Qassim University
College of Computer | كلية الحاسب
جامعة القصيم

System Administration and Maintenance

Project Part (1)

Table of Contents

Introduction

The field of system administration and maintenance is crucial in the rapidly changing world of technology. In line with the concepts and abilities covered in the "*System Administration and Maintenance*" course, this report explores the creation of automated scripts to maximize a variety of system administration tasks. The primary goals of the course are to increase productivity, decrease human error, and maintain the seamless functioning of computer systems through the use of automation, which is what these scripts seek to accomplish.

The report is divided into four sections, each of which focuses on different aspects of system administration: process monitoring, port status checking, weekly encrypted backups, and user activity logging. These responsibilities support effective system management and are in line with the "*System Administration and Maintenanc*e" curriculum.

Weekly Scheduled Encrypted Backup

1. Write a script using your favorite editor. The script should do a weekly scheduled encrypted backup for your main partition (where the programs and files exists). then send the backup file to your email (you may need to install some packages).

```
#!/bin/bash
# source where the main partion and destination whre the backup will be
OURCE_DIR="/dev/sda1/"
DESTINATION_DIR="/home/lama/"
```

- in this box we have 2 variables

first one : **SOURCE_DIR** This variable stores the source directory or partition from which the backup will be performed.
second one : **DESTINATION_DIR** This variable stores the destination directory where the backup will be stored.

```
# the email address to send the backup file to
EMAIL_ADDRESS="lamazz600@gmail.com"
# the encryption password
ENCRYPTION_PASSWORD="1234567890"
```

- in this box we have 2 variables

EMAIL_ADDRESS: This variable stores the email address to which the backup file will be sent.
ENCRYPTION_PASSWORD: This variable stores the encryption password that will be used to encrypt the backup file.

```
# Generating a timestamp for the backup file
TIMESTAMP=$(date +%Y%m%d%H%M%S)
BACKUP_FILE="backup_${TIMESTAMP}.tar.gz"

# Creating the backup archive
tar -czf "${BACKUP_FILE}" "${OURCE_DIR}"
```

- **TIMESTAMP** :This line generates a timestamp using the `date` command, which will be used to create a unique backup file name.
- **BACKUP_FILE** :This line creates the backup file name by combining the prefix "backup_" with the generated timestamp and the file extension ".tar.gz".
- **`tar -czf "${BACKUP_FILE}" "${SOURCE_DIR}`** : This line uses the `tar` command with options `-czf` to create a compressed archive (`-c`), compress it using gzip (`-z`), and save it to the specified backup file name (`-f`). The source directory is specified as the content to be backed up.

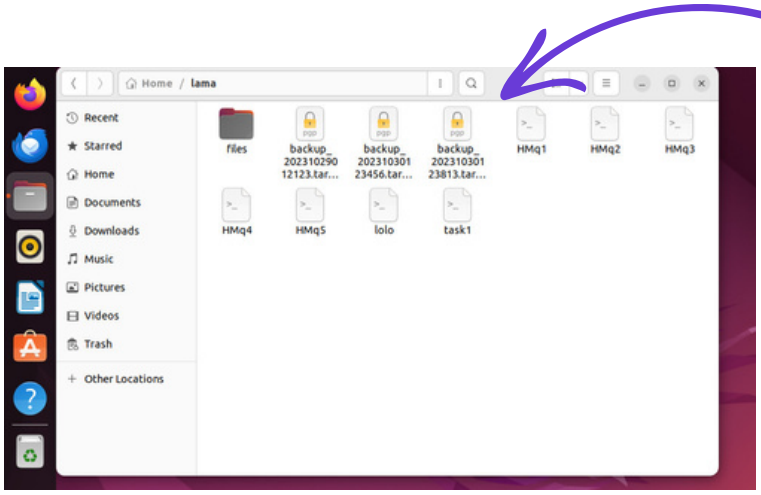
```
# Encrypt the backup archive
gpg --batch --yes --passphrase "${ENCRYPTION_PASSWORD}" -c
"${BACKUP_FILE}"
# Remove the unencrypted backup file
rm "${BACKUP_FILE}"
```

- **gpg --batch --yes --passphrase "${ENCRYPTION_PASSWORD}" -c "${BACKUP_FILE}** : This line uses the `gpg` command to encrypt the backup file. The `--batch` and `--yes` options prevent interactive prompts, and `--passphrase` specifies the encryption password. The encrypted backup file will have the extension ".gpg".

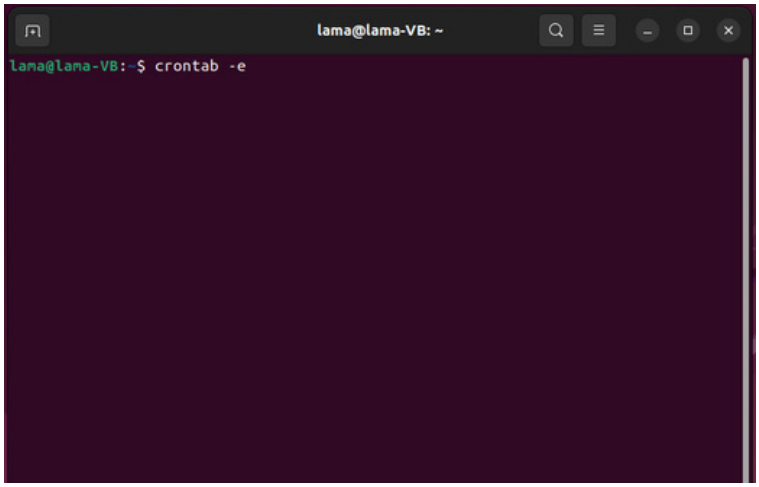
- **rm "${BACKUP_FILE}"**: This line removes the unencrypted backup file to ensure only the encrypted version remains.

```
# Send the encrypted backup file to the email address
echo "Encrypted backup file attached." | mutt -s "Weekly Backup" -a
"${BACKUP_FILE}.gpg" -- "${EMAIL_ADDRESS}"
```

- **echo "Encrypted backup file attached." | mutt -s "Weekly Backup" -a "${BACKUP_FILE}.gpg" -- "${EMAIL_ADDRESS}"**: This line sends an email using the `mutt` command. The email message body is specified as "Encrypted backup file attached." The subject of the email is "Weekly Backup." The encrypted backup file is attached using the `-a` option, and the recipient's email address is specified at the end.



- Here is the encrypted backup file



- In order to perform the weekly encrypted backup we use the command crontab to make the scheduling

- After clicking enter window will open to set the schedule at any interval
- ctrl to save
- Now automatically there will be an encrypted backup every Sunday at 2 AM

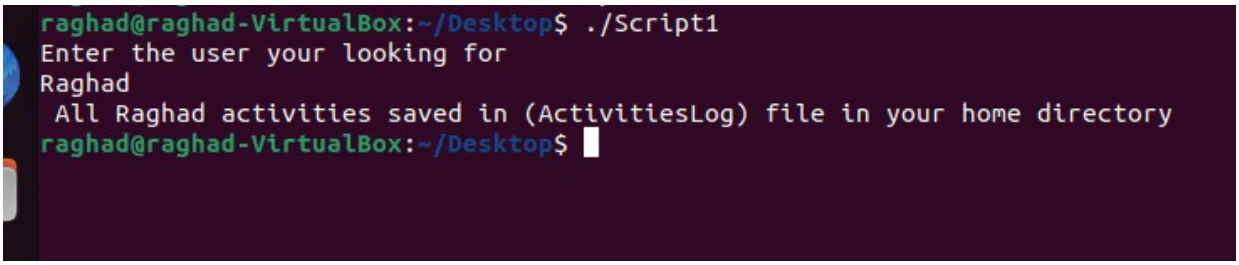
Saving User Activities in The Log File

2. Write a script in which you get a user name as input, then search for all activities related to this user registered in the authentication log file. The results should be saved in a file named ActivitiesLog.txt under your home directory

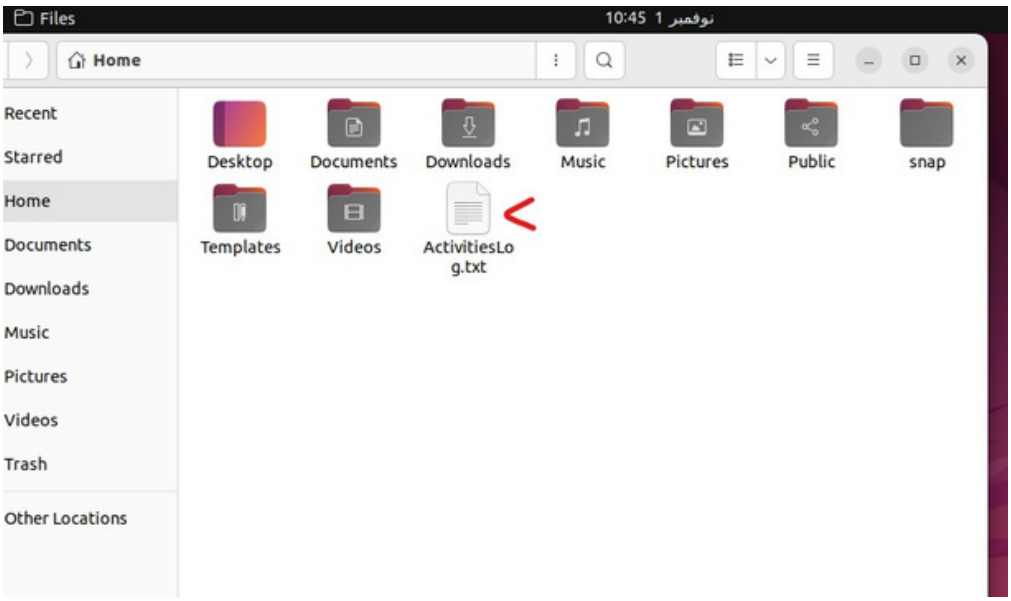
```
#!/bin/bash
#Take input from user.
echo "Enter the user your looking for "
read User

#Log file path.
log_file="/var/log/auth.log"

 grep "$User" "$log_file" > ~/ActivitiesLog.txt

 echo " All $User activities saved in (ActivitiesLog) file in your home directory"
```

- The authentication actions associated with a user-specified username are recorded by this script. whenever the user enters their username, it uses **grep** to search the **"/var/log/auth.log"** file for occurrences of that username and stores the results in "**ActivitiesLog.txt**" in the user's home directory (\). The user is then notified that the log capture was successful.

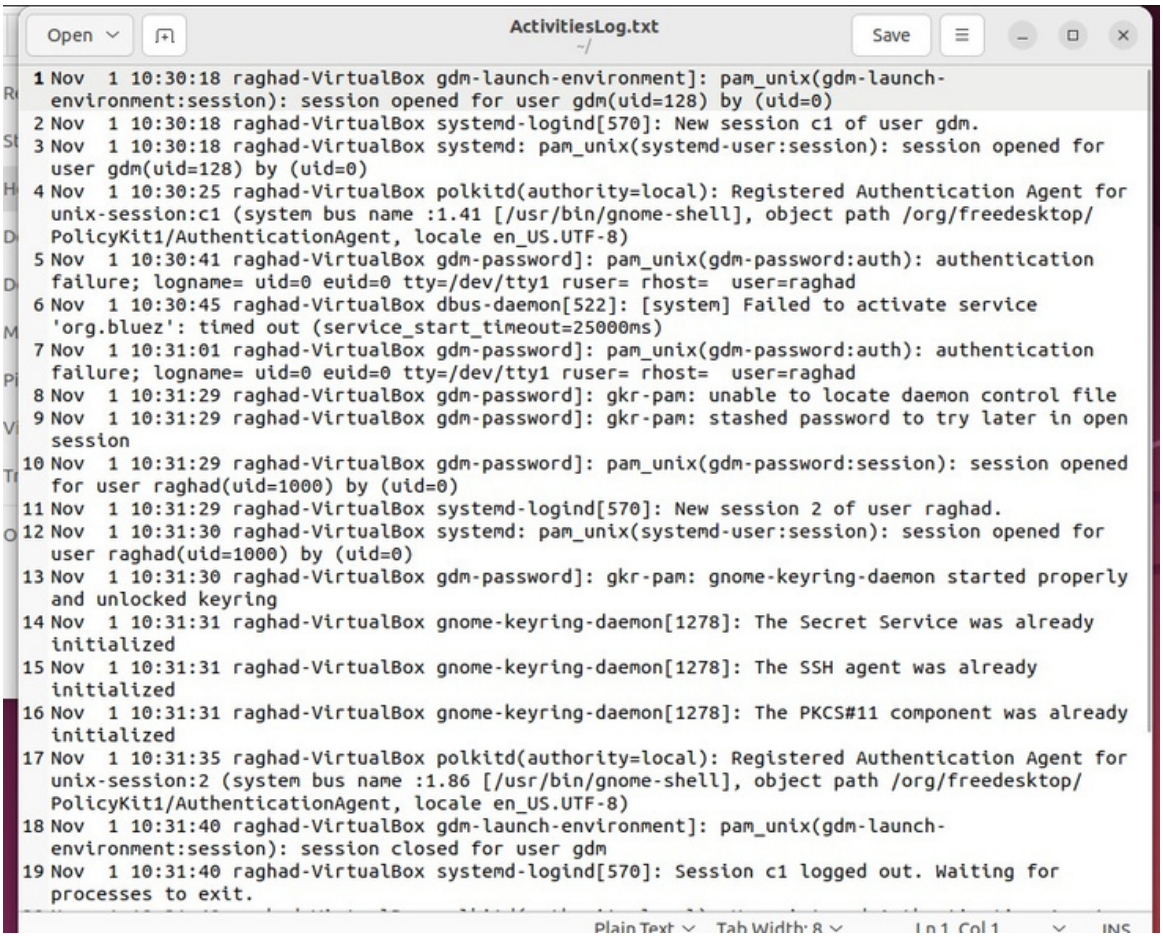
- Upon execution, this code prompts the user to input a username, searches the authentication log file for related entries, and saves them in "ActivitiesLog.txt." It then confirms the successful completion of this action with a message.



- “ActivitiesLog.txt” file location :



- “ActivitiesLog.txt” contents :



List The Top Running processes

3-Write a script in which you get a user name as input, then list the top 5 running processes run by the given user and consume the most memory on your machine.

```
#!/bin/bash

#Asking the user for username
echo "Please enter the username"
read username

#list information about top 5 processes
ps aux pid,user,%mem,%cpu,comm --sort=-%mem | awk -v user="$username" '$2 == user
{ print }' | head -n 5
```

- **The `ps`** command is used to list the processes running on the system
- **-e**: Selects all processes**.**
- **-o:** allow you to specify attributes
- **--sort=-%mem:** Sorts the output based on the `%mem ,so highest memory usage appear first.
- **awk command** is a powerful text processing tool. It processes the input data line by line, applying the specified pattern ($2 == user). In this case, the pattern matches lines where the second field (the username) matches the input username.
- **The head -n 5** it print only the first 5 lines**.**

- The output of the script will be a list of the five processes that consume the most memory and CPU for the specified user after asked the user to enter username

```
amal27@amal27-VirtualBox:~/Desktop/project$ sh task3.sh
Please enter the username
amal27
   3515 amal27    9.9  8.5 Isolated Web Co
   2803 amal27    4.7  0.3 firefox
   1346 amal27    4.7  0.3 gnome-shell
   3094 amal27    1.3  0.0 Privileged Cont
   5772 amal27    1.2  0.0 Isolated Servic
```

Checking Port Number Status

4. Write a script to check if a given port (command line argument) is open or not, based on the firewall configuration and print a message indicating the status of the port.

```
#!/bin/bash
# Check if the script is provided with exactly one argument. If not, it prints an error message
if [ "$#" -ne 1 ]; then
 echo "Error: Incorrect number of arguments provided."
 echo "Please Enter only one Argument."
 # Exit the script with an error status
 exit 1
fi
```

- This block checks if the number of input arguments provided to the script is not equal to 1. The variable $# holds the number of arguments. If there is not exactly one argument, it prints a error message to the user, then exit with with 1 indicating there's incorrect output

```
# Save the first argument as the port number
port="$1"
# Define the timeout duration in seconds for processing time
timeout=1

# nc to check if the specified port on the local machine (localhost) is open within the
specified timeout
# -zv perform a port scan in a way that is non-intrusive
# -w sets the timeout duration
nc -zv -w "$timeout" localhost "$port" &>/dev/null
```

- First, the script assigns the first argument passed to it (the port number) to a variable named **PORT**, and defines another variable **timeout** with a value of **1** to represent the duration in seconds.
- then, the **nc** command (netcat) a utility for reading from and writing to network connections. Is used to check if the specified port on the local machine (localhost) is open within the specified timeout.
- The **-zv** performs a port scan in a way that is non-intrusive, allowing you to determine whether a particular port is open without establishing a full connection.
- **&>** this is the syntax used for redirecting both standard output (stdout) and the standard error (stderr) command.
- **/dev/null** this is a special device file in Unix-like systems that discards all data written to it and provides no data to any process that reads from it. This ensures that the script only displays the desired messages about the status of the port and keeps the terminal output clean without any additional information.

```
# Check the exit status of the last command
if [ "$?" -eq 0 ]; then
 # If the exit status is 0, the port is open
 echo "The Port $port is open"
else
 # If the exit status is not 0, the port is closed
 echo "The Port $port is closed"
fi
```

- This block checks the **$?** variable that holds the exit status of the last command. If the exit status is 0 (Correct Output), it means the port is open, so it prints a message confirming that. If the exit status is not 0(Incorrect Output), it means the port is closed, so it prints a message indicating that.

- First, by running this command, it will perform a comprehensive TCP connect scan on all ports of the localhost, providing information about the status of each port, whether they are open, closed, or filtered

```
khuzama@ubuntuonm2:~/Desktop/project$ sudo nmap -n -PN -sT -p- localhost
[sudo] password for khuzama:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-02 18:55 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000034s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
631/tcp open  ipp

Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds
khuzama@ubuntuonm2:~/Desktop/project$
```

- Then, after running the script with the desired port number as a command line argument. If the port is open or closed, it will print the appropriate message.

```
khuzama@ubuntuonm2:~$ cd Desktop
khuzama@ubuntuonm2:~/Desktop$ cd project
khuzama@ubuntuonm2:~/Desktop/project$ ./task4.sh 22
The Port 22 is open
khuzama@ubuntuonm2:~/Desktop/project$ ./task4.sh 631
The Port 631 is open
khuzama@ubuntuonm2:~/Desktop/project$ ./task4.sh 80
The port 80 is closed
khuzama@ubuntuonm2:~/Desktop/project$ ./task4.sh 22 631
Error: Incorrect number of arguments provided.
Please Enter only one Argument.
khuzama@ubuntuonm2:~/Desktop/project$
```

REFERENCES

how-to-schedule-tasks-on-ubuntu

Linux Log Files Location And How Do I View Logs Files on Linux?

How to List Running Processes in Linux

ps command in Linux with Examples

How To Find Top Running Processes

How to check open ports

Check If a Port is Open in Linux

How to Check Open and Listening Ports on Linux