|  |  |
| --- | --- |
|  | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

Operating Systems-(CMPS 405)

**Task: Project Phase 1**

*Fall 2024*

**Project Group Members:**

Marwan Hashish ID:201701546

Abdulla Jamali ID: 202104080

Nasser Aljufairi ID: 202005009

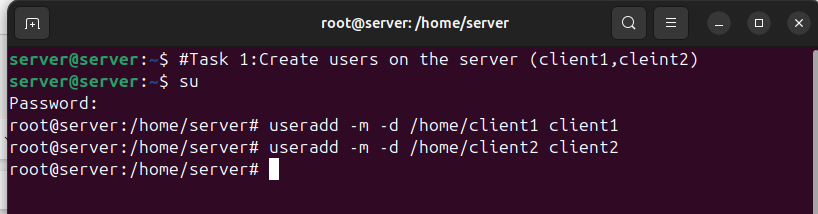
Murshed Al-Muhannadi ID: 201706102

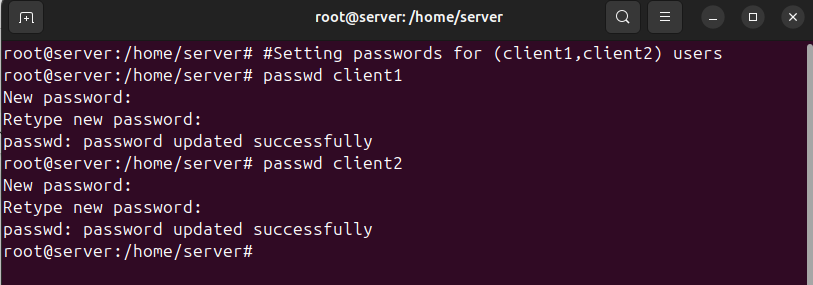
**Instructor**: ***Eng.Heba Dawoud***

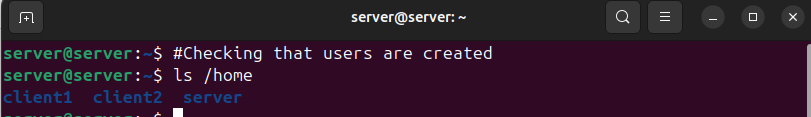
|  |  |  |
| --- | --- | --- |
| **student name** | **Student part in the Project** | **participation percentage** |
| Marwan Hashish | 1-Server-side tasks (VM1):  Configuration tasks SSH and SFTP  Shell scripts: network.sh, traceroute.sh  (70% of the server-side tasks)  2- Helped with (search.sh, system.sh) | 25% |
| Abdulla Jamali | 1. Login.sh 2. Check.sh (client1) 3. Helped with search.sh | 25% |
| Nasser Aljufairi | Shell scripts: 1-system.sh,  2-search.sh | 25% |
| Murshed Al-Muhannadi | 1. Login.sh 2. Clientinfo.sh 3. Helped with serach.sh | 25% |

**Task 1: Setup Server side (VM1)**

**-Create users on the server (client 1, client2)**

-Setting passwords for client1 & client2

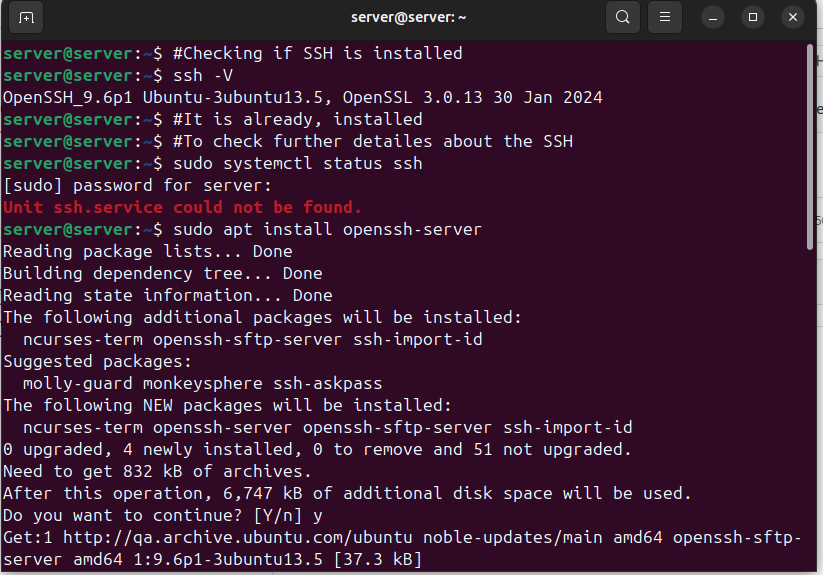
-Checking that users are created (client1, client2):

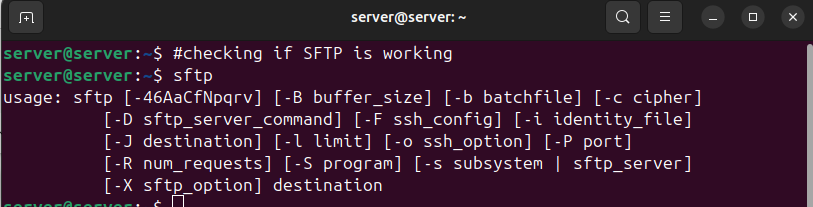


**-Install and enable SHHD**

-Checking if SSH is installed  
-The command to install SSH if it’s not installed: $ sudo apt install openssh-server

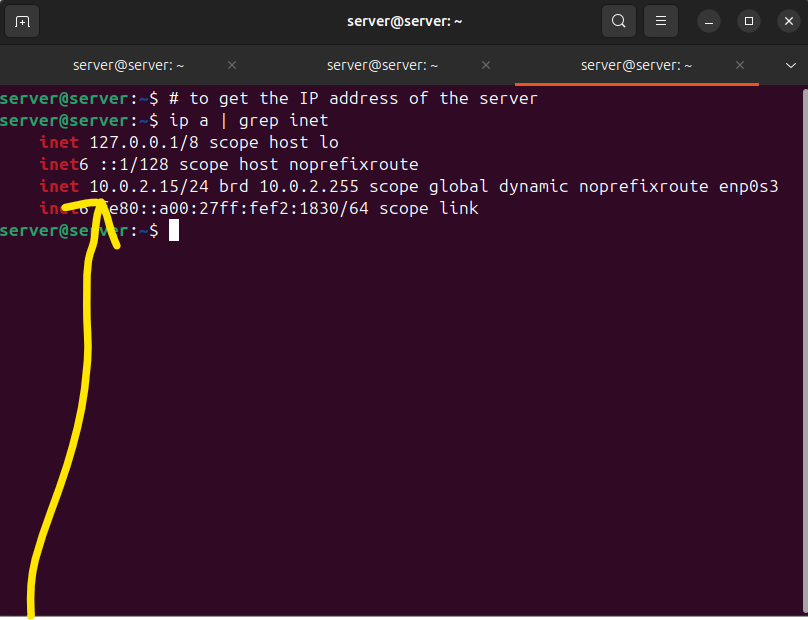
But it was already installed so no need for this step.

-Check if SFTP is working

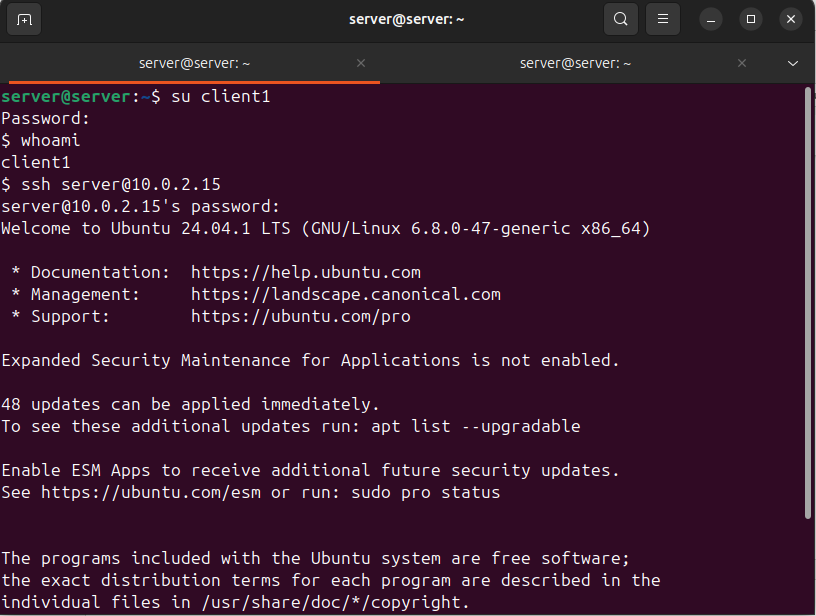


**Task 2 : Configuration**

To Get the IP of the server we did:



Accessing the server from client 1 using SSH and the ip address of the server we just retrieved:

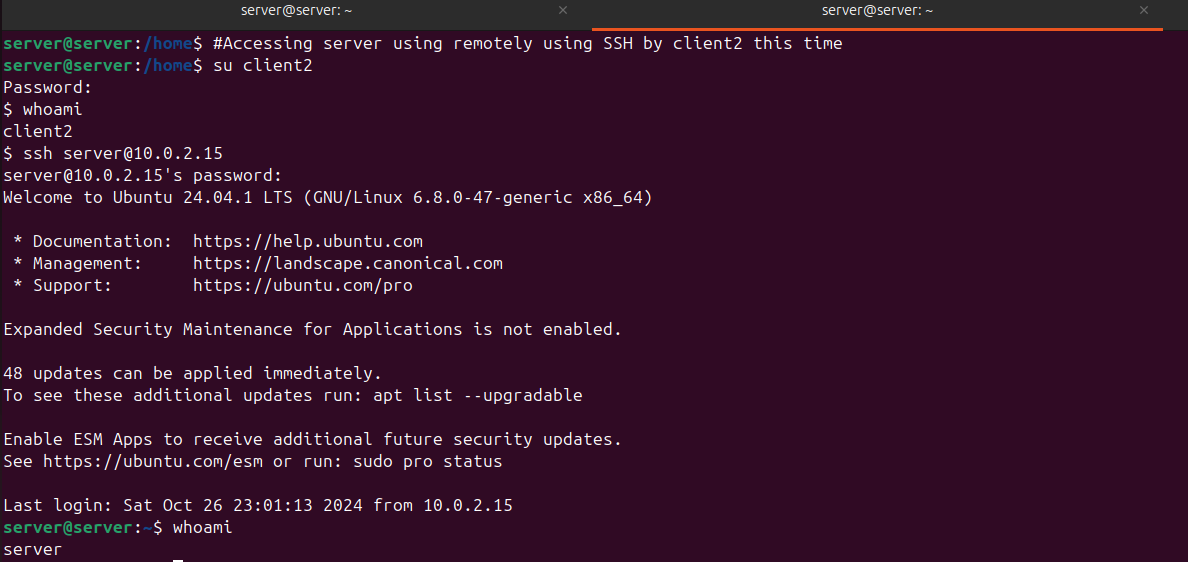


We managed to access the server using SSH (remotely) by client1 user:

A black screen with white text

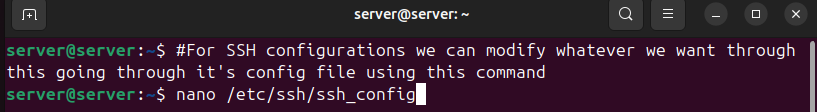
Description automatically generated

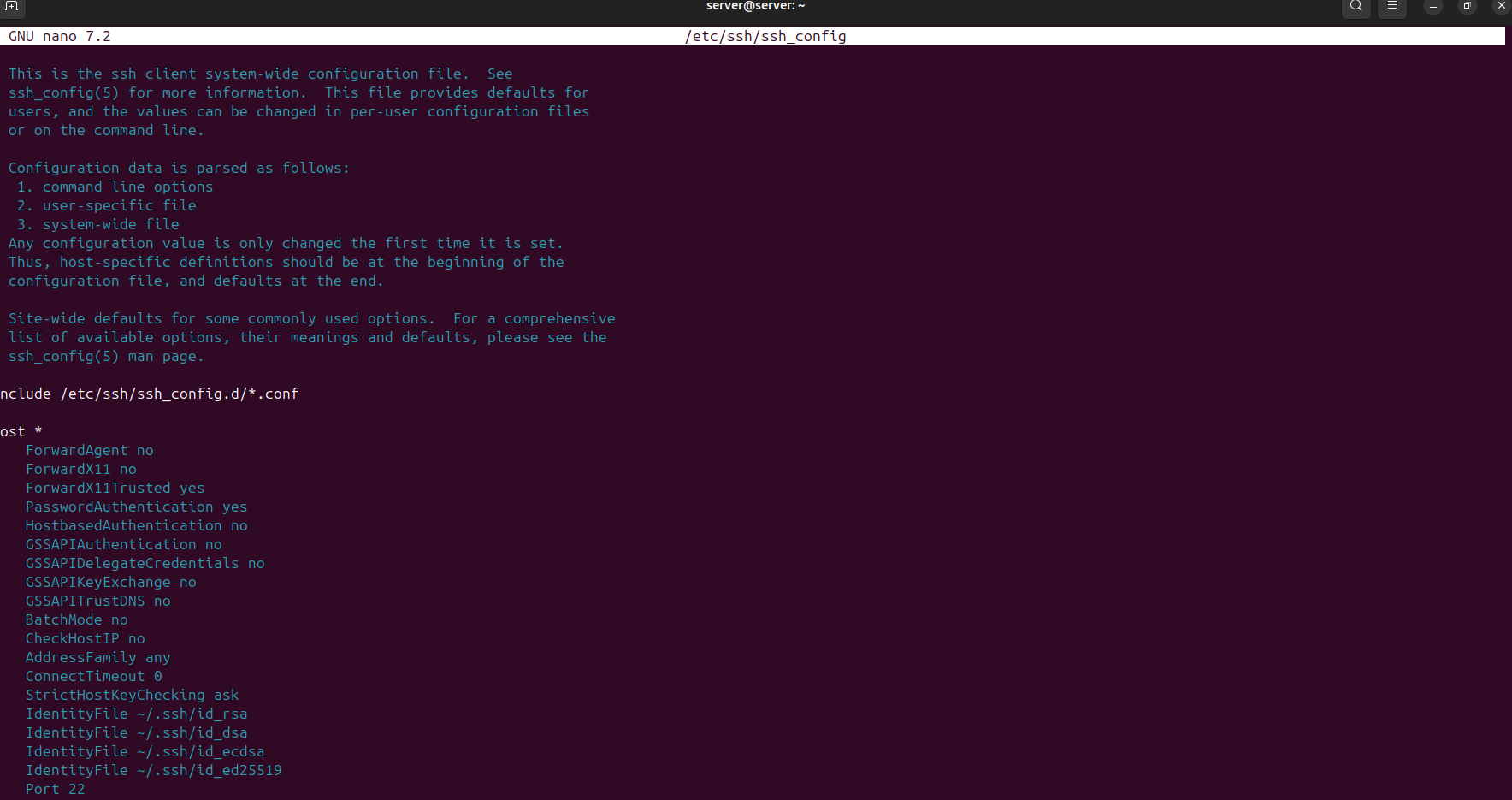
-We deduce that SSH is enabled successfully for client 1, we now will do the same for client2:



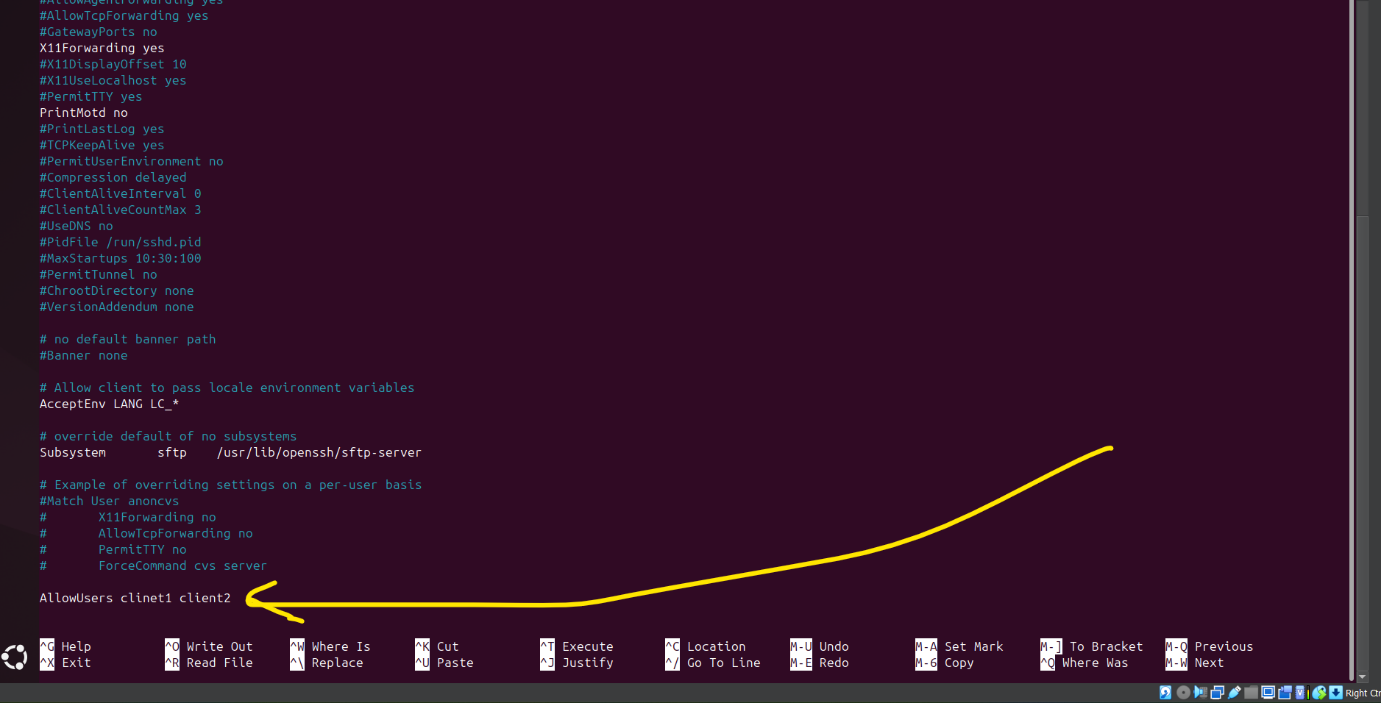
-We deduce that SSH is enabled successfully for client 2

-Mostly the configuration of SSH is done however:



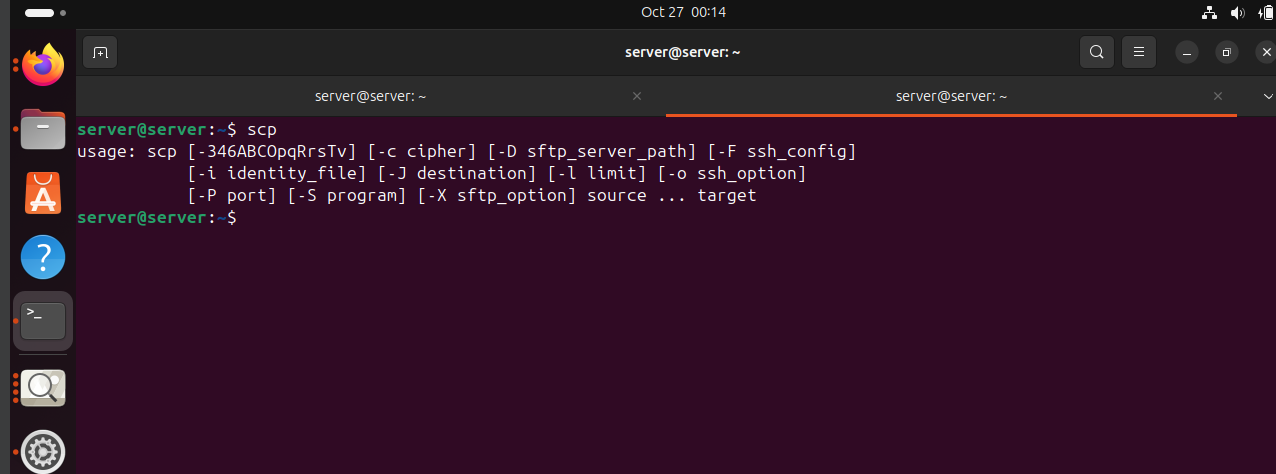


-We can enable or disable based on our needs using: sudo nano /etc/ssh/ssh\_config.



-We add this option for it to only allow the users: client1, client2

-Checking the SCP is working



**Task 3: Shell Scripting**

**Network.sh script:**

# Verify and install missing network tools

if ! command -v ping &> /dev/null || ! command -v traceroute &> /dev/null; then

echo "Required network tools not found, installing..."

sudo apt-get update && sudo apt-get install -y iputils-ping traceroute

echo "Network tools successfully installed."

fi

# Function to execute traceroute if ping fails

execute\_traceroute() {

local target=$1

echo "Initiating traceroute to $target..."

./traceroute.sh "$target"

}

# Initialize counters for summary

checks\_total=0

successful\_checks=0

failed\_checks=0

# Define retry limits

retry\_limit=3

# Iterate over each target IP passed as arguments

for target in "$@"; do

echo "Checking connectivity for $target (Maximum attempts: $retry\_limit)"

checks\_total=$((checks\_total + 1))

attempt\_count=1

connection\_success=false

# Retry pinging up to retry\_limit times

while (( attempt\_count <= retry\_limit )); do

echo "Attempt $attempt\_count: Pinging $target..."

# Ping with 5-second timeout

if ping -c 1 -W 5 "$target" &> /dev/null; then

echo "$(date '+%Y-%m-%d %H:%M:%S') - Successful connection to $target." | tee -a network.log

successful\_checks=$((successful\_checks + 1))

connection\_success=true

break # Stop retries on success

else

echo "Attempt $attempt\_count failed to reach $target."

fi

attempt\_count=$((attempt\_count + 1))

done

# Execute traceroute if all attempts fail

if [ "$connection\_success" = false ]; then

echo "All attempts failed for $target. Executing traceroute..."

execute\_traceroute "$target"

failed\_checks=$((failed\_checks + 1))

fi

done

# Generate final summary report

echo "📊 Connectivity Test Summary 📊"

echo "Total Targets Tested: $checks\_total"

echo "Successful Connections: $successful\_checks"

echo "Failed Connections: $failed\_check

**Traceroute.sh script**

#!/bin/bash

# Function to log messages to both console and log file

log\_message() {

echo "$1" | tee -a network.log

}

target\_ip=$1

log\_message "Starting traceroute to $target\_ip..."

{

echo "Displaying Routing Table:"

route -n

echo "System Hostname: $(hostname)"

echo "Performing DNS Resolution Test:"

nslookup google.com

echo "Tracing Route to Google (google.com):"

traceroute google.com

echo "Pinging Google to verify connectivity:"

ping -c 3 google.com

} | tee -a network.log

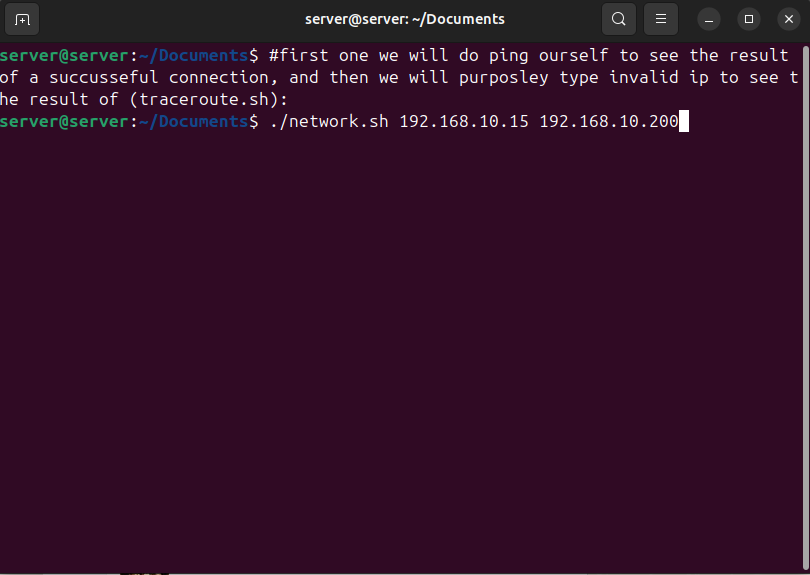
log\_message "Traceroute for $target\_ip completed. (Reboot disabled during testing phase)"

# Uncomment the following line to allow reboot after testing completion

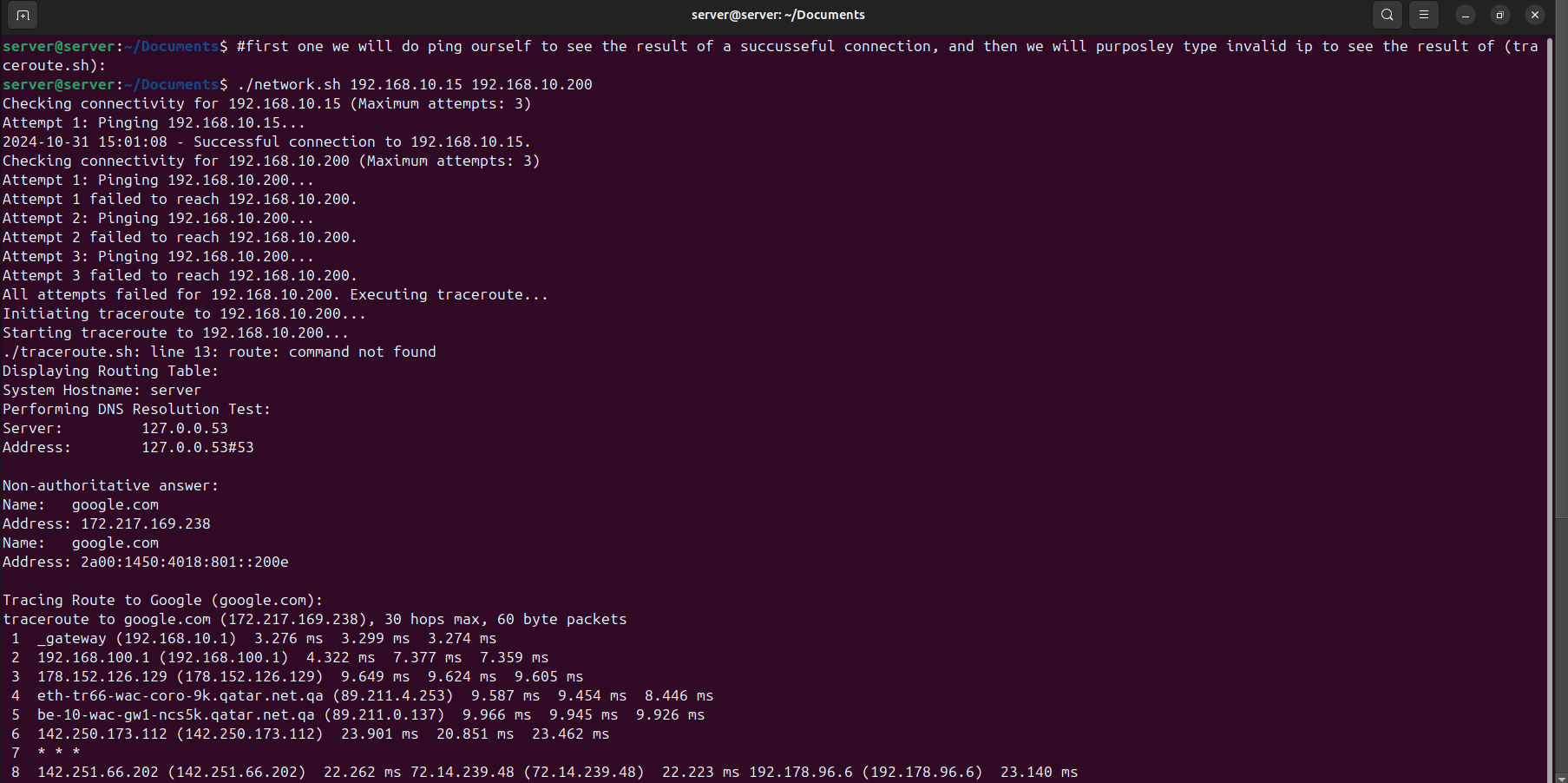
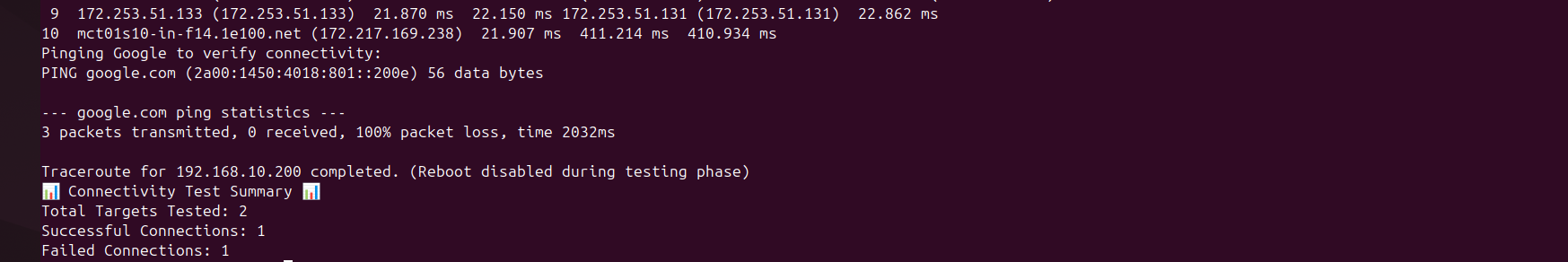
# sudo reboot

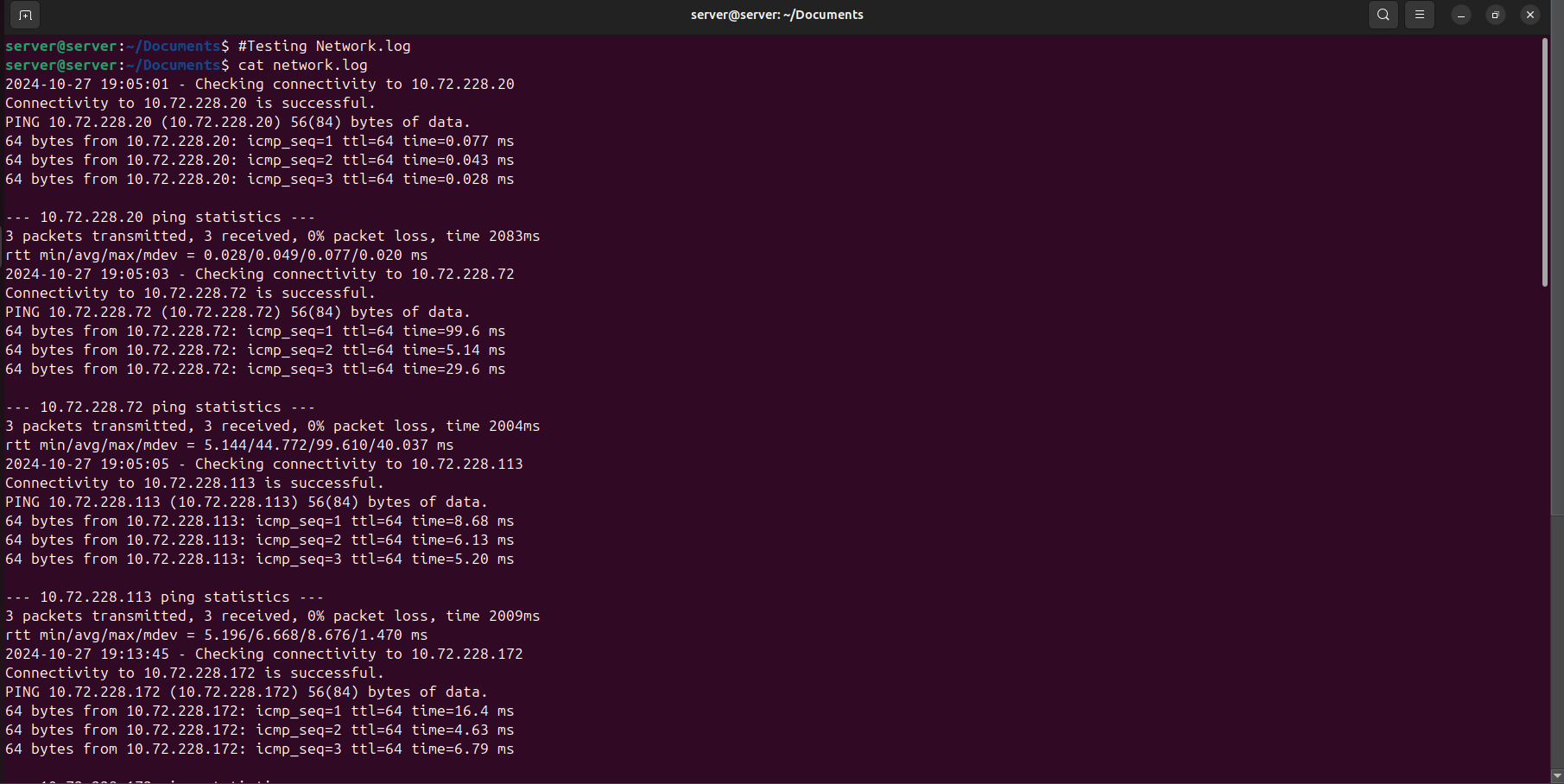
**Testing Network.sh and traceroute scripts:**

We will conduct two tests. First, we’ll ping the server itself to observe the output of the Network.sh script in the case of a successful connection. The second test will use an invalid IP address to examine the output of the traceroute.sh script.:

****

**Result of running (Network.sh) & (traceroute.sh):**

****

**Checking that that script logs all activities related to network connectivity:**

**System.sh script**

#!/bin/bash

# System.sh - Script to display disk and memory usage details for the HOME directory

# This script will generate logs for disk usage, memory, CPU details, and save them to log files.

# Written for Operating Systems Lab (CMPS 405)

# Function to log disk usage information

log\_disk\_usage() {

echo "Checking disk usage in the HOME directory..."

local log\_file="disk\_info.log"

{

echo "========== Disk Usage Report =========="

echo "Report Date: $(date)"

echo "----------------------------------------"

echo "Total Disk Space in HOME Directory:"

du -sh ~ 2>/dev/null

echo "\nDisk Usage for Directories and Subdirectories in HOME Directory:"

du -h ~ --max-depth=2 2>/dev/null

echo "----------------------------------------"

} | tee "$log\_file"

echo "Disk usage data logged to $log\_file"

}

# Function to log memory and CPU details

log\_memory\_cpu\_info() {

echo "Gathering memory and CPU data..."

local log\_file="mem\_cpu\_info.log"

{

echo "========== Memory and CPU Info =========="

echo "Report Date: $(date)"

echo "-----------------------------------------"

# Memory usage information (free and used memory percentage)

echo "Memory Usage Summary:"

free\_output=$(free -m)

total\_memory=$(echo "$free\_output" | awk '/^Mem:/ {print $2}')

used\_memory=$(echo "$free\_output" | awk '/^Mem:/ {print $3}')

free\_memory=$(echo "$free\_output" | awk '/^Mem:/ {print $4}')

used\_percentage=$(( (used\_memory \* 100) / total\_memory ))

free\_percentage=$(( (free\_memory \* 100) / total\_memory ))

echo "Memory Used: $used\_percentage%"

echo "Memory Free: $free\_percentage%"

echo "-----------------------------------------"

# CPU information

echo "CPU Information:"

cpu\_model=$(lscpu | grep "Model name" | awk -F ':' '{print $2}' | sed 's/^ \*//g')

cpu\_cores=$(lscpu | grep "^CPU(s):" | awk -F ':' '{print $2}' | sed 's/^ \*//g')

echo "CPU Model: $cpu\_model"

echo "CPU Cores: $cpu\_cores"

echo "-----------------------------------------"

} | tee "$log\_file"

echo "Memory and CPU data logged to $log\_file"

}

# Main execution starts here

# Log disk usage and memory/CPU info

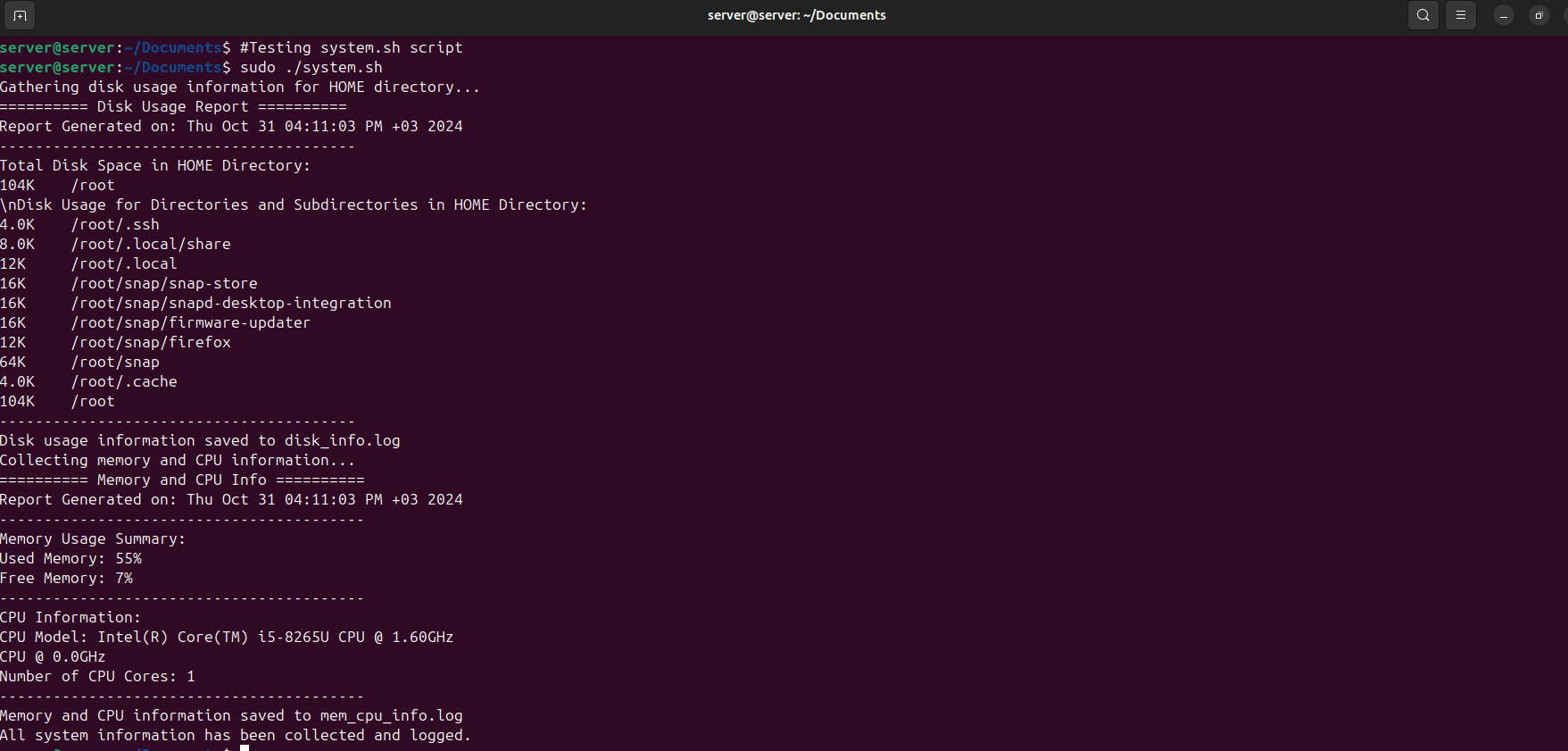
log\_disk\_usage

log\_memory\_cpu\_info

# Final message

echo "System information has been gathered and saved."

**Testing system.sh script:**

****

**Clientinfo.sh Script**

#!/bin/bash

# get the server details from the user

read -p "Enter the server username: " SERVER\_USER

read -p "Enter the server IP address: " SERVER\_IP

read -p "Enter the remote directory path on the server: " SERVER\_DIR

# Directory to store logs on the client

LOG\_DIR="$HOME/process\_logs"

mkdir -p "$LOG\_DIR"

# Log file name with a timestamp

LOG\_FILE="$LOG\_DIR/process\_info.log"

# Gather process information

{

    echo "Process Tree of All Currently Running Processes:"

    pstree -A

    echo -e "\nList of Dead or Zombie Processes:"

    ps aux | awk '$8 ~ /Z/ {print $0}'

    echo -e "\nCPU Usage Related to Processes:"

    top -b -n 1 | head -n 20

    echo -e "\nMemory Usage of Running Processes:"

    free -h

    echo -e "\nTop 5 Resource-Consuming Processes (by CPU usage):"

    ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head -n 6

} > "$LOG\_FILE"

# Securely copy the log file to the server using SCP

scp "$LOG\_FILE" "$SERVER\_USER@$SERVER\_IP:$SERVER\_DIR"

# Instructions for scheduling the script using a cron job

SCRIPT\_PATH=$(realpath "$0") #gets the path of the script

CRON\_JOB="0 \* \* \* \* $SCRIPT\_PATH"

# Check if the cron job is already present

(crontab -l | grep -F "$SCRIPT\_PATH")

if [ $? -ne 0 ]; then #check if exit status of last command not equal to zero

    # If not present, add the cron job

    (crontab -l; echo "$CRON\_JOB") | crontab -

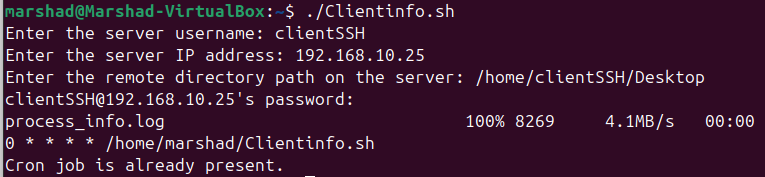
    echo "Cron job added to run the script every hour."

else #if it is equal to zero then it is present

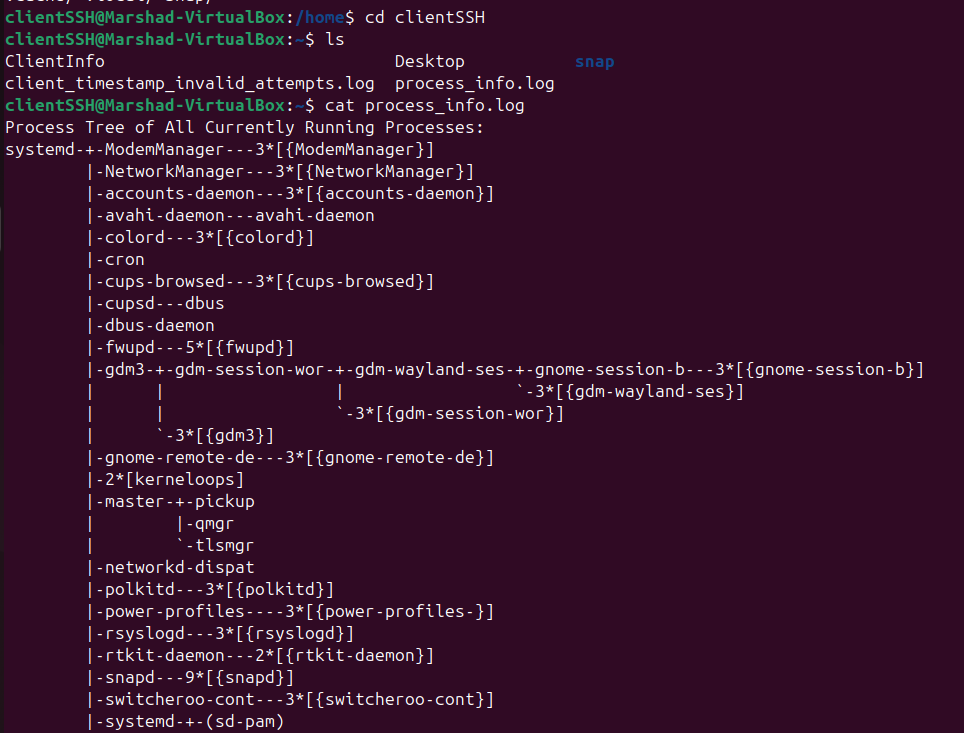
    echo "Cron job is already present."

Fi

**Clientinfo.sh testing:**



**The copied file on the server**

****

**Login.sh script**

#!/bin/bash

LOGFILE="invalid\_attempts.log"

MAX\_ATTEMPTS=3

CORRECT\_USERNAME="server" #we have to change this to to the correct username

CORRECT\_PASSWORD="#cmps450#" # and the password needs to be correct

#function that will log invalid attempts to the logfile

log\_attempt() {

local username=$1

echo "$(date): Invalid server login attempt for user '$username'" >> $LOGFILE

}

# we read the servername and the ip from the client

read -p "Enter Server Name: " servername

read -p "Enter Server IP: " server\_ip

# the server ip has to be correct otherwise sftp is not gonna work

for ((attempt=1; attempt<=MAX\_ATTEMPTS; attempt++)); do

echo

#we get the password for the server here

read -sp "Enter password: " server\_password

# check here if the credentials are correct

if [[ "$servername" == "$CORRECT\_USERNAME" && "$server\_password" == "$CORRECT\_PASSWORD" ]]; then

echo "Credentials are correct proceeding to SSH login..."

# login using SSH

sshpass -p "$server\_password" ssh -o StrictHostKeyChecking=no "$servername@$server\_ip"

exit 0

else

log\_attempt $(whoami)

echo "Invalid server credentials. Attempt $attempt of $MAX\_ATTEMPTS."

fi

done

# if the loop exits it means that max attemps have been reached

echo "Unauthorized server user!"

sshpass -p "$CORRECT\_PASSWORD" sftp -o StrictHostKeyChecking=no "$CORRECT\_USERNAME@$server\_ip" << EOF

put $LOGFILE client\_timestamp\_invalid\_attempts.log

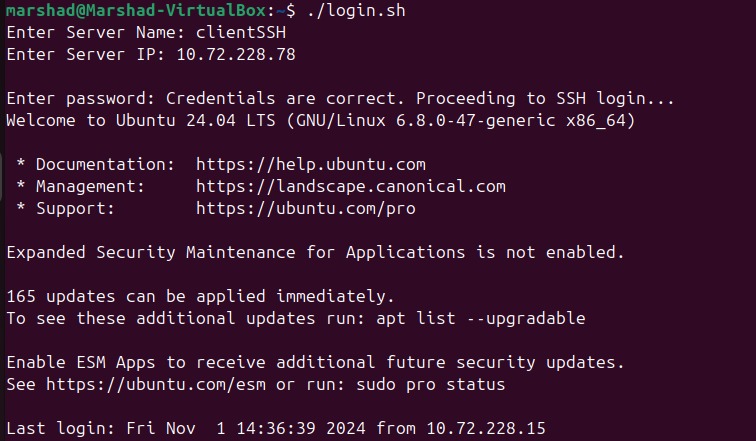
bye

EOF

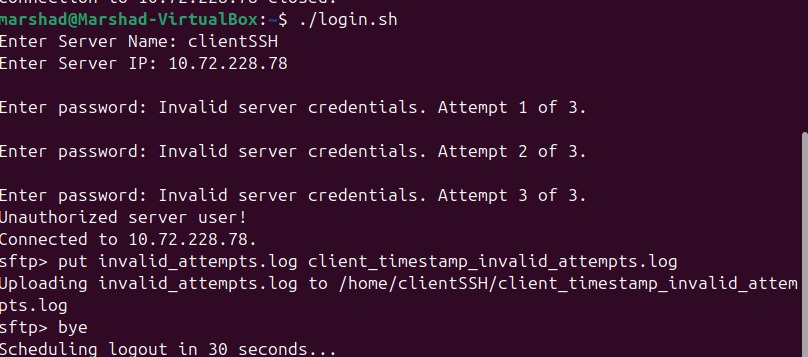
echo "Scheduling logout in 30 seconds... bye bye"

sleep 30 && pkill -KILL -u "$USER" &

**Testing Login.sh script**

****

Login script successful for (client1)



Login script password failed for client1.

**Check.sh script**

#!/bin/bash

# Define log file

LOGFILE="perm\_change.log"

# Find files with permission 777

echo "Searching for files with permission 777..."

files=$(find / -type f -perm 0777 2>/dev/null)

# Check if any files were found

if [ -z "$files" ]; then

echo "No files found with permission 777."

else

echo "Files with permission 777:"

echo "$files"

# Change permissions to 700 and log changes

for file in $files; do

chmod 700 "$file"

echo "Changed permissions for: $file" >> "$LOGFILE"

done

fi

# Display log content

echo "Changes logged in $LOGFILE:"

cat "$LOGFILE"

**Testing check.sh script**

A computer screen shot of a program

Description automatically generated

Here it showed that Check.sh works, if there are any file that has the 777 permission it changes it to 700, also it shows the files that have been changed before to 700.

**Search.sh script**

#!/bin/bash

# search.sh - Script to find files larger than 1M in the user's account

# Logs results to 'bigfile' and emails the system administrator if files are found

# Variables

output\_file="bigfile"

admin\_email="mh1701546@qu.edu.qa"

# Find all files larger than 1M in the user's account

echo "Searching for files larger than 1M in the account..."

{

echo "========== Large Files Report =========="

echo "Report Generated on: $(date)"

echo "----------------------------------------"

# Find and log files larger than 1M

find ~ -type f -size +1M | tee "$output\_file"

file\_count=$(find ~ -type f -size +1M | wc -l)

echo "Number of files larger than 1M: $file\_count"

echo "----------------------------------------"

} | tee "$output\_file"

# If the output file is not empty, email the system administrator

if [ -s "$output\_file" ]; then

echo "Large files found. Sending email to system administrator..."

mail -s "Large Files Report" "$admin\_email" < "$output\_file"

fi

# Final message

echo "Search complete. Results saved to $output\_file."

**Testing search.sh**

In the terminal we see bigfile content and that the email is sent the admin email we set.

A close-up of a screen

Description automatically generated

We had to to ensure the SMPT is working and we used a Gmail account as relay to reach @qu.edu.qa domain server.

**Email received successfully with large files over 1MB:**

**A blue background with black text

Description automatically generated**

**A screenshot of a phone

Description automatically generated**