**CMPS 405 PROJECT**

**Project Phase 1: Shell Scripting  
  
Project Contributors:   
  
1. Sheikh Hasin Ishrak**

**2. Zubair Jashim - 202108715**

**3. Mohammed Al-Adbi - 202009684**

**4.Khalid Haji - 202010980**

**Network.sh:**#!/bin/bash

# checks if target IPs are provided

# checks if user provided any arguments. if less than one then it exists

if [ $# -lt 1 ]; then

echo "Usage: ./network.sh <target\_IPs>"

exit 1

fi

# Define target IPs

TARGET\_IPS=("$@")

LOG\_FILE="network.log"

# Log the current date and time into the log file

log\_date() {

echo "$(date +"%Y-%m-%d %H:%M:%S")" | tee -a $LOG\_FILE

}

# pings the target IP and log the results

ping\_test() {

local target\_ip=$1 # stores arguments in a local variable

for i in {1..3}; do

log\_date # logs date

echo "Pinging $target\_ip (Attempt $i)..." | tee -a $LOG\_FILE # logs attempts

# pings 4 times with a time out response of 5 seconds

# if the ping works, it prints out that it did, logs it and exits the function

if ping -c 4 -W 5 "$target\_ip" > /dev/null; then

echo "Connectivity with $target\_ip is ok" | tee -a $LOG\_FILE

return 0

# if pinging fails, it logs it, and continue by handing the ip address to trace router

else

echo "Ping to $target\_ip failed" | tee -a $LOG\_FILE

fi

done

# If ping fails after 3 attempts, run traceroute.sh

echo "Connectivity with $target\_ip failed after 3 attempts. Running traceroute.sh..." | tee -a $LOG\_FILE

./traceroute.sh "$target\_ip"

}

# Ping each IP given in the command

for ip in "${TARGET\_IPS[@]}"; do

ping\_test "$ip"

done

**Tracerout.sh:**

#!/bin/bash

LOG\_FILE="network.log"

# checks if target IPs are provided

# checks if user provided any arguments. if less than one then it exists

if [ $# -lt 1 ]; then

echo "Usage: ./traceroute.sh <target\_IP>"

exit 1

fi

TARGET\_IP=$1

# Log the current date and time

log\_date() {

echo "$(date +"%Y-%m-%d %H:%M:%S")" | tee -a $LOG\_FILE

}

# log diagnostic info

log\_date

echo "Traceroute diagnostics for $TARGET\_IP" | tee -a $LOG\_FILE

# display the routing table

echo "Displaying routing table:" | tee -a $LOG\_FILE

netstat -r | tee -a $LOG\_FILE

# display hostname

echo "Hostname: $(hostname)" | tee -a $LOG\_FILE

# test local DNS server

echo "Testing local DNS server..." | tee -a $LOG\_FILE

nslookup google.com | tee -a $LOG\_FILE

# traceroute to target IP

echo "Tracing route to $TARGET\_IP..." | tee -a $LOG\_FILE

traceroute "$TARGET\_IP" | tee -a $LOG\_FILE

# check if google.com is reachable

echo "Pinging google.com..." | tee -a $LOG\_FILE

ping -c 4 google.com | tee -a $LOG\_FILE

# reboot the machine if traceroute fails

if ! traceroute "$TARGET\_IP" | grep "ms"; then

echo "Traceroute failed. Rebooting the machine..." | tee -a $LOG\_FILE

reboot

else

echo "Traceroute successful." | tee -a $LOG\_FILE

fi

**System.sh:**

#!/bin/bash

# Define log files

DISK\_LOG="disk\_info.log"

MEM\_CPU\_LOG="mem\_cpu\_info.log"

# Get disk usage information for the HOME directory

echo "=== Disk Usage Information for HOME Directory ===" | tee -a $DISK\_LOG

du -h --max-depth=1 "$HOME" | tee -a $DISK\_LOG

# Show overall disk space information

echo -e "\n=== Overall Disk Space Information ===" | tee -a $DISK\_LOG

df -h "$HOME" | tee -a $DISK\_LOG

# Get memory and CPU information

echo -e "\n=== Memory and CPU Information ===" | tee -a $MEM\_CPU\_LOG

# Get used and free memory percentage

free\_memory=$(free | grep Mem | awk '{print $4}')

total\_memory=$(free | grep Mem | awk '{print $2}')

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

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

# Get CPU model and number of cores

cpu\_model=$(grep -m 1 'model name' /proc/cpuinfo | awk -F: '{print $2}' | xargs)

cpu\_cores=$(nproc)

# Display memory usage

echo "Used Memory Percentage: $used\_memory\_percentage%" | tee -a $MEM\_CPU\_LOG

echo "Free Memory Percentage: $free\_memory\_percentage%" | tee -a $MEM\_CPU\_LOG

echo "CPU Model: $cpu\_model" | tee -a $MEM\_CPU\_LOG

echo "Number of CPU Cores: $cpu\_cores" | tee -a $MEM\_CPU\_LOG

# Display completion message

echo -e "\nInformation logged to $DISK\_LOG and $MEM\_CPU\_LOG."

**Login.sh:**  
  
#!/bin/bash

# Constants

MAX\_ATTEMPTS=3

LOG\_FILE="client\_timestamp\_invalid\_attempts.log"

SERVER\_USER="vm1" # Replace with your server username

SERVER\_IP=""

REMOTE\_LOG\_PATH="/home/$SERVER\_USER/$LOG\_FILE"

# Prompt for server IP if not set

if [ -z "$SERVER\_IP" ]; then

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

fi

# Function to log invalid attempts

log\_invalid\_attempt() {

local username=$1

local timestamp=$(date '+%Y-%m-%d %H:%M:%S')

echo "[$timestamp] Invalid login attempt for user: $username" >> "$LOG\_FILE"

# Copy log file to server using sftp

echo "put $LOG\_FILE $REMOTE\_LOG\_PATH" | sftp "$SERVER\_USER@$SERVER\_IP"

}

# Function to schedule logout

schedule\_logout() {

# Schedule shutdown in 30 seconds

sudo shutdown -h +0.5 "Unauthorized access attempt detected. System will logout in 30 seconds."

}

# Main login attempt handling

attempt\_count=0

while [ $attempt\_count -lt $MAX\_ATTEMPTS ]; do

# Prompt for credentials

read -p "Username: " username

read -s -p "Password: " password

echo

# Attempt SSH login (using sshpass to automate password entry)

if sshpass -p "$password" ssh -o StrictHostKeyChecking=no "$username@$SERVER\_IP" 'exit'; then

echo "Login successful!"

exit 0

else

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

remaining=$((MAX\_ATTEMPTS - attempt\_count))

# Log the invalid attempt

log\_invalid\_attempt "$username"

if [ $remaining -gt 0 ]; then

echo "Invalid credentials. $remaining attempts remaining."

else

echo "Unauthorized user!"

# Schedule logout

schedule\_logout

exit 1

fi

fi

done

**Check.sh:**  
#!/bin/bash

# check.sh - Script to find files with 777 permissions and change them to 700

# Constants

LOG\_FILE="perm\_change.log"

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

# Create or clear the log file

echo "Permission Change Log - Started at $TIMESTAMP" > "$LOG\_FILE"

echo "----------------------------------------" >> "$LOG\_FILE"

# Function to log changes

log\_change() {

local file=$1

local old\_perm=$2

echo "[$TIMESTAMP] Changed permissions for: $file" >> "$LOG\_FILE"

echo " Old permissions: $old\_perm" >> "$LOG\_FILE"

echo " New permissions: 700" >> "$LOG\_FILE"

echo "----------------------------------------" >> "$LOG\_FILE"

}

# Function to display file information

display\_file\_info() {

local file=$1

local perm=$2

echo "Found file with 777 permissions:"

echo " File: $file"

echo " Current permissions: $perm"

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

}

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

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

# Find files with 777 permissions in the current user's home directory

while IFS= read -r file; do

if [ -n "$file" ]; then

# Get the original permissions in human-readable format

old\_perm=$(stat -c "%A" "$file")

# Display information on screen

display\_file\_info "$file" "$old\_perm"

# Change permissions to 700

chmod 700 "$file"

# Log the change

log\_change "$file" "$old\_perm"

fi

done < <(find "$HOME" -type f -perm 777 2>/dev/null)

# Summary

total\_files=$(grep -c "Changed permissions" "$LOG\_FILE")

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

echo "Operation completed at $(date '+%Y-%m-%d %H:%M:%S')"

echo "Total files processed: $total\_files"

echo "Details have been saved to: $LOG\_FILE"

# Add summary to log file

echo "----------------------------------------" >> "$LOG\_FILE"

echo "Summary: Total files processed: $total\_files" >> "$LOG\_FILE"

echo "Operation completed at $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOG\_FILE"

**Search.sh:**  
  
#!/bin/bash

# search.sh - Script to find files larger than 1MB and notify admin

# Configuration

ADMIN\_EMAIL="QUID@qu.edu.qa" # Administrator's email address

BIGFILE="bigfile" # Output file name

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

# Function to send email

send\_email() {

local file\_count=$1

local email\_body="

Date: $TIMESTAMP

Number of files found: $file\_count

File Details:

$(cat "$BIGFILE")

This is an automated message from the file search script.

"

# Using mail command to send email

echo "$email\_body" | mail -s "Large File Report - $TIMESTAMP" "$ADMIN\_EMAIL"

}

# Function to check if mail is installed

check\_mail\_installed() {

if ! command -v mail &> /dev/null; then

echo "Mail command not found. Installing mailutils..."

sudo apt-get update

sudo apt-get install -y mailutils

fi

}

# Create or clear the bigfile

echo "File Search Report - $TIMESTAMP" > "$BIGFILE"

echo "----------------------------------------" >> "$BIGFILE"

echo "Searching for files larger than 1MB..."

# Find files larger than 1M and store details

while IFS= read -r file; do

if [ -n "$file" ]; then

# Get file size in human-readable format

size=$(du -h "$file" | cut -f1)

# Get last modification time

mod\_time=$(stat -c "%y" "$file")

# Add file information to bigfile

echo "File: $file" >> "$BIGFILE"

echo "Size: $size" >> "$BIGFILE"

echo "Last modified: $mod\_time" >> "$BIGFILE"

echo "----------------------------------------" >> "$BIGFILE"

fi

done < <(find "$HOME" -type f -size +1M 2>/dev/null)

# Count the number of files found

file\_count=$(grep -c "File: " "$BIGFILE")

# Add summary to bigfile

echo -e "\nSummary:" >> "$BIGFILE"

echo "Search Date: $TIMESTAMP" >> "$BIGFILE"

echo "Total files found: $file\_count" >> "$BIGFILE"

# Check if any files were found

if [ $file\_count -gt 0 ]; then

echo "Found $file\_count files larger than 1MB."

echo "Results have been saved to $BIGFILE"

# Ensure mail is installed

check\_mail\_installed

# Send email to administrator

echo "Sending email notification to administrator..."

send\_email "$file\_count"

if [ $? -eq 0 ]; then

echo "Email notification sent successfully."

else

echo "Error sending email notification."

fi

else

echo "No files larger than 1MB were found."

echo "No email notification will be sent."

fi

# Display search complete message

echo "Search operation completed at $(date '+%Y-%m-%d %H:%M:%S')"

**Clientinfo.sh:**  
  
#!/bin/bash

# clientinfo.sh - Script to gather and transfer system information hourly

# Configuration

SERVER\_USER="vm1" # Replace with your server username

SERVER\_IP="" # Replace with your server IP

LOCAL\_LOG="process\_info.log"

REMOTE\_PATH="/home/$SERVER\_USER/client\_logs"

TIMESTAMP=$(date '+%Y-%m-%d\_%H-%M-%S')

REMOTE\_FILE="process\_info\_${TIMESTAMP}.log"

# Prompt for server IP if not set

if [ -z "$SERVER\_IP" ]; then

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

fi

# Function to gather process tree

get\_process\_tree() {

echo "=== Process Tree ===" >> "$LOCAL\_LOG"

echo "Timestamp: $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOCAL\_LOG"

pstree -p >> "$LOCAL\_LOG"

echo -e "\n" >> "$LOCAL\_LOG"

}

# Function to find zombie processes

get\_zombie\_processes() {

echo "=== Dead/Zombie Processes ===" >> "$LOCAL\_LOG"

echo "Timestamp: $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOCAL\_LOG"

ps aux | awk '$8=="Z"' >> "$LOCAL\_LOG"

echo -e "\n" >> "$LOCAL\_LOG"

}

# Function to get CPU usage

get\_cpu\_usage() {

echo "=== CPU Usage by Process ===" >> "$LOCAL\_LOG"

echo "Timestamp: $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOCAL\_LOG"

ps aux --sort=-%cpu | head -n 11 >> "$LOCAL\_LOG"

echo -e "\n" >> "$LOCAL\_LOG"

}

# Function to get memory usage

get\_memory\_usage() {

echo "=== Memory Usage by Process ===" >> "$LOCAL\_LOG"

echo "Timestamp: $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOCAL\_LOG"

ps aux --sort=-%mem | head -n 11 >> "$LOCAL\_LOG"

echo -e "\n" >> "$LOCAL\_LOG"

}

# Function to get top resource consumers

get\_top\_resources() {

echo "=== Top 5 Resource-Consuming Processes ===" >> "$LOCAL\_LOG"

echo "Timestamp: $(date '+%Y-%m-%d %H:%M:%S')" >> "$LOCAL\_LOG"

echo "By CPU Usage:" >> "$LOCAL\_LOG"

ps aux --sort=-%cpu | head -n 6 | tail -n 5 >> "$LOCAL\_LOG"

echo -e "\nBy Memory Usage:" >> "$LOCAL\_LOG"

ps aux --sort=-%mem | head -n 6 | tail -n 5 >> "$LOCAL\_LOG"

echo -e "\n" >> "$LOCAL\_LOG"

}

# Function to transfer file to server

transfer\_to\_server() {

# Create remote directory if it doesn't exist

ssh "$SERVER\_USER@$SERVER\_IP" "mkdir -p $REMOTE\_PATH"

# Copy file to server

scp "$LOCAL\_LOG" "$SERVER\_USER@$SERVER\_IP:$REMOTE\_PATH/$REMOTE\_FILE"

if [ $? -eq 0 ]; then

echo "File transferred successfully to server"

# Clean up local log file

> "$LOCAL\_LOG"

else

echo "Error transferring file to server"

fi

}

# Main execution function

gather\_and\_transfer() {

# Clear existing log

> "$LOCAL\_LOG"

# Gather all information

get\_process\_tree

get\_zombie\_processes

get\_cpu\_usage

get\_memory\_usage

get\_top\_resources

# Transfer to server

transfer\_to\_server

}

# Setup cron job if it doesn't exist

setup\_cron() {

# Check if cron job already exists

if ! crontab -l 2>/dev/null | grep -q "$0"; then

# Add to crontab (runs every hour)

(crontab -l 2>/dev/null; echo "0 \* \* \* \* $(readlink -f $0)") | crontab -

echo "Cron job has been set up to run every hour"

fi

}

# If script is run directly (not by cron)

if [ "$1" != "CRON" ]; then

# Set up cron job

setup\_cron

echo "Initial run of information gathering..."

gather\_and\_transfer

echo "Script has been set up to run every hour"

echo "You can check the cron logs for execution details"

else

# When run by cron

gather\_and\_transfer

fi

**Contributions:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Tasks** | **Overall project contribution percentage** |
| Zubair Jashim | Login.sh, Check.sh | 25% |
| Sheikh Hasin Ishrak | Clientinfo.sh | 25% |
|  |  |  |
|  |  |  |