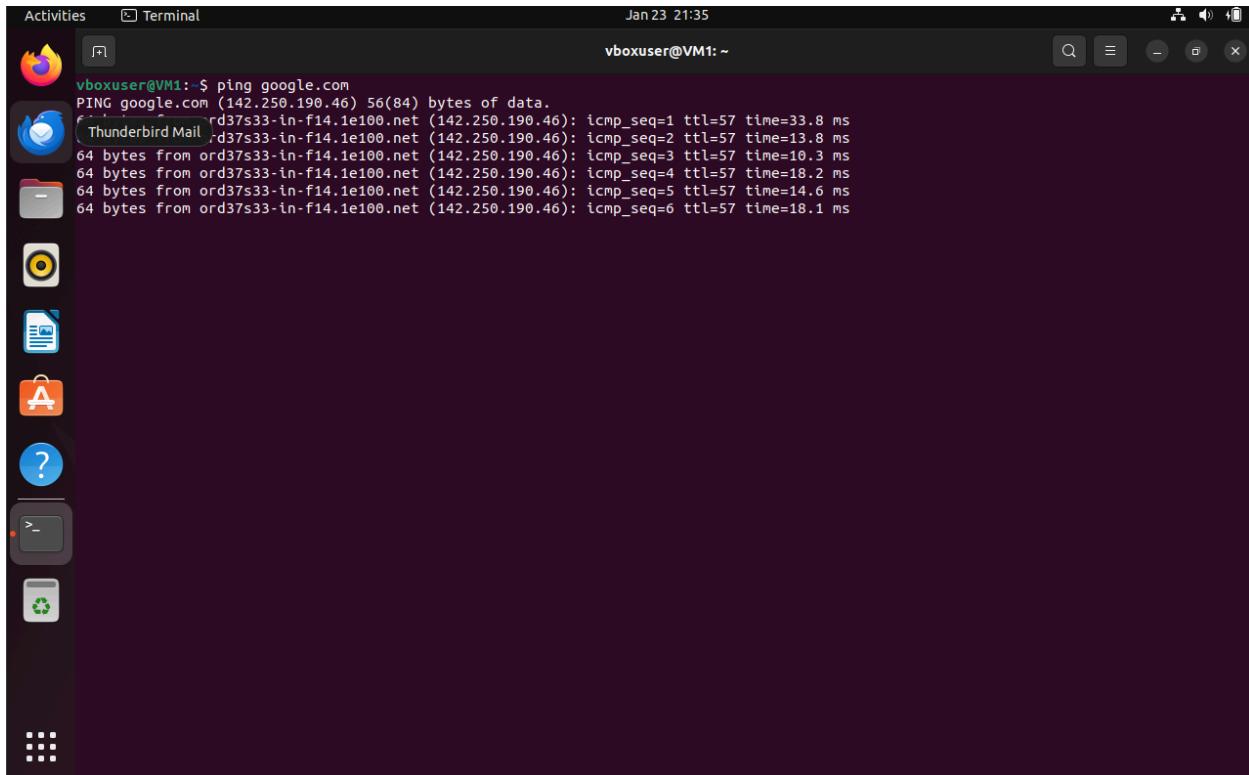


55) ping:

Description: Tests the reachability of a host on a network.

Example Usage: ping example.com

CLOUD COMPUTING ASSIGNMENT 01



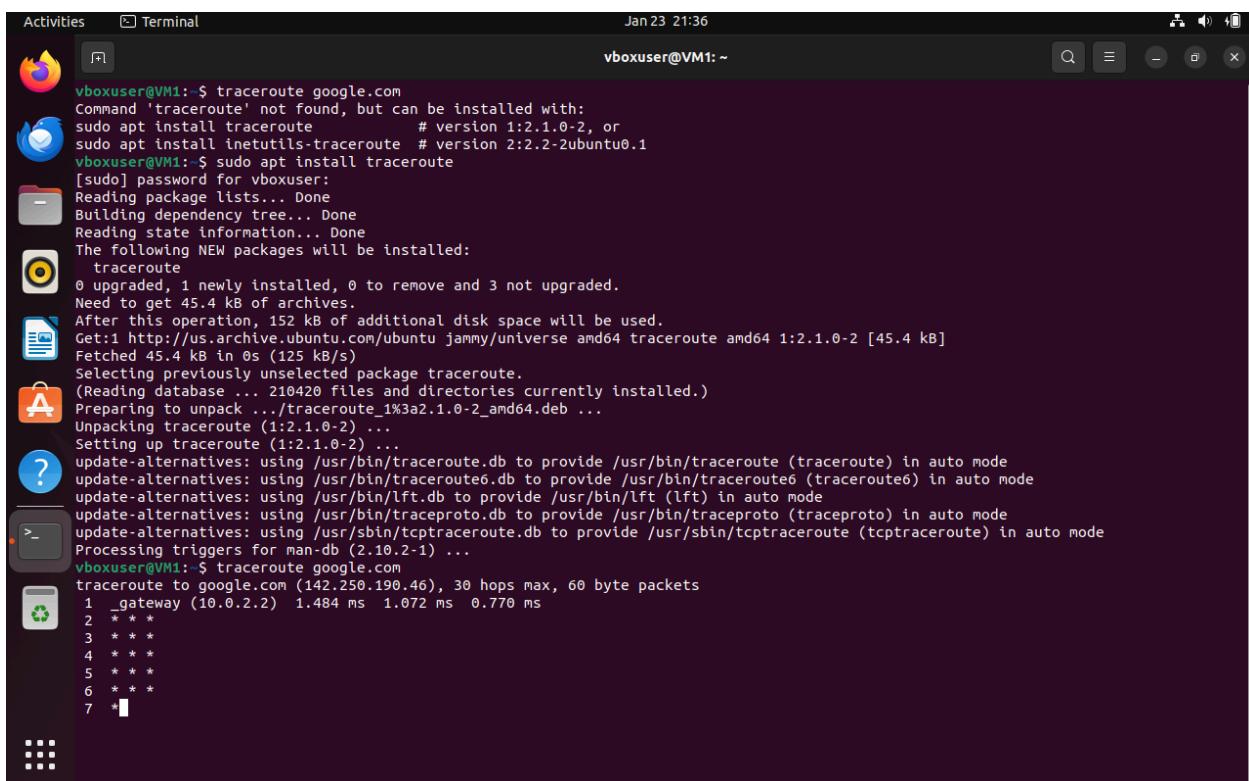
A screenshot of a Linux desktop environment. The terminal window shows the command `ping google.com` being run, with the output indicating a successful ping to an IP address (142.250.190.46) through several routers (ord37s33-in-f14.1e100.net). The desktop interface includes a dock with icons for file manager, terminal, browser, and system tools.

```
vboxuser@VM1:~$ ping google.com
PING google.com (142.250.190.46) 56(84) bytes of data.
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=1 ttl=57 time=33.8 ms
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=2 ttl=57 time=13.8 ms
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=3 ttl=57 time=10.3 ms
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=4 ttl=57 time=18.2 ms
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=5 ttl=57 time=14.6 ms
64 bytes from ord37s33-in-f14.1e100.net (142.250.190.46): icmp_seq=6 ttl=57 time=18.1 ms
```

56) traceroute:

Description: Traces the route that packets take to reach a network host.

Example Usage: traceroute example.com



A screenshot of a Linux desktop environment. The terminal window shows the command `traceroute google.com` being run. It first attempts to install the package, then shows the traceroute path to Google's IP address (142.250.190.46) through various routers. The desktop interface includes a dock with icons for file manager, terminal, browser, and system tools.

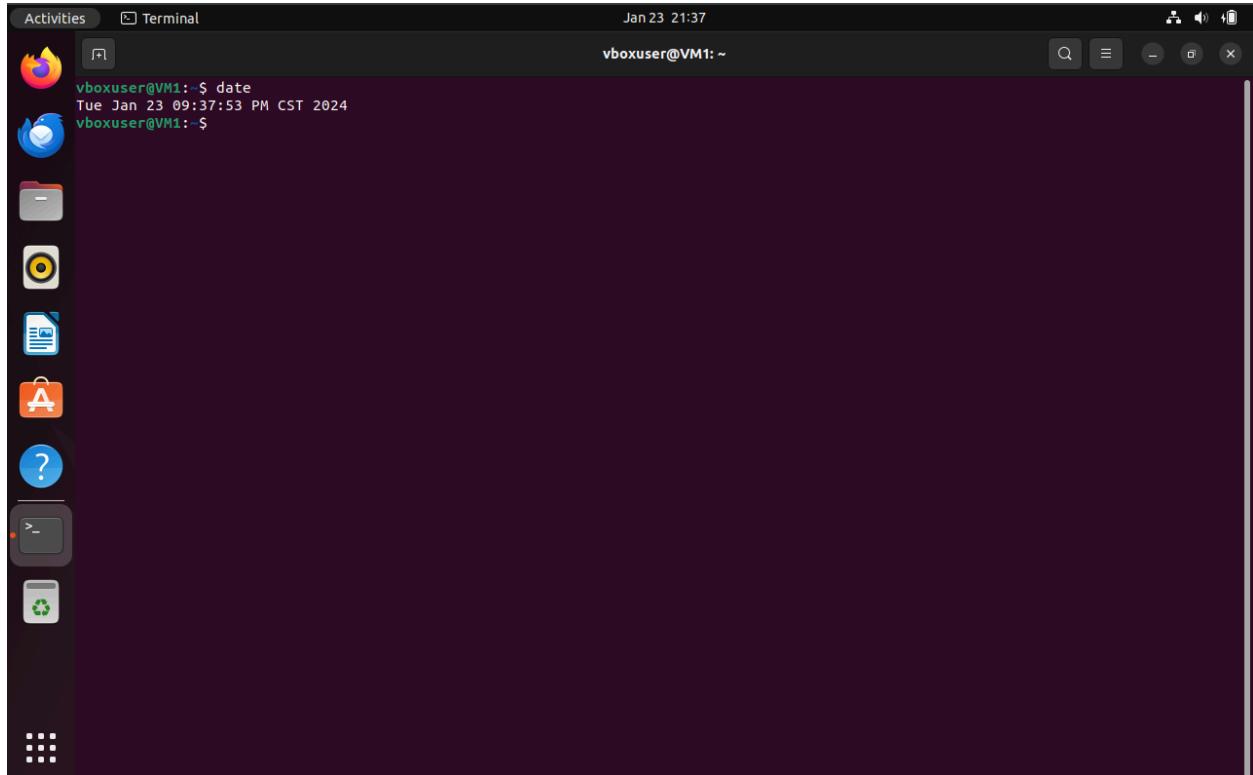
```
vboxuser@VM1:~$ traceroute google.com
Command 'traceroute' not found, but can be installed with:
sudo apt install traceroute          # version 1:2.1.0-2, or
sudo apt install inetutils-traceroute # version 2:2.2-2ubuntu0.1
vboxuser@VM1:~$ sudo apt install traceroute
[sudo] password for vboxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  traceroute
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 45.4 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 traceroute amd64 1:2.1.0-2 [45.4 kB]
Fetched 45.4 kB in 0s (125 KB/s)
Selecting previously unselected package traceroute.
(Reading database ... 210420 files and directories currently installed.)
Preparing to unpack .../traceroute_1%3a2.1.0-2_amd64.deb ...
Unpacking traceroute (1:2.1.0-2) ...
Setting up traceroute (1:2.1.0-2) ...
update-alternatives: using /usr/bin/traceroute.db to provide /usr/bin/traceroute (traceroute) in auto mode
update-alternatives: using /usr/bin/traceroute6.db to provide /usr/bin/traceroute6 (traceroute6) in auto mode
update-alternatives: using /usr/bin/lft.db to provide /usr/bin/lft (lft) in auto mode
update-alternatives: using /usr/bin/traceproto.db to provide /usr/bin/traceproto (traceproto) in auto mode
update-alternatives: using /usr/sbin/tcptraceroute.db to provide /usr/sbin/tcptraceroute (tcptraceroute) in auto mode
Processing triggers for man-db (2.10.2-1) ...
vboxuser@VM1:~$ traceroute google.com
traceroute to google.com (142.250.190.46), 30 hops max, 60 byte packets
 1 _gateway (10.0.2.2)  1.484 ms  1.072 ms  0.770 ms
 2 * * *
 3 * * *
 4 * * *
 5 * * *
 6 * * *
 7 *
```

CLOUD COMPUTING ASSIGNMENT 01

57) date:

Description: Displays the current date and time.

Example Usage: date

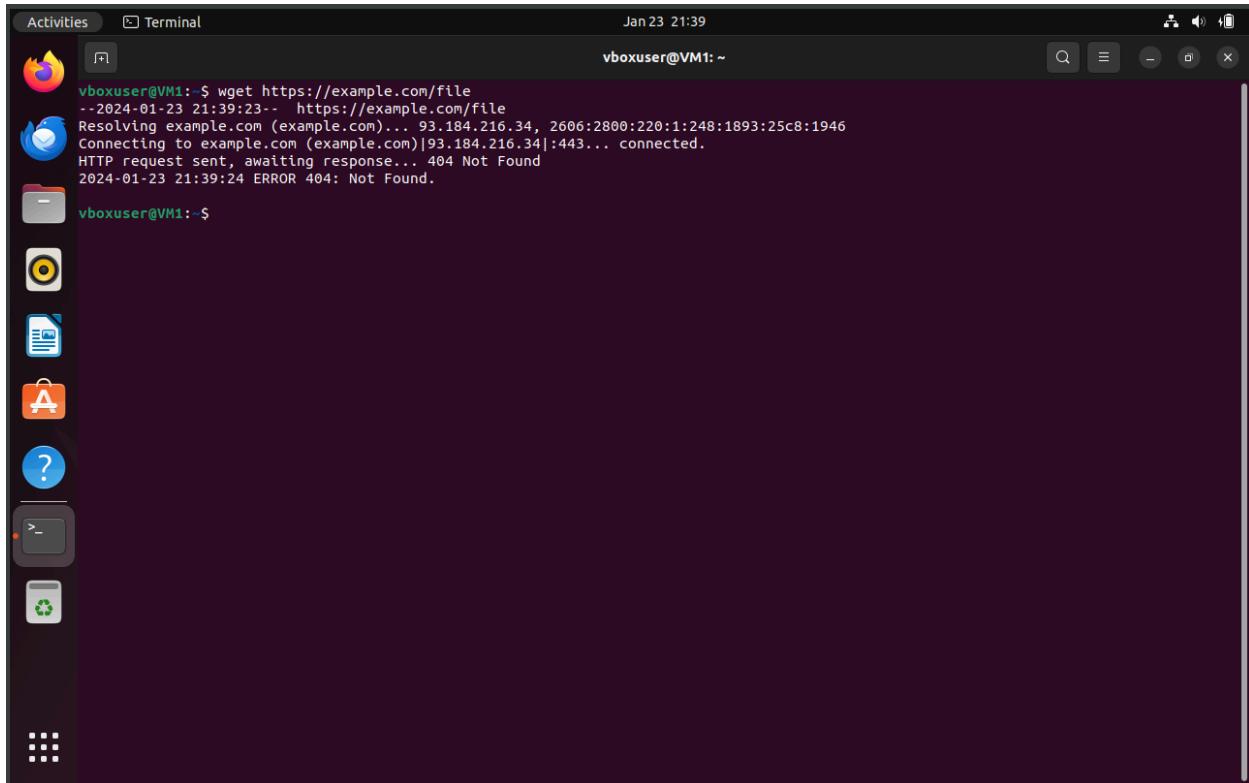


58) wget

Description: Downloads files from the internet.

Example Usage: wget https://example.com/file

CLOUD COMPUTING ASSIGNMENT 01



A screenshot of a Linux desktop environment. On the left, there is a vertical dock containing icons for various applications: a browser (Firefox), file manager, terminal, file browser, system settings, and others. The main window is a terminal window titled "Terminal" with the command "vboxuser@VM1: ~". The terminal output shows the user running "wget https://example.com/file" and receiving a 404 Not Found error. The date and time at the top of the terminal window are "Jan 23 21:39".

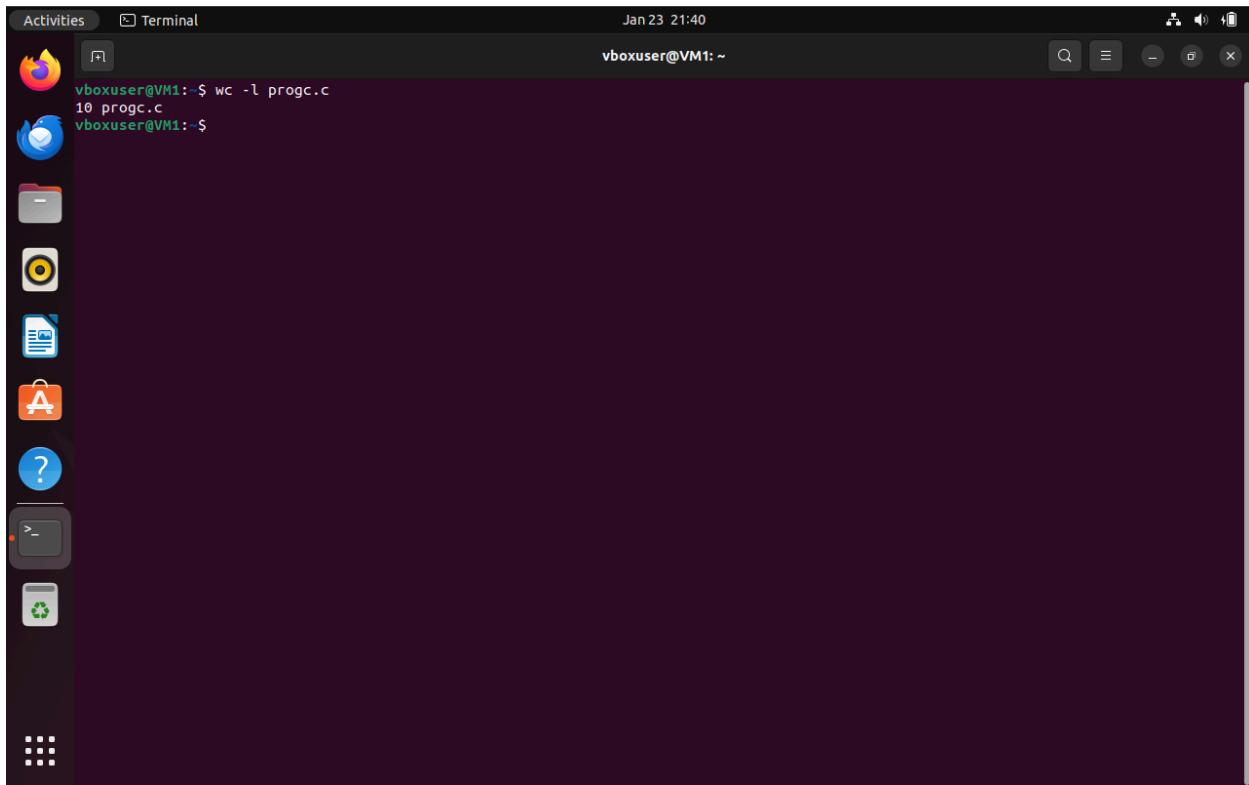
```
vboxuser@VM1:~$ wget https://example.com/file
--2024-01-23 21:39:23-- https://example.com/file
Resolving example.com (example.com)... 93.184.216.34, 2606:2800:220:1:248:1893:25c8:1946
Connecting to example.com (example.com)|93.184.216.34|:443... connected.
HTTP request sent, awaiting response... 404 Not Found
2024-01-23 21:39:24 ERROR 404: Not Found.
```

59) wc:

Description: Counts lines, words, and characters in a file.

Example Usage: wc filename.txt

CLOUD COMPUTING ASSIGNMENT 01

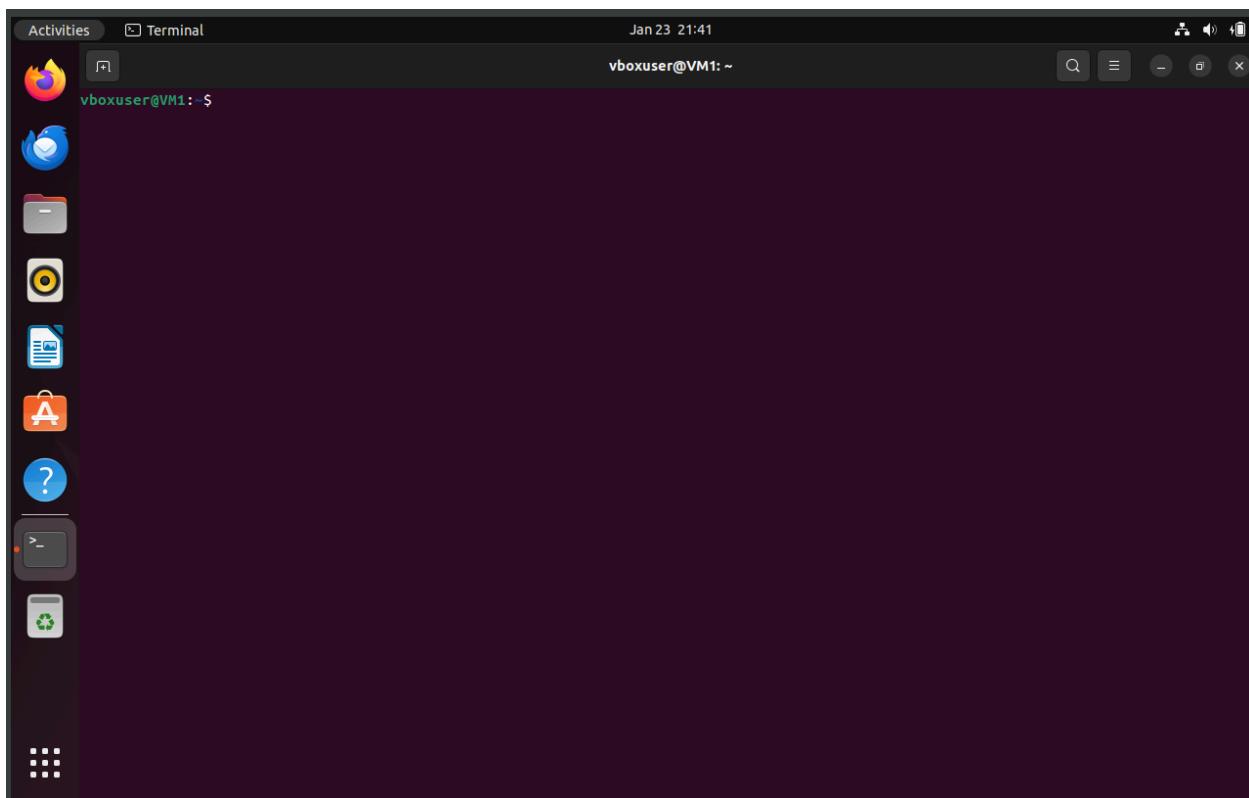
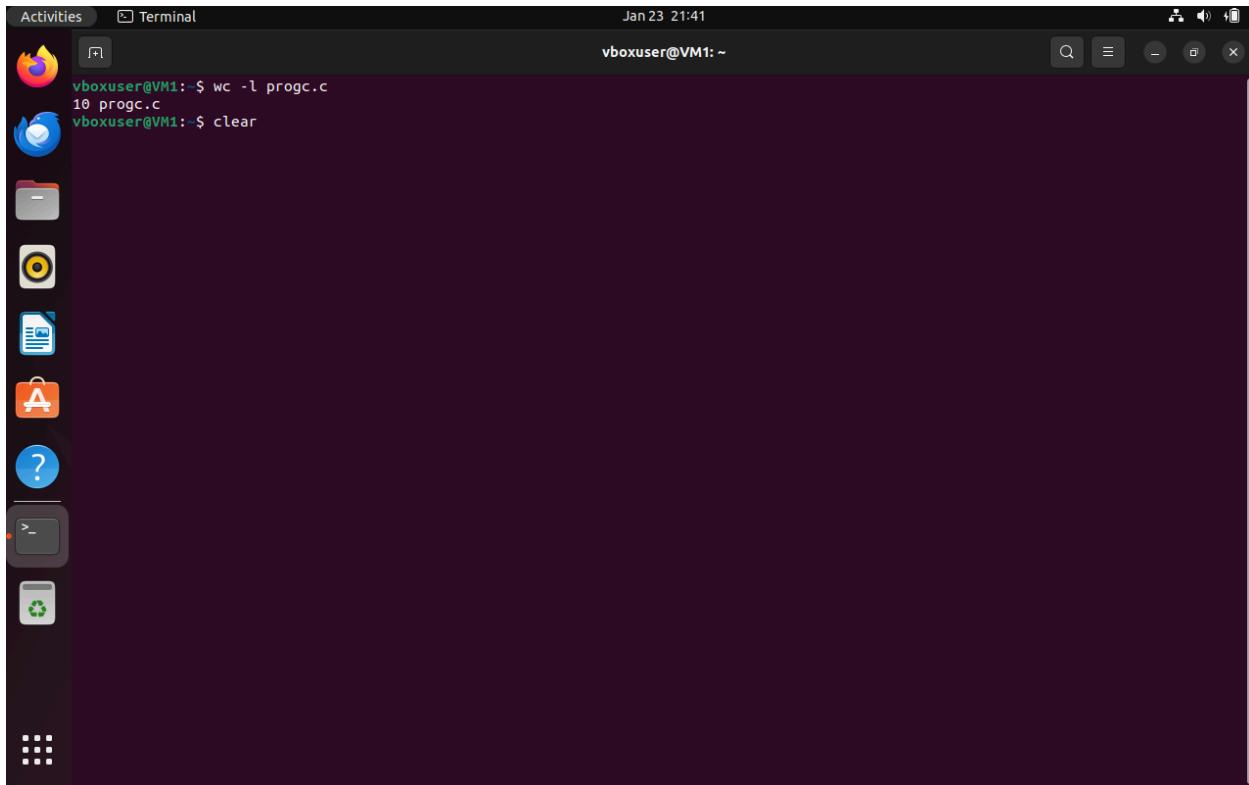


60) clear:

Description: Clears the terminal screen.

Example Usage: clear

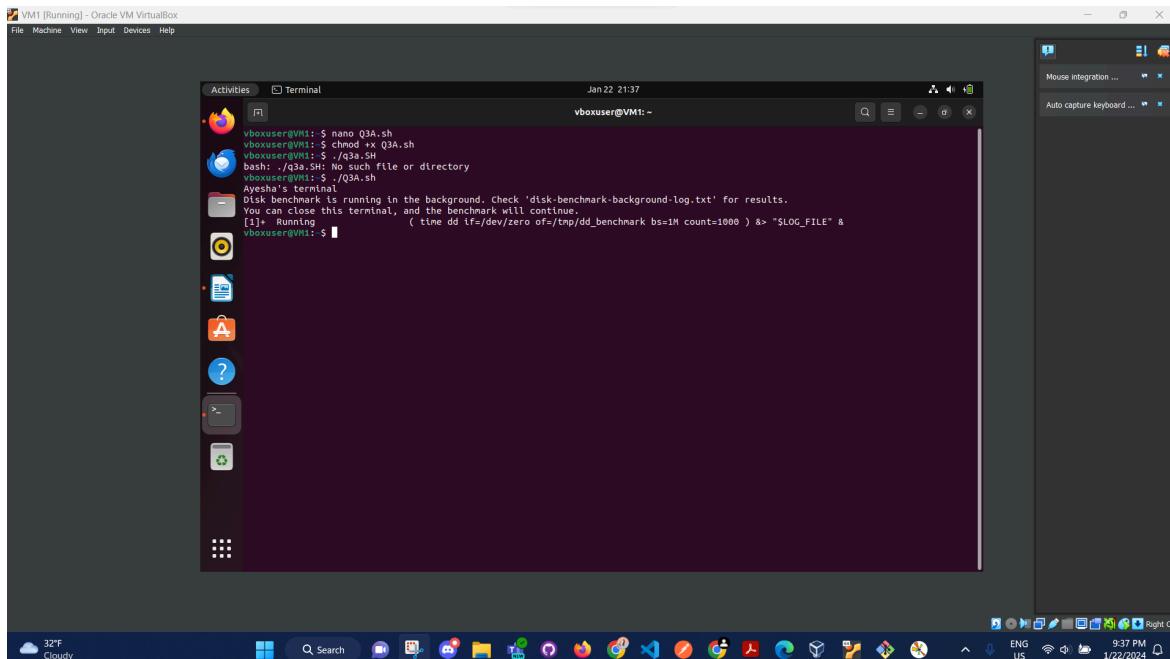
CLOUD COMPUTING ASSIGNMENT 01



CLOUD COMPUTING ASSIGNMENT 01

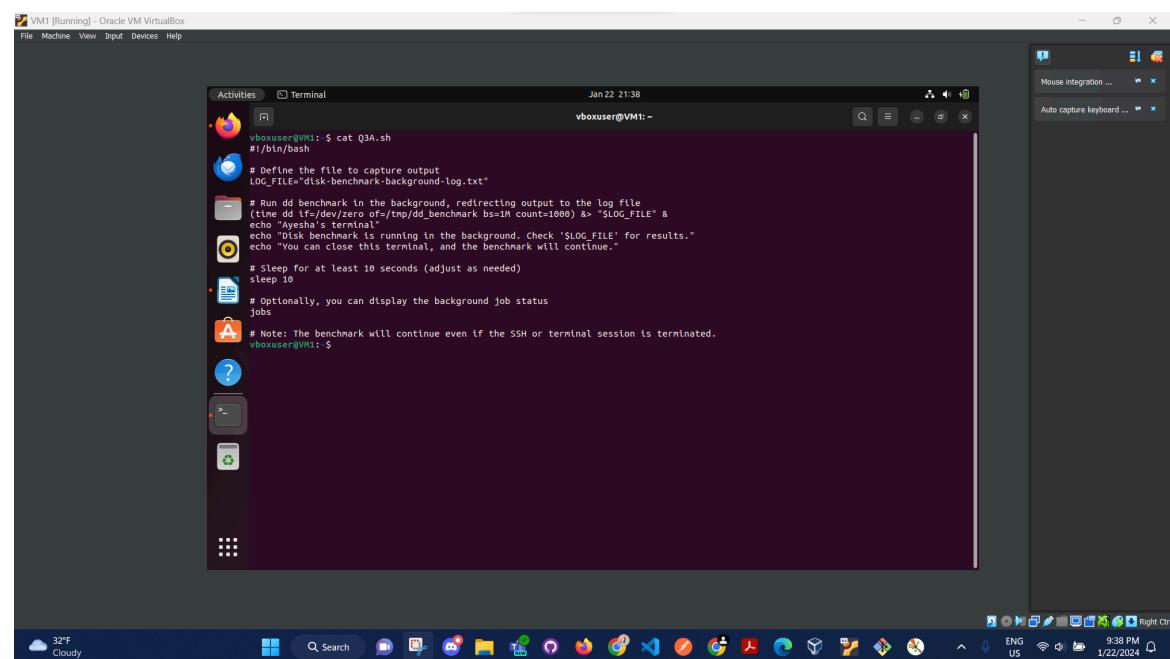
3. Write Bash scripts to do the following:

- Write a script called "disk-benchmark-background.sh" that uses the dd command to run a benchmark against the local disk in the background, that captures all the output (both standard out and error output) to a file "disk-benchmark-background-log.txt". Use the "time" command to show how long the benchmark took to complete. The benchmark should run for at least 10 seconds, and it should be completed even if the ssh (or bash) session is terminated.



A screenshot of a Linux desktop environment (Ubuntu) within a VirtualBox VM. The desktop has a dark theme with a dock at the bottom containing various application icons. A terminal window titled 'Activities Terminal' is open, showing the following command being run:

```
vboxuser@VM1:~$ nano Q3A.sh
vboxuser@VM1:~$ chmod +x Q3A.sh
vboxuser@VM1:~$ ./Q3A.sh
bash: ./Q3A.sh: No such file or directory
vboxuser@VM1:~$ ./Q3A.sh
Ayesha's terminal
[1] 1 Running
Disk benchmark is running in the background. Check 'disk-benchmark-background-log.txt' for results.
You can close this terminal, and the benchmark will continue.
[1]+ 1 Running                  ( time dd if=/dev/zero of=/tmp/dd_benchmark bs=1M count=1000 ) &> "$LOG_FILE" &
vboxuser@VM1:~$
```



A screenshot of a Linux desktop environment (Ubuntu) within a VirtualBox VM. The desktop has a dark theme with a dock at the bottom containing various application icons. A terminal window titled 'Activities Terminal' is open, showing the contents of a file named 'Q3A.sh':

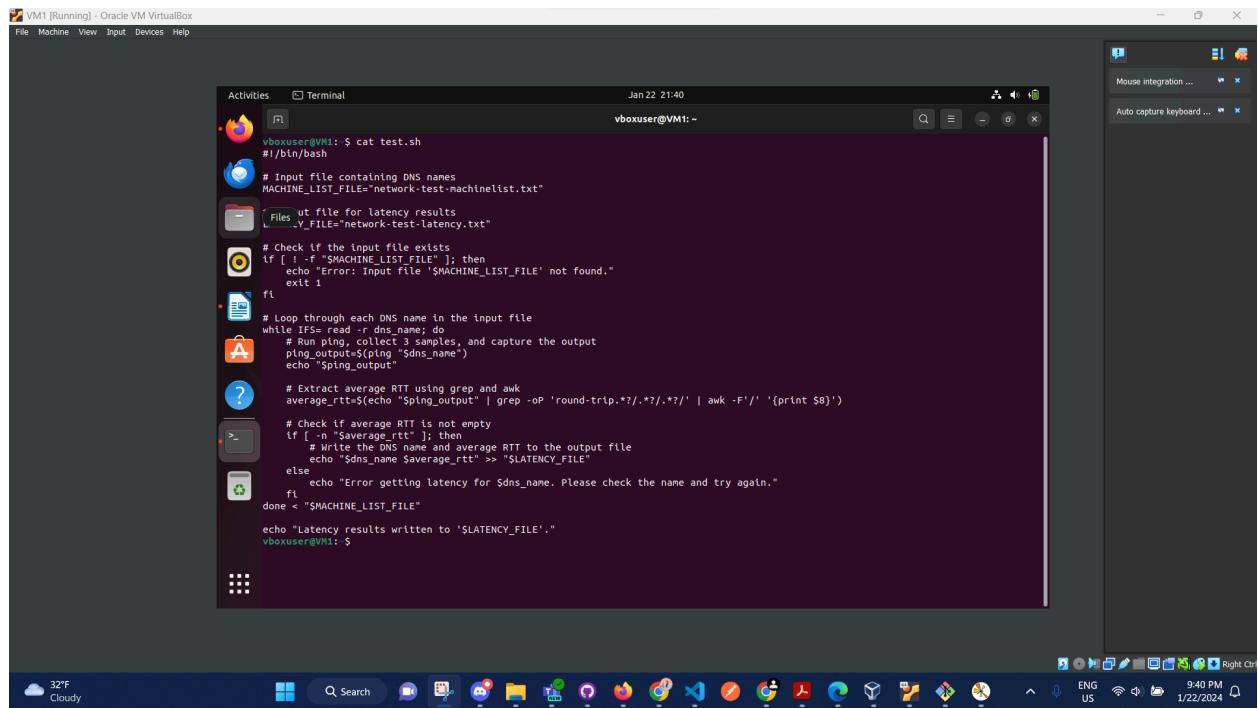
```
vboxuser@VM1:~$ cat Q3A.sh
#!/bin/bash

# Define the file to capture output
LOG_FILE="disk-benchmark-background-log.txt"

# Run dd benchmark in the background, redirecting output to the log file
(time dd if=/dev/zero of=/tmp/dd_benchmark bs=1M count=1000) >> "$LOG_FILE" &
echo "Ayesha's terminal"
echo "Disk benchmark is running in the background. Check '$LOG_FILE' for results."
echo "You can close this terminal, and the benchmark will continue."
sleep 10
# Optionally, you can display the background job status
jobs
# Note: The benchmark will continue even if the SSH or terminal session is terminated.
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

- b. Write a script called “network-test.sh” that takes input a file “network-test-machinelist.txt” with a list of DNS names (e.g. google.com, iit.edu, anl.gov), each name on a separate line, and runs the ping utility collecting 3 samples from each DNS name, and writing the RTT (round trip time) average latency into a file “network-test-latency.txt” where each line will have the DNS name and average RTT separated by a space. Make sure it works with at least 10 DNS names, but it should work for an unspecified number of DNS names.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Activities Terminal" and the date and time are "Jan 22 21:40". The command being run is "cat test.sh". The script content is as follows:

```
vboxuser@VM1:~$ cat test.sh
#!/bin/bash

# Input file containing DNS names
MACHINE_LIST_FILE="network-test-machinelist.txt"

# UT file for latency results
_Y_FILE="network-test-latency.txt"

# Check if the input file exists
if [ ! -f "$MACHINE_LIST_FILE" ]; then
    echo "Error: Input file '$MACHINE_LIST_FILE' not found."
    exit 1
fi

# Loop through each DNS name in the input file
while IFS= read -r dns_name; do
    # Run ping, collect 3 samples, and capture the output
    ping_output=$(ping "$dns_name")
    echo "$ping_output"

    # Extract average RTT using grep and awk
    average_rtt=$(echo "$ping_output" | grep -oP 'round-trip.*?/.*/\{print \$0\}' | awk -F'/' '{print $1}')

    # Check if average RTT is not empty
    if [ -n "$average_rtt" ]; then
        # Write the DNS name and average RTT to the output file
        echo "$dns_name $average_rtt" >> "$_Y_FILE"
    else
        echo "Error getting latency for $dns_name. Please check the name and try again."
    fi
done < "$MACHINE_LIST_FILE"

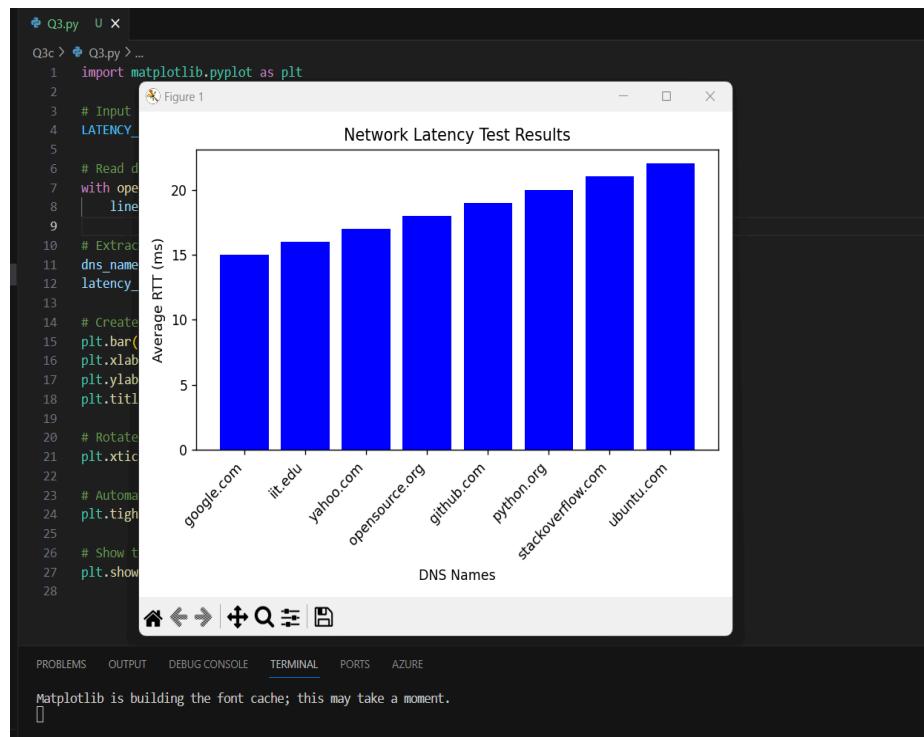
echo "Latency results written to '$_Y_FILE'."
vboxuser@VM1:~$
```

CLOUD COMPUTING ASSIGNMENT 01

- c. Write a Python matplotlib script to generate a graph of the “network-test-latency.txt” data. The graph should automatically adjust to the number of entries, and the scale of the data.

```
File Edit Selection View Go Run Terminal Help ⌘ Assignment-1
EXPLORER ... Q3c > Q3c.py > ...
ASSIGNMENT-1
Q3a
disk-benchmark-back... U
$ disk-benchmark-back... U
Q3b
network-test-latency... U
$ network-test-machin... U
$ network-testsh U
Q3c
network-test-latency... U
$ Q3c.py U
requirements.txt U
.gitattributes
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
Matplotlib is building the font cache; this may take a moment.
OUTLINE TIMELINE
main* ⌘ 0 △ 0 % 0
python Q3c bash
ENGLISH 9:32 PM 1/22/2024
Cloudy
Windows taskbar icons
3.11.7 64-bit
```

```
Q3c > Q3c.py > ...
1 import matplotlib.pyplot as plt
2
3 # Input file containing latency data
4 LATENCY_FILE = "network-test-latency.txt"
5
6 # Read data (variable) file: TextIOWrapper
7 with open(LA
8     lines = file.readlines()
9
10 # Extract DNS names and latency values
11 dns_names = [line.split()[0] for line in lines]
12 latency_values = [float(line.split()[1]) for line in lines]
13
14 # Create a bar graph
15 plt.bar(dns_names, latency_values, color='blue')
16 plt.xlabel('DNS Names')
17 plt.ylabel('Average RTT (ms)')
18 plt.title('Network Latency Test Results')
19
20 # Rotate x-axis labels for better readability
21 plt.xticks(rotation=45, ha='right')
22
23 # Automatically adjust the layout to fit the data
24 plt.tight_layout()
25
26 # Show the graph
27 plt.show()
```



CLOUD COMPUTING ASSIGNMENT 01

4. Answer the following questions about VMs:

- a. In the system configuration of the VM, explain how changing the number of processors changes the behavior of your VM. Explain a scenario where you want to set this to the minimum, and a scenario where you want to set it to the maximum. Why is setting it to the maximum potentially a bad idea?

Adjusting the number of processors in a virtual machine (VM) has a profound impact on performance, primarily through the allocation of virtual CPUs. Let's delve into scenarios at both ends of the spectrum – minimum and maximum settings – and explore the nuanced effects on VM behavior.

Impact of Processor Configuration on VM Behavior:

1. Enhanced Performance with Increased Processors:

- *Analogy:* Increasing processors in a VM is akin to expanding a workforce, offering heightened parallel processing capacity.
- *Example:* Consider a VM hosting a web server experiencing a surge in incoming requests. Augmenting processors facilitates efficient handling of concurrent tasks, leading to improved response times.

2. Resource Contention and Overhead with Fewer Processors:

- *Analogy:* Reducing processors can lead to resource contention, similar to downsizing a team, resulting in prolonged task execution due to inadequate computational power.
- *Example:* In scenarios with a VM executing complex computational tasks, lowering the processor count may lead to slower execution times due to insufficient resources.

3. Balancing Act for Optimal Resource Utilization:

- *Analogy:* Achieving optimal VM behavior involves a delicate balance of processors for efficient resource utilization without causing contention or underutilization.
- *Example:* In a VM serving as a virtual desktop for multiple users, an adequate number of processors ensures smooth multitasking without one user's activities hindering others.

4. Considerations for Host Machine Resources:

- *Analogy:* Crucially, host machine resources must be considered. Allocating excess processors without sufficient physical cores may lead to resource exhaustion and degraded performance.
- *Example:* If a host machine has limited CPU cores, assigning excess processors to multiple VMs can result in contention, adversely affecting overall system performance.

This exploration underscores the importance of carefully configuring processors to optimize VM behavior, considering the diverse demands of different scenarios and the underlying host machine resources.

CLOUD COMPUTING ASSIGNMENT 01

Scenario for Minimizing Processor Allocation:

- In situations where multiple VMs share the same host and have low individual processing requirements, configuring processors to the minimum ensures equitable distribution of resources.
 - Illustration 1: Within a development environment running lightweight VMs concurrently, minimal processors promote efficient resource utilization.
 - Illustration 2: When engaged in batch processing for data analysis tasks with a more sequential nature, setting processors to the minimum is adequate for reasonable execution.

Scenario for Maximizing Processor Allocation:

- When dealing with a single VM hosting resource-intensive applications, maximizing processors contributes to enhanced overall performance.
 - Illustration 1: A high-traffic e-commerce website experiences peak times and benefits from maximum processors to efficiently manage increased workloads.
 - Illustration 2: A database server handling substantial concurrent transactions thrives with maximum processors, improving responsiveness and overall throughput.

Caution Regarding Maximum Processor Allocation:

- While maximizing processors can elevate performance, it is imperative to consider the overall resources of the host machine to prevent potential resource contention.
 - Cautionary Example: Allocating an excess number of processors without sufficient physical cores may result in resource exhaustion, adversely impacting the performance of all VMs.

In essence, adjusting VM processor configurations demands a thoughtful assessment of specific workloads and host resources, ensuring optimal performance without introducing contention or risking resource depletion.

- b. **In the system configuration of the VM, under the Acceleration Tab, explain the difference between the paravirtualization options: None, Legacy, Minimal, Hyper-V, and KVM. Explain which one would be best to use with Ubuntu Linux, and why.**

The elucidation below delves into the paravirtualization options available under the Acceleration Tab within the system configuration of a virtual machine (VM).

Overview of the Acceleration Tab: In the configuration settings of a VM, the Acceleration Tab provides users with a spectrum of paravirtualization options aimed at enhancing performance. These options bring forth specialized interfaces and optimizations, proving especially advantageous for tasks that demand substantial computing resources.

CLOUD COMPUTING ASSIGNMENT 01

Paravirtualization Options:

1. None:

- *Definition:* The "None" option means there are no paravirtualization features, relying solely on traditional virtualization methods.
- *Examples/Usage:*
 1. *Compatibility Priority:* Choose "None" for broad compatibility with various guest operating systems, especially those lacking support for paravirtualized drivers.
 2. *Legacy System Emulation:* Opt for "None" when emulating legacy systems that are incompatible with paravirtualization enhancements.
 3. *Generic Testing Environments:* Ideal for testing environments that support various operating systems without specific optimizations.

2. Legacy:

- *Definition:* The "Legacy" option provides basic paravirtualization support, often emulating legacy interfaces for compatibility.
- *Examples/Usage:*
 1. *Supporting Older Operating Systems:* Opt for "Legacy" when running VMs with older guest operating systems lacking compatibility with modern paravirtualized drivers.
 2. *Legacy Software Testing:* Suitable for environments focused on testing legacy software or systems.
 3. *Hardware Compatibility:* Appropriate when dealing with hardware dependent on legacy interfaces for proper functioning.

3. Minimal:

- *Definition:* "Minimal" paravirtualization offers essential features without unnecessary complexity, maintaining a balanced approach.
- *Examples/Usage:*
 1. *Lightweight Workloads:* Choose "Minimal" for VMs with lightweight workloads, benefiting from basic paravirtualization without added intricacy.
 2. *General Testing Scenarios:* Suitable for creating VMs in general testing scenarios, achieving a balance between enhanced performance and simplicity.
 3. *Educational Environments:* Effective in educational settings that prefer a simplified paravirtualization setup.

4. Hyper-V:

- *Definition:* The "Hyper-V" option customizes paravirtualization support for Microsoft's virtualization platform.
- *Examples/Usage:*
 1. *Running on Hyper-V Hosts:* Opt for "Hyper-V" when the VM operates on a Hyper-V host, ensuring optimized performance and compatibility.
 2. *Integration with Hyper-V Features:* Ideal for utilizing features unique to the Hyper-V environment, such as enhanced integration services.
 3. *Cross-Platform Compatibility:* Useful when maintaining compatibility with both Hyper-V and non-Hyper-V environments is essential.

CLOUD COMPUTING ASSIGNMENT 01

- **Definition:** The "KVM" (Kernel-based Virtual Machine) paravirtualization option is crafted for peak performance on Linux hosts leveraging the KVM hypervisor.
- **Examples/Usage:**
 1. *Linux Server Virtualization:* Opt for "KVM" when your VM operates on a Linux server utilizing KVM, ensuring optimal performance through native virtualization capabilities.
 2. *Ubuntu Linux Hosts:* Choose "KVM" specifically for Ubuntu Linux hosts, aligning with the inherent virtualization capabilities of the Linux environment.
 3. *Efficient Resource Utilization:* Ideal when emphasizing efficient resource utilization on virtualization hosts based on the Linux platform.

Best Option for Ubuntu Linux:

Given the strong support for KVM in Ubuntu Linux, selecting the "KVM" paravirtualization option proves to be the most fitting. This decision guarantees peak performance and harmonizes with the inherent virtualization capabilities of Linux, delivering effective and smooth virtualization on Ubuntu hosts.

- c. **In storage devices when configuring the VM, there are multiple types of storage controllers: explain the difference between the IDE, SATA, and NVMe controller. Give an example for each type of storage controller of a scenario where you may want to use this type of controller.**

When establishing a virtual machine (VM) and configuring storage settings, the term "storage controllers" refers to virtualized components responsible for managing communication between the VM and its storage devices. These controllers replicate the functionalities of physical storage controllers within the virtual environment. Various virtualization platforms provide diverse types of storage controllers, including IDE, SATA, SCSI, and NVMe controllers. The selection of a storage controller depends on factors such as the VM's operating system and the required storage devices, providing flexibility in VM configuration.

Here is a breakdown outlining the distinctions between storage controllers(IDE, SATA, and NVMe),in the context of configuring virtual machines (VMs).

Controller Type:

1. IDE (Integrated Drive Electronics):

- **Description:** IDE serves as a traditional storage controller primarily used in older systems.
- **Key Differences:**
 - a) The transfer rate is comparatively slower.
 - b) Only two devices are allowed for one channel.
 - c) It is compatible with older hard drives and optical drives.

CLOUD COMPUTING ASSIGNMENT 01

- **Real-Life Examples:**

Scenario 1: Legacy System Support

Explanation: In older systems without compatibility for SATA or NVMe, opting for an IDE controller enables the connection of traditional hard drives or optical drives.

Scenario 2: Legacy Software Testing

Explanation: When conducting tests on legacy software dependent on IDE controllers, configuring a VM with an IDE controller accurately replicates the hardware environment.

Scenario 3: Basic Data Storage

Explanation: In situations where high-speed data transfer is not a critical requirement, such as basic file storage needs, IDE controllers are suitable for cost-effective solutions.

2. NVMe (Non-Volatile Memory Express):

- **Description:** In the realm of SSDs, NVMe stands as a high-performance storage controller.
- **Key Differences:**
 - a) NVMe exhibits the fastest data transfer rates among its counterparts.
 - b) It is optimal for use with SSDs and offers low latency.
 - c) NVMe is a preferred choice for high-performance workloads due to its support for multiple queues and parallelism.

- **Real-Life Examples:**

Scenario 1: Mainstream Desktop Virtualization

Explanation: SATA controllers find widespread usage in VMs emulating mainstream desktop environments, utilizing regular hard drives or SSDs for storage.

Scenario 2: Multimedia Content Editing

Explanation: For VMs engaged in multimedia content editing, where accelerated data transfer rates are advantageous, SATA controllers offer an optimal balance of performance and cost.

Scenario 3: Distributed Database Storage

Explanation: In situations demanding distributed database storage, SATA controllers facilitate the support of multiple drives, ensuring data redundancy and reliability.

3. SATA (Serial ATA):

- **Description:** SATA stands out as a widely used controller that integrates well with modern systems.
- **Key Differences:**
 - a) It boasts faster transfer rates compared to IDE.
 - b) SATA supports the hot-swapping of drives.
 - c) It is commonly used with both SSDs and hard drives.

CLOUD COMPUTING ASSIGNMENT 01

- **Real-Life Examples:**

Scenario 1: High-Performance Virtualization

Explanation: NVMe controllers stand out in virtual environments that demand high-performance storage, such as those running intricate simulations or resource-intensive applications.

Scenario 2: Real-Time Data Processing

Explanation: For VMs engaged in real-time data processing tasks, like financial transactions or scientific simulations, NVMe controllers guarantee minimal latency and swift data access.

Scenario 3: Big Data Analytics

Explanation: VMs dedicated to big data analytics find advantage in NVMe controllers due to their exceptional data transfer rates, contributing to the efficiency of data-intensive operations.

Hence, the selection of a storage controller—IDE, SATA, or NVMe—hinges on the distinct performance requirements and use cases of the virtualized environment. Each controller type possesses unique strengths, catering to diverse storage needs within VM configurations.

- d. In the network configuration of the VM, there are multiple types of network adapters: explain the difference between NAT, Bridged Adapter, Internal Network, and Host-only Network. Give an example for each type of network of a scenario where you may want to use this type of network.

Network Adapters:

Network adapters within virtual machines facilitate communication between the virtual machines and the external network, playing a pivotal role in determining how VMs connect and engage with the broader network environment.

Types of Network Adapters:

1. NAT (Network Address Translation):

- *Description:* NAT allows a VM to utilize the host's IP address, providing internet access while concealing the VM's individual IP address.
- *Example Scenario:* When a VM requires internet access without directly exposing its own IP address.

2. Bridged Adapter:

- *Description:* The Bridged Adapter connects a VM directly to the physical network, presenting it as a distinct device on the same network as the host.
- *Example Scenario:* In situations where a VM needs to function as an independent device with its own IP address on the network.

3. Internal Network:

CLOUD COMPUTING ASSIGNMENT 01

- *Description:* The Internal Network facilitates communication between VMs on the same host but isolates them from external networks.
 - *Example Scenario:* When establishing an isolated network for VMs needing interaction with each other but not with the external network.
- 4. Host-only Network:**
- *Description:* The Host-only Network allows communication between VMs and the host while keeping them isolated from external networks.
 - *Example Scenario:* In cases where VMs need exclusive communication with each other and the host without external network access.

Real-Life Examples for Each Network Adapter:

1. NAT:

- *Scenario 1: Software Development Environment*
 - *Explanation:* When developing software requiring internet access for updates or downloads, NAT ensures seamless connectivity without exposing individual VM IP addresses.
- *Scenario 2: Web Browsing Sandbox*
 - *Explanation:* Establishing a sandbox environment for web browsing, where VMs share the host's IP address, protects individual VMs from potential internet threats.
- *Scenario 3: System Updates*
 - *Explanation:* In situations where VMs need to fetch updates from external servers, NAT allows them to share a common external IP address for update retrieval.

2. Bridged Adapter:

- *Scenario 1: Web Server Testing*
 - *Explanation:* When testing a web server within a VM that needs to be accessible from external devices, using a Bridged Adapter ensures the VM has its own IP address on the host's network.
- *Scenario 2: Remote Device Testing*
 - *Explanation:* In scenarios where VMs need to communicate with physical devices on the host's network, using a Bridged Adapter provides direct access to the external network.
- *Scenario 3: Network Monitoring VM*
 - *Explanation:* Creating a VM for network monitoring that needs to capture external network traffic is facilitated by a Bridged Adapter, allowing direct interaction with the external network.

3. Internal Network:

- *Scenario 1: Database Server and Client VMs*
 - *Explanation:* When setting up VMs for a database server and client applications that need to communicate with each other, using an Internal Network ensures internal communication without external access.
- *Scenario 2: Development and Testing Environment*
 - *Explanation:* In a development environment where VMs simulate different components of a system, using an Internal Network keeps the communication isolated within the host for testing purposes.
- *Scenario 3: Training Labs*

CLOUD COMPUTING ASSIGNMENT 01

- *Explanation:* Setting up training labs with VMs that need to interact with each other but not with external networks can be achieved using an Internal Network for a controlled learning environment.

4. Host-only Network:

- *Scenario 1: Secure File Sharing*
 - *Explanation:* When VMs need to share files securely with the host but not with external networks, using a Host-only Network ensures isolated and secure communication.
- *Scenario 2: Development and Debugging Environment*
 - *Explanation:* Creating a VM for development and debugging purposes where the VM needs exclusive communication with the host is facilitated by a Host-only Network.
- *Scenario 3: Collaborative Project Environment*
 - *Explanation:* In collaborative projects where VMs need to interact with each other and the host but remain isolated from external networks, a Host-only Network provides a secure collaboration environment.

The selection of a network adapter type—NAT, Bridged Adapter, Internal Network, or Host-only Network—depends on specific use cases and requirements, offering flexibility in configuring VMs for various networking scenarios. Each network adapter type serves distinct purposes, from internet access to isolated internal communication, ensuring efficient and secure network configurations within virtualized environments.

e. For the USB configuration of the VM, explain the difference between USB 1.1, 2.0, and 3.0 controllers.

Details regarding USB controllers—USB 1.1, USB 2.0, and USB 3.0—are outlined as follows:

USB 1.1:

- *Description:* USB 1.1, also referred to as USB 1.0, marked the initial widespread adoption of the USB standard in 1996.
- *Key Characteristics:*
 - *Maximum Data Transfer Rate:* 12 Mbps.
 - *Connector:* Standard-A and Standard-B connectors.
 - *Introduced features like Plug and Play and hot-swapping.*
 - *Utilized for basic peripherals such as keyboards, mice, and printers.*

USB 2.0:

- *Description:* Released in 2000, USB 2.0 represented an improvement over USB 1.1, providing higher data transfer rates.
- *Key Characteristics:*
 - *Maximum Data Transfer Rate:* 480 Mbps.
 - *Backward compatible with USB 1.1.*
 - *Enhanced power management.*

CLOUD COMPUTING ASSIGNMENT 01

- *Supported a broader range of devices, including external hard drives and digital cameras.*

USB 3.0:

- **Description:** Introduced in 2008, USB 3.0 brought significant advancements in speed and efficiency.
- **Key Characteristics:**
 - **Maximum Data Transfer Rate:** 5 Gbps.
 - **Introduced a new SuperSpeed transfer mode.**
 - **Larger bandwidth for faster data transfer.**
 - **Improved power efficiency.**
 - **Backward compatible with USB 2.0.**
- **USB 1.1 (USB 1.0):**
 - **Description:** USB 1.1, also known as USB 1.0, was the initial widely adopted USB standard introduced in 1996.
 - **Data Transfer Rate:** 12 Mbps.
 - **Connector Types:** Standard-A, Standard-B.
 - **Key Features:** Introduced Plug and Play, hot-swapping; used for basic peripherals like keyboards, mice, and printers.
- **USB 2.0:**
 - **Description:** USB 2.0, released in 2000, improved upon USB 1.1, offering a higher data transfer rate of 480 Mbps.
 - **Backward Compatibility:** Yes, with USB 1.1.
 - **Enhancements:** Introduced enhanced power management.
 - **Extended Device Support:** Included external hard drives and digital cameras.
- **USB 3.0:**
 - **Description:** USB 3.0, introduced in 2008, brought significant advancements in speed and efficiency.
 - **Data Transfer Rate:** 5 Gbps.
 - **Transfer Mode:** Introduced SuperSpeed transfer mode.
 - **Connector Types:** Standard-A, Standard-B, Micro-B.
 - **Power Efficiency:** Improved power management.
 - **Backward Compatibility:** Yes, with USB 2.0.

USB technology has evolved over these standards, providing users with increasingly faster and more efficient data transfer options across various devices.