

Qatar University

College of engineering

Department of computer science and engineering

Fall\_2021

cmpS 405 – operating systems

14557\_b55

Instructor: Dr. Aws Yousif

Lab Project – Phase 1

September 18, 2021

Marim Elhanafy 201803468

Hagar Elsayed 201805123

Mallak Abunimeh 201903622

1. Group tasks and contribution:

|  |  |  |
| --- | --- | --- |
| Name | Tasks | Contribution percentage |
| Marim Elhanafy | FileSystem.sh  MS-Word document | 40% |
| Hagar Elsayed | Running.sh | 30% |
| Mallak Abunimeh | Performance.sh | 30% |

1. Source code:

* **Running.sh**

#!/bin/sh

display() { #first subroutine

#Display a message only

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "`whoami` is currently logged in the linux system"

echo "The current month calender `cal`"

echo "The time on the linux system is `date`"

idle=`who -u | awk '{print $6 $7}'`

echo "`who -b` and system idle $idle"

echo "The current directory working path is "`pwd`

echo "The current shell is $SHELL"

size=$(du -sh | awk '{print $1}')

echo "My home directory is $HOME with the directory size in my home = $size"

echo "USAGE: This script will run infinitely and check the system time till it reaches the time of 11:59 PM then it will start doing some backups and system checks"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo "Waitng for results..."

}

run() { #second subroutine

time="2359" #Time setted to 11:59 PM

time2="0009" #Time setted to 12:09 AM

while :; do #Infinite loop to keep the program running

currenttime=`date +"%H%M"` #Getting the current time

if [ "$currenttime" -eq "$time" ]; then #Checking if current time is 12 AM or no

sh Connectivity.sh #Calling the Connectivity script

sh ControlTraffic.sh #Calling the ControlTraffic script

sleep 60 #Sleeping for 1 min to force the execution to be 1 time in this min 12:00

fi #Close if statment

if [ "$currenttime" -eq "$time" ]; then #Checking if current time is 12 AM or no

sh Performance.sh #Calling the performance script

sh FileSystem.sh #Calling the FileSystem script

sleep 60 #Sleeping for 1 min to force the execution to be 1 time in this min 12:00

fi #Close if statment

done #End of while loop

}

display #To call the first subroutine

run #To call the second subroutine

trap "`sh Trap.sh`" SIGINT SIGTERM

* **Performance.sh**

#!/bin/sh

diskUsage(){ #First subroutine

du -h ~ > tmp.txt #To get the disk usage of home and save it to file

sort -r -o Disk\_Usage.txt tmp.txt #To sort by reversing the disk usage of home and save it to file

rm tmp.txt

dmesg > memo-HDMessage\_Log.txt #To get the kernal output and save it to file

lscpu > cpu\_inf.txt #To get the CPU information and save it to file

wc -w memo-HDMessage\_Log.txt > Message\_Count.txt #To get the word count of kernal output and save it to file

tar -cvf Phase1.tar.gz Disk\_Usage.txt cpu\_inf.txt Message\_Count.txt #To create compressed folder with the 4 files

time=`date +"%H%M%S"` #To get the current time

mkdir "$time" #Create directory and name it with the current time

mv Phase1.tar.gz "$time" #To move the compressed file to the created directory

}

permissions(){ #Second subroutine

dirName=`date +"%y%m%d%H%M%S"` #To get current date and Time

mkdir ~/"$dirName" #Create directory with current date and Time

for file in ~/\*; do #for loop to trace the home directory

if [ -w $file ]; then #Checkes if file has write permission

if [ -r $file ]; then

cp $file ~/"$dirName" #Copy that file to the created directory

chmod u=r ~/"$dirName"$file #To set the copied file to be read only

fi #end of if statment

fi

done #end of for statment

}

CountEnd(){ #Third subroutine

echo "Number of files that has read: "

ls ~/"$dirName" | wc -l #To get the number of files that has read permission

echo "The backup is completed..." #End of backup message

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

}

diskUsage #To call the first subroutine

permissions #To call the second subroutine

CountEnd #To call the third subroutine

* **FileSystem.sh**

#!/bin/sh

dateTime(){

OUTFILE="OUTFILE.txt" #To store the file that will contain the output

cat /dev/null > $OUTFILE #To clear the file

HOLDFILE="HOLDFILE.txt" #To store the file that will contain the holded data

cat /dev/null > $HOLDFILE #To clear the file

DATESTAMP=`date +"%h %d %Y at %T"` #variable store date

echo "Date/Time of Search: "$DATESTAMP"" >> $OUTFILE #To save the date and time of operation to outfile

}

findFiles(){

echo "Searching for Files Larger Than 8 Mb starting in $HOME" #just a message

echo "Please Standby for the Search Results..."

find ~ -size +8M >> $HOLDFILE #To save the files that has size larger than 8 MB in holdfile

FILESIZE=$(stat -c%s "$HOLDFILE") #Variable to get the size of holdfile

if [ "$FILESIZE" -eq "0" ] #To check that if it's empty or no

then

echo "No files were found that are larger than 8 MB"

echo "Exiting..."

exit

fi

#if the holdfile is not empty so:

NUMOFFILES=`wc -l < $HOLDFILE` #get the number of lines in the holdfile that represent number of files

echo $NUMOFFILES >> $OUTFILE #Write the number of file to the outfile

echo "Number of files found: $NUMOFFILES" #display the number of files found

cat $HOLDFILE >> $OUTFILE #To write the data of holdfile to outfile

}

printOutput(){

cat $OUTFILE #To read the outfile

echo "These search results are stored in `readlink -f $OUTFILE`" #To get the file path

echo "Search complete...Exiting..."

}

dateTime

findFiles

printOutput

1. Output console view:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated