OS LAB 3

22k-5195 Laiba Fatima

IN LAB:

1:

#!/bin/bash

# Log File Analyzer # Accept log file as input log\_file="$1"

# Count total number of requests total\_requests=$(wc -l < "$log\_file") # Determine number of unique IP addresses

unique\_ips=$(awk '{print $1}' "$log\_file" | sort -u | wc -l)

# Identify top 5 most frequent IP addresses

top\_ips=$(awk '{print $1}' "$log\_file" | sort | uniq -c | sort -nr | head -5)

# Calculate total size of data transferred

total\_size=$(awk '{sum += $10} END {print sum}' "$log\_file")

# Generate summary report

echo "Log File Analysis Report" > analysis\_report.txt echo "-------------------------" >> analysis\_report.txt echo "Total Requests: $total\_requests" >> analysis\_report.txt echo "Unique IP Addresses: $unique\_ips" >> analysis\_report.txt echo "Top 5 IP Addresses:" >> analysis\_report.txt echo "$top\_ips" >> analysis\_report.txt

echo "Total Size of Data Transferred: $total\_size bytes" >> analysis\_report.txt echo "Report generated on: $(date)" >> analysis\_report.txt

OR

#!/bin/bash

# Check if the input log file is provided

if [ -z "$1" ]; then

echo "Usage: $0 <log\_file>"

exit 1

fi

# Assign the input log file to a variable

logfile="$1"

# Count the total number of requests

total\_requests=$(wc -l < "$logfile")

# Determine the number of unique IP addresses

unique\_ips=$(awk '{print $1}' "$logfile" | sort | uniq | wc -l)

# Identify the top 5 most frequent IP addresses and their corresponding request counts

top\_ips=$(awk '{print $1}' "$logfile" | sort | uniq -c | sort -rn | head -n 5)

# Calculate the total size of data transferred

total\_size=$(awk '{sum+=$NF} END {print sum}' "$logfile")

# Generate a summary report

summary\_report="Total requests: $total\_requests\n"

summary\_report+="Unique IP addresses: $unique\_ips\n"

summary\_report+="Top 5 most frequent IP addresses:\n$top\_ips\n"

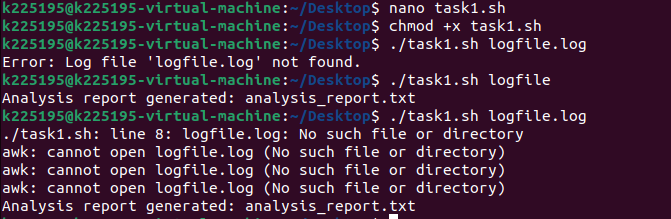
summary\_report+="Total data transferred (in bytes): $total\_size\n"

# Save the summary report to an output file

output\_file="log\_summary\_report.txt"

echo -e "$summary\_report" > "$output\_file"

echo "Summary report generated: $output\_file"



POST LAB:

1:

#!/bin/bash

directory\_path="$1"

new\_file\_name\_pattern="$2"

if [ ! -d "$directory\_path" ]; then

echo "Error: Directory not found."

exit 1

fi

if [ -z "$(ls -A "$directory\_path")" ]; then

echo "Error: Directory is empty."

exit 1

fi

if [ -z "$new\_file\_name\_pattern" ]; then

echo "Error: Please provide a new file name pattern."

exit 1

fi

counter=1

for file in "$directory\_path"/\*; do

if [ -f "$file" ]; then

file\_extension="${file##\*.}"

new\_file\_name="${new\_file\_name\_pattern}${counter}.${file\_extension}"

mv "$file" "$directory\_path/$new\_file\_name"

if [ $? -eq 0 ]; then

echo "Renamed '$file' to '$new\_file\_name'"

else

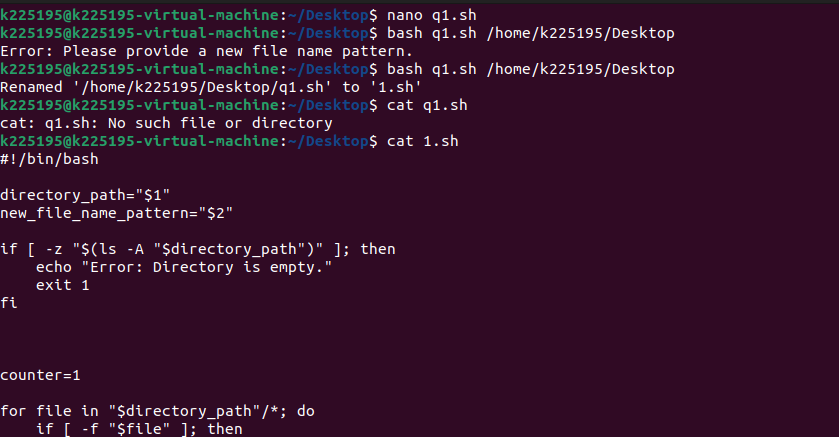
echo "Error: Failed to rename '$file'."

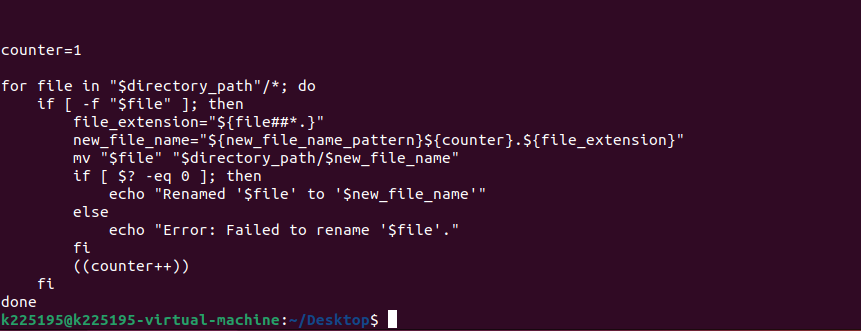
fi

((counter++))

fi

done





2:

#!/bin/bash

# Function to remove old files

remove\_old\_files() {

local directory="$1"

local days="$2"

local count=0

while IFS= read -r -d '' file; do

if [[ -f "$file" ]]; then

if [[ $(($(date +%s) - $(stat -c %Y "$file"))) -gt $((days \* 86400)) ]]; then

rm "$file"

((count++))

fi

fi

done < <(find "$directory" -type f -print0)

echo "Removed $count files older than $days days."

}

# Function to remove empty directories recursively

remove\_empty\_directories() {

local directory="$1"

local count=0

while IFS= read -r -d '' dir; do

if [[ -d "$dir" && -z "$(ls -A "$dir")" ]]; then

rmdir "$dir"

((count++))

fi

done < <(find "$directory" -type d -print0)

echo "Removed $count empty directories."

}

# Main script

if [[ $# -ne 1 ]]; then

echo "Usage: $0 <directory>"

exit 1

fi

directory="$1"

if [[ ! -d "$directory" ]]; then

echo "Error: $directory is not a valid directory."

exit 1

fi

read -p "Enter the number of days to consider as old for files: " days

if [[ ! $days =~ ^[0-9]+$ ]]; then

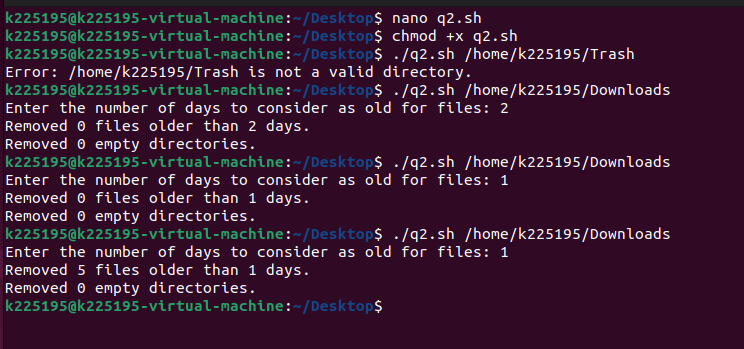
echo "Error: Invalid input for days."

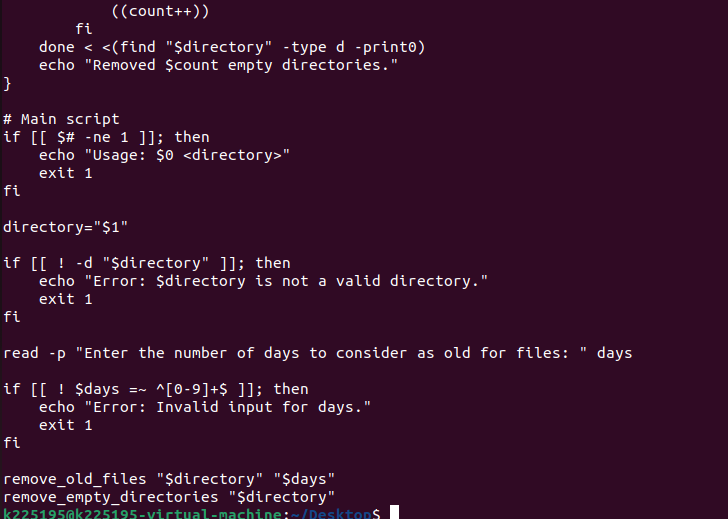
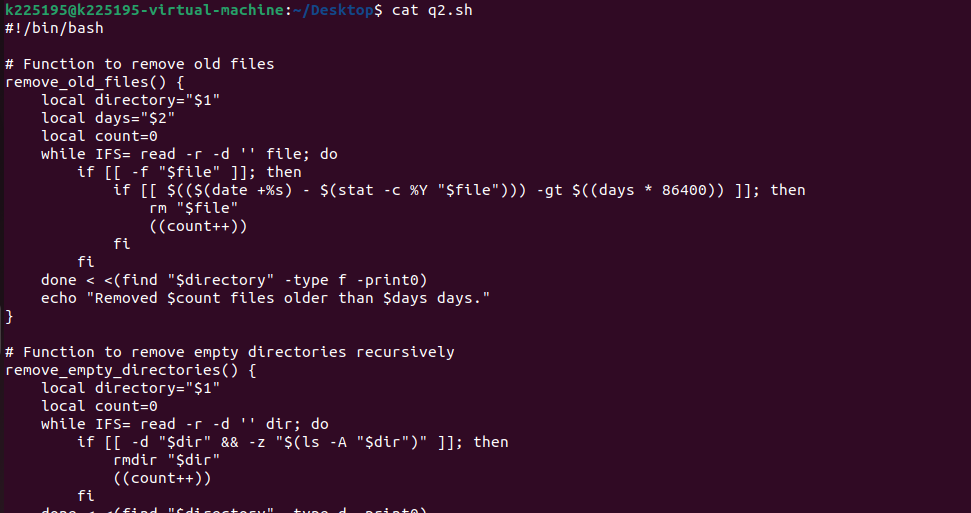
exit 1

fi

remove\_old\_files "$directory" "$days"

remove\_empty\_directories "$directory"





3:

#!/bin/bash

# Function to collect CPU usage

get\_cpu\_usage() {

cpu\_usage=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

echo "CPU Usage: $cpu\_usage%"

}

# Function to collect memory usage

get\_memory\_usage() {

memory\_usage=$(free | grep Mem | awk '{printf "%.2f", $3/$2 \* 100}')

echo "Memory Usage: $memory\_usage%"

}

# Function to collect disk space

get\_disk\_space() {

disk\_space=$(df -h | awk '$NF=="/"{printf "Disk Usage: %s", $5}')

echo "$disk\_space"

}

# Function to collect network traffic

get\_network\_traffic() {

network\_traffic=$(ifstat 1 1 | awk 'NR==3{print "Incoming: " $6 " | Outgoing: " $8}')

echo "$network\_traffic"

}

# Function to calculate average values

calculate\_average() {

duration=$1

# Calculate average CPU usage

avg\_cpu=$(top -bn$duration -d1 | grep "Cpu(s)" | tail -n$duration | awk '{total += (100 - $8)} END {print total/NR}')

# Calculate average memory usage

avg\_memory=$(free -m | awk 'NR==2{total += $3} END {print total/NR}')

echo "Average CPU Usage (over $duration seconds): $avg\_cpu%"

echo "Average Memory Usage (over $duration seconds): $avg\_memory MB"

}

# Function to generate report

generate\_report() {

report\_file="system\_report\_$(date +'%Y-%m-%d\_%H-%M-%S').txt"

echo "System Report" > "$report\_file"

echo "------------------------" >> "$report\_file"

get\_cpu\_usage >> "$report\_file"

get\_memory\_usage >> "$report\_file"

get\_disk\_space >> "$report\_file"

get\_network\_traffic >> "$report\_file"

calculate\_average "$1" >> "$report\_file"

echo "Report generated: $report\_file"

}

# Main script

echo "System Monitoring Script"

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

echo "1. Generate report"

echo "2. Exit"

read -p "Enter your choice: " choice

case $choice in

1)

read -p "Enter duration for average calculation (in seconds): " duration

generate\_report "$duration"

;;

2)

echo "Exiting script."

exit 0

;;

\*)

echo "Invalid choice. Exiting."

exit 1

;;

esac

