IT361 PROJECT
G-10

SYSTEM ADMINISTRATION AND MAINTENANCE 2023 – 2024 (FIRST SEMESTER) PROJECT - PART 1

BY:
REVAN ALGHANAYM 411202102
REEM ALRSHEED 411202119

INTRODUCTION

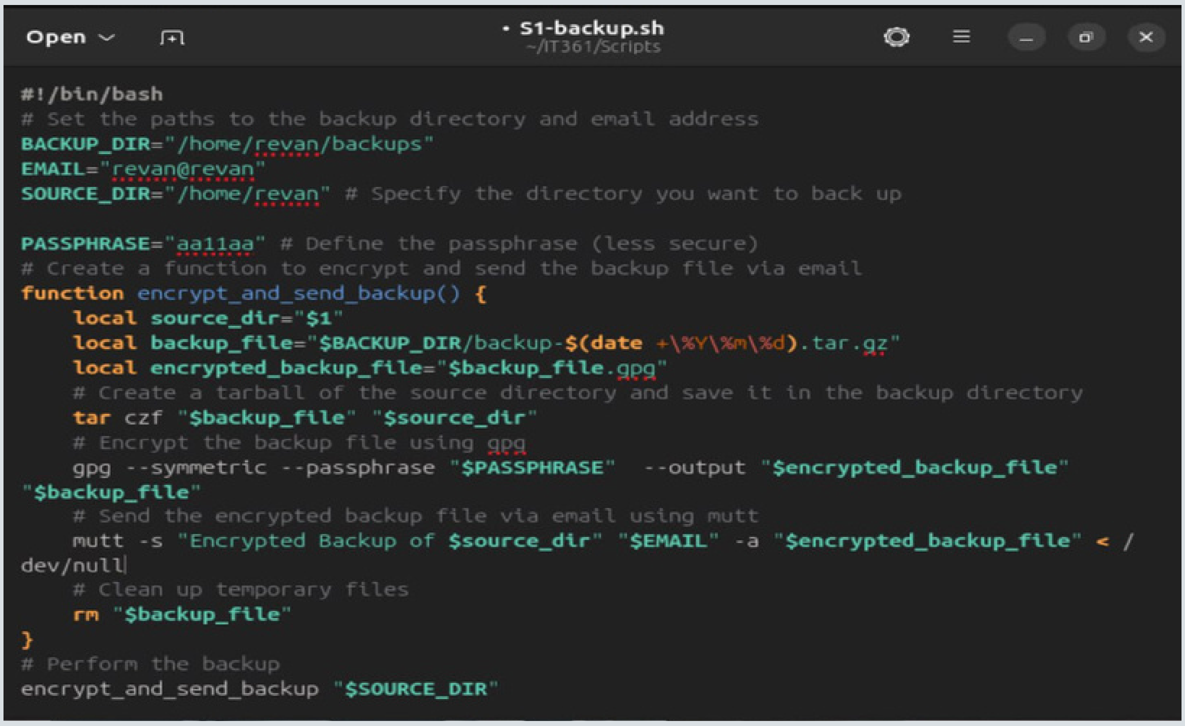
This report discusses four crucial tasks that cover system administration and security. These jobs are intended to strengthen system defences, improve user access and activity monitoring, and automate necessary processes.

The first task was Automated Encrypted Backup where a script is created to conduct scheduled encrypted backups, ensuring data security and redundancy. Then the second task was User Activity Logging,This task focuses on monitoring user activities, with the script gathering and organizing this data for audit purposes. The third task was Resource Monitoring, A script is developed to identify the top 5 memory-intensive processes initiated by a specified user, optimizing resource allocation. And the last task is Port Status Verification, The final task evaluates port accessibility based on firewall settings, delivering crucial information for network security.

As we outline each task's script, its explanation, and the output for each one.

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).

The script



```
#!/bin/bash
# Set the paths to the backup directory and email address
BACKUP_DIR="/home/revan/backups"
EMAIL="revan@revan"
SOURCE_DIR="/home/revan" # Specify the directory you want to back up

PASSPHRASE="aa11aa" # Define the passphrase (less secure)
# Create a function to encrypt and send the backup file via email
function encrypt_and_send_backup() {
    local source_dir="$1"
    local backup_file="$BACKUP_DIR/backup-$(date +\%Y\%m\%d).tar.gz"
    local encrypted_backup_file="$backup_file.gpg"
    # Create a tarball of the source directory and save it in the backup directory
    tar czf "$backup_file" "$source_dir"
    # Encrypt the backup file using gpg
    gpg --symmetric --passphrase "$PASSPHRASE"  --output "$encrypted_backup_file"
"$backup_file"
    # Send the encrypted backup file via email using mutt
    mutt -s "Encrypted Backup of $source_dir" "$EMAIL" -a "$encrypted_backup_file" < /
dev/null
    # Clean up temporary files
    rm "$backup_file"
}
# Perform the backup
encrypt_and_send_backup "$SOURCE_DIR"
```

The packgeas we needed -> tar, gnupg, mutt

How the Script works:

- First we specify the directory where the backups will be stored. `BACKUP_DIR=“/home/revan/backups”`.
- Then the email that address to which the backup will be sent.`EMAIL=“revan@revan”`.
- Than the source directory that you want to back up `SOURCE_DIR=“/home/revan”`.
- After that we defines a passphrase for encrypting the backup file. `PASSPHRASE=“aa11aa”
- Then inside the function we defined the source directory and the name of the backup file and we attach it to the encrypted_backup_file as we will encrypted it .
- The command `**tar czf** "$backup_file" “$source_dir"` creates a compressed tarball (tar.gz) of the source directory and saves it in the `backup_file`.
- We still need to encrypt the file so we use GnuPG (**gpg**) command to symmetrically encrypt the `backup_file` using the passphrase provided in `PASSPHRASE`, and it saves the result as the `encrypted_backup_file`.
- And then using the **Mutt** email client command to sends the email to the address
- Lastly in the function is command **rm** “$backup_file" to remove and clean up temporary files.
- Last line is to call the function and initiating the backup process.

continue

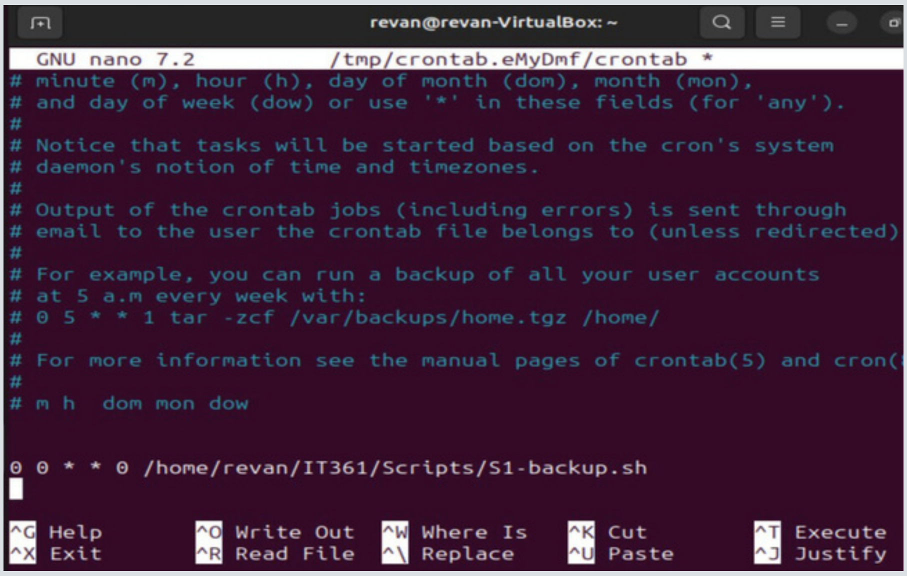
The weekly scheduling:

After writing the script we need to schedule it to run weekly, so we used the cron command Which is a standard method of scheduling tasks to run on the server, we run crontab -e in the terminal to edit the crontab file, then add the following line:

0 0 * * 0 /home/revan/IT361/Scripts/S1-backup.sh

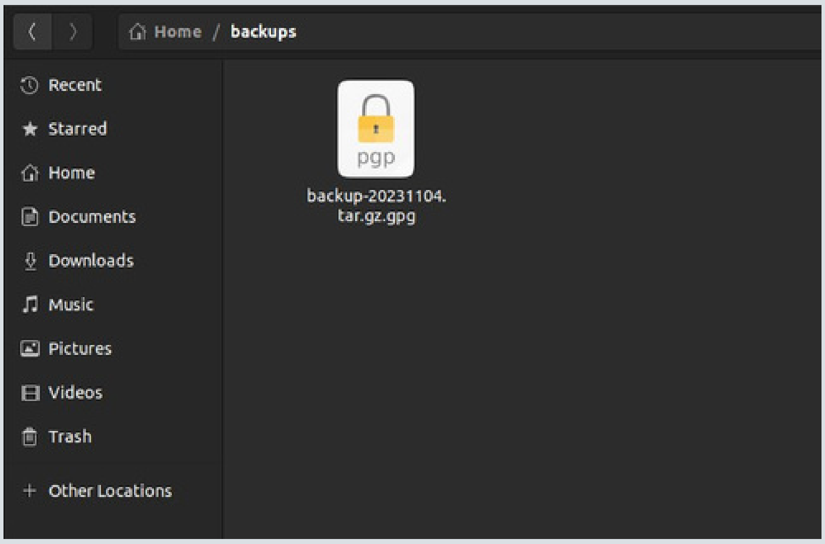
This will run our script every Sunday at midnight (00:00).

The Crontab file



```
revan@revan-VirtualBox: ~
  GNU nano 7.2             /tmp/crontab.eMyDmf/crontab *
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected)
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(
#
# m h  dom mon dow


0 0 * * 0 /home/revan/IT361/Scripts/S1-backup.sh

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

The output

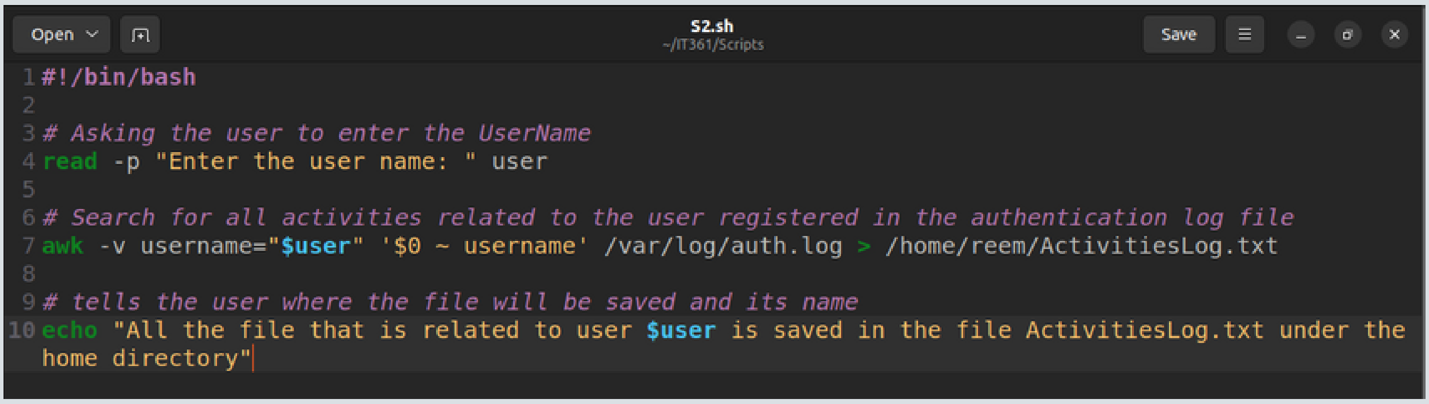


Home / backups

backup-20231104.tar.gz.gpg

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.

The script



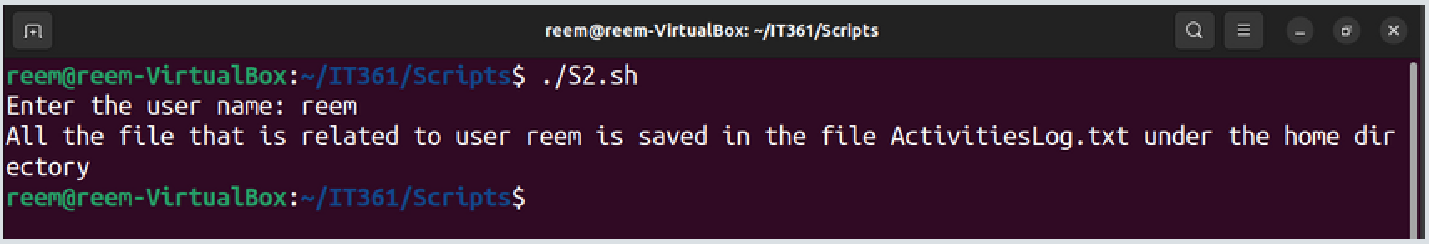
```bash
#!/bin/bash

# Asking the user to enter the UserName
read -p "Enter the user name: " user

# Search for all activities related to the user registered in the authentication log file
awk -v username="$user" '$0 ~ username' /var/log/auth.log > /home/reem/ActivitiesLog.txt

# tells the user where the file will be saved and its name
echo "All the file that is related to user $user is saved in the file ActivitiesLog.txt under the home directory"
```

The packages we needed -> gawk

How the Script works:

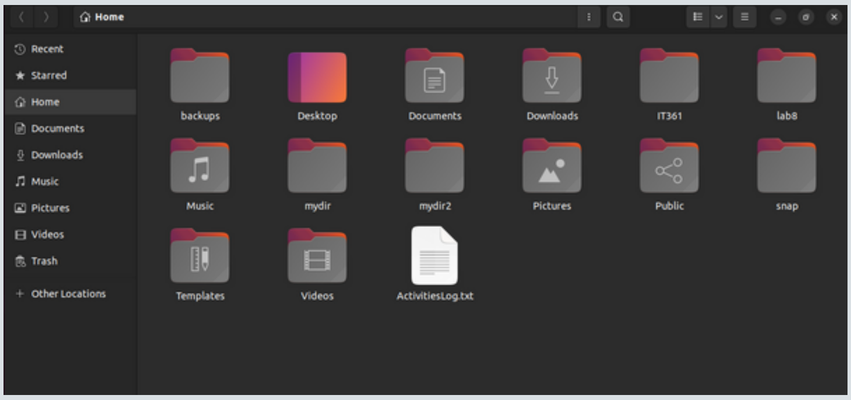
- **awk** command to select data—one or more pieces of individual text—based on a pattern you provide.
- First, we ask the user to enter the userName using read-p.
- then we assign a value to a variable using the 'awk -v username="$user" '
- '$0 ~ username' Search for lines in /var/log/auth.log to check if the value in the username variable is found and printed in > ActivitiesLog.txt file under the path /home/user/ActivitiesLog.txt.

In terminal

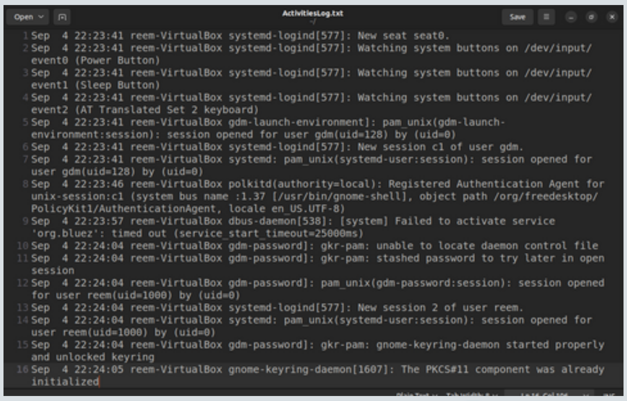


```
reem@reem-VirtualBox:~/IT361/Scripts$ ./S2.sh
Enter the user name: reem
All the file that is related to user reem is saved in the file ActivitiesLog.txt under the home dir
ectory
reem@reem-VirtualBox:~/IT361/Scripts$
```

The ActivitiesLog.txt



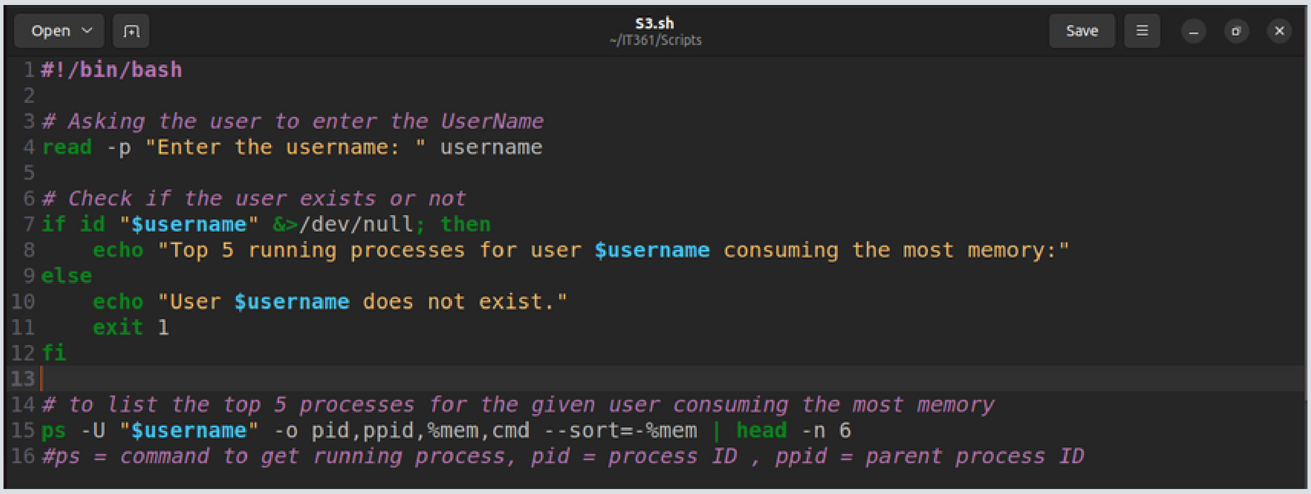
The output



```
Sep  4 22:23:41 reem-VirtualBox systemd-logind[577]: New seat seat0.
Sep  4 22:23:41 reem-VirtualBox systemd-logind[577]: Watching system buttons on /dev/input/event0 (Power Button)
Sep  4 22:23:41 reem-VirtualBox systemd-logind[577]: Watching system buttons on /dev/input/event1 (Sleep Button)
Sep  4 22:23:41 reem-VirtualBox systemd-logind[577]: Watching system buttons on /dev/input/event2 (AT Translated Set 2 keyboard)
Sep  4 22:23:41 reem-VirtualBox gdm-launch-environment]: pam_unix(gdm-launch-environment:session): session opened for user gdm(uid=128) by (uid=0)
Sep  4 22:23:41 reem-VirtualBox systemd-logind[577]: New session c1 of user gdm.
Sep  4 22:23:41 reem-VirtualBox systemd: pam_unix(systemd-user:session): session opened for user gdm(uid=128) by (uid=0)
Sep  4 22:23:46 reem-VirtualBox polkitd(authority=local): Registered Authentication Agent for unix-session:c1 (system bus name :1.37 [/usr/bin/gnome-shell], object path /org/freedesktop/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Sep  4 22:23:57 reem-VirtualBox dbus-daemon[538]: [system] Failed to activate service 'org.bluez': timed out (service_start_timeout=25000ms)
Sep  4 22:24:04 reem-VirtualBox gdm-password]: gkr-pam: unable to locate daemon control file
Sep  4 22:24:04 reem-VirtualBox gdm-password]: gkr-pam: stashed password to try later in open session
Sep  4 22:24:04 reem-VirtualBox gdm-password]: pam_unix(gdm-password:session): session opened for user reem(uid=1000) by (uid=0)
Sep  4 22:24:04 reem-VirtualBox systemd-logind[577]: New session 2 of user reem.
Sep  4 22:24:04 reem-VirtualBox systemd: pam_unix(systemd-user:session): session opened for user reem(uid=1000) by (uid=0)
Sep  4 22:24:04 reem-VirtualBox gdm-password]: gkr-pam: gnome-keyring-daemon started properly and unlocked keyring
Sep  4 22:24:05 reem-VirtualBox gnome-keyring-daemon[1607]: The PKCS#11 component was already initialized
```

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.

The script

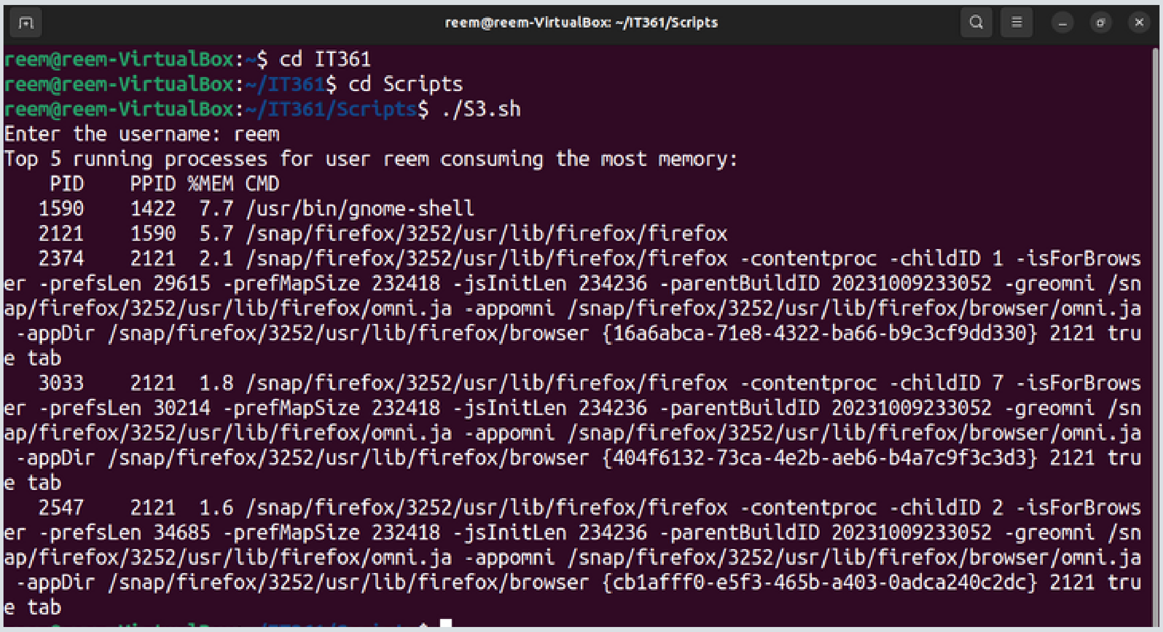


```bash
#!/bin/bash

# Asking the user to enter the UserName
read -p "Enter the username: " username

# Check if the user exists or not
if id "$username" &>/dev/null; then
    echo "Top 5 running processes for user $username consuming the most memory:"
else
    echo "User $username does not exist."
    exit 1
fi

# to list the top 5 processes for the given user consuming the most memory
ps -U "$username" -o pid,ppid,%mem,cmd --sort=-%mem | head -n 6
#ps = command to get running process, pid = process ID , ppid = parent process ID
```

How the Script works:

- **ps** command the process status it used to learn about the system's processes.
- We ask the user to enter the UserName
- Then we used an IF statement to check if the user exists or not.
- The **ps-U option** filters processes based on the authentic user name the user provides .
- **-o** to view processes according to user-defined format, and we use it to define the process ID **'pid'**, parent process ID **'ppid'**, the command that launched the process **'cmd'**, and memory usage **'%mem'** ;that we will use to know the most memory used.
- To show the **'%mem'** in descending order, we used **--sort=-%mem | head -n 6** to display the top 6 lines, the header, and the top 5 processes.

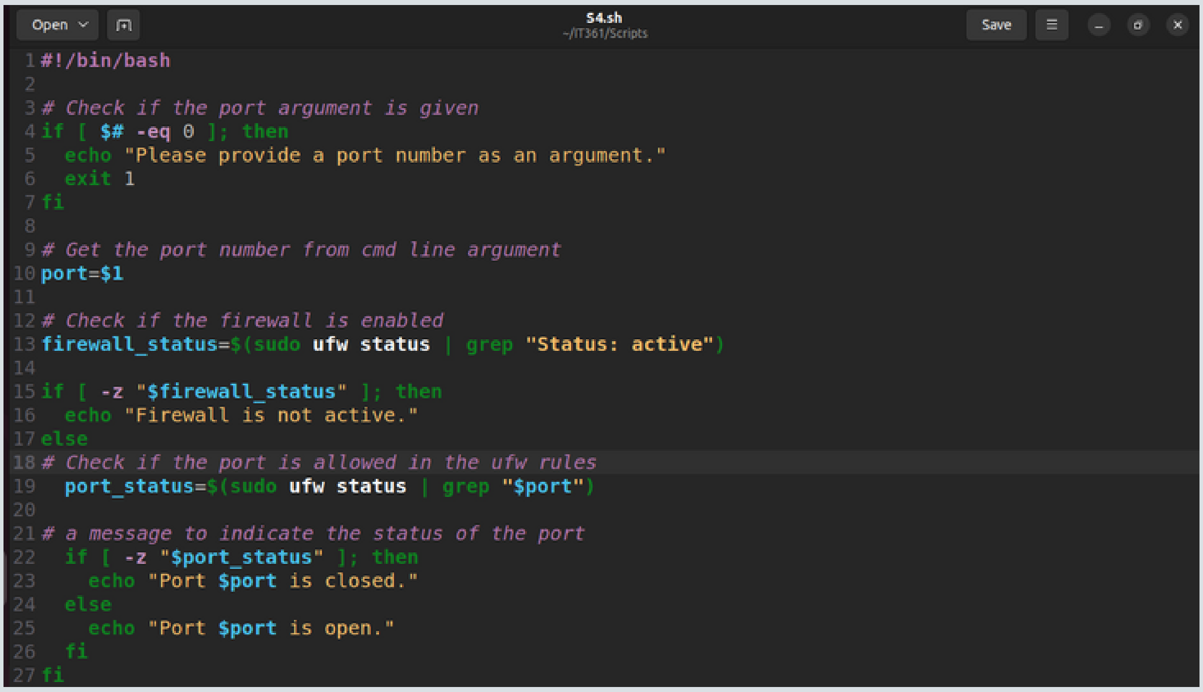
The output



```
reem@reem-VirtualBox:~$ cd IT361
reem@reem-VirtualBox:~/IT361$ cd Scripts
reem@reem-VirtualBox:~/IT361/Scripts$ ./S3.sh
Enter the username: reem
Top 5 running processes for user reem consuming the most memory:
    PID    PPID %MEM CMD
   1590    1422  7.7 /usr/bin/gnome-shell
   2121    1590  5.7 /snap/firefox/3252/usr/lib/firefox/firefox
   2374    2121  2.1 /snap/firefox/3252/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrows
er -prefsLen 29615 -prefMapSize 232418 -jsInitLen 234236 -parentBuildID 20231009233052 -greomni /sn
ap/firefox/3252/usr/lib/firefox/omni.ja -appomni /snap/firefox/3252/usr/lib/firefox/browser/omni.ja
 -appDir /snap/firefox/3252/usr/lib/firefox/browser {16a6abca-71e8-4322-ba66-b9c3cf9dd330} 2121 tru
e tab
   3033    2121  1.8 /snap/firefox/3252/usr/lib/firefox/firefox -contentproc -childID 7 -isForBrows
er -prefsLen 30214 -prefMapSize 232418 -jsInitLen 234236 -parentBuildID 20231009233052 -greomni /sn
ap/firefox/3252/usr/lib/firefox/omni.ja -appomni /snap/firefox/3252/usr/lib/firefox/browser/omni.ja
 -appDir /snap/firefox/3252/usr/lib/firefox/browser {404f6132-73ca-4e2b-aeb6-b4a7c9f3c3d3} 2121 tru
e tab
   2547    2121  1.6 /snap/firefox/3252/usr/lib/firefox/firefox -contentproc -childID 2 -isForBrows
er -prefsLen 34685 -prefMapSize 232418 -jsInitLen 234236 -parentBuildID 20231009233052 -greomni /sn
ap/firefox/3252/usr/lib/firefox/omni.ja -appomni /snap/firefox/3252/usr/lib/firefox/browser/omni.ja
 -appDir /snap/firefox/3252/usr/lib/firefox/browser {cb1afff0-e5f3-465b-a403-0adca240c2dc} 2121 tru
e tab
```

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

The script



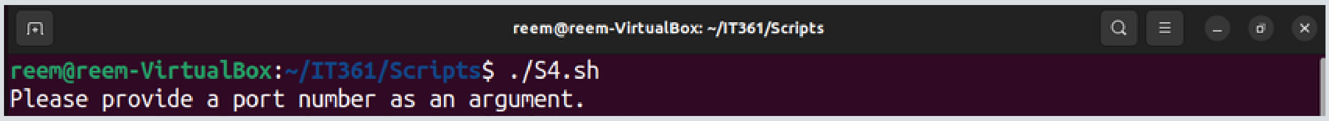
```bash
#!/bin/bash

# Check if the port argument is given
if [ $# -eq 0 ]; then
  echo "Please provide a port number as an argument."
  exit 1
fi

# Get the port number from cmd line argument
port=$1

# Check if the firewall is enabled
firewall_status=$(sudo ufw status | grep "Status: active")

if [ -z "$firewall_status" ]; then
  echo "Firewall is not active."
else
# Check if the port is allowed in the ufw rules
  port_status=$(sudo ufw status | grep "$port")

# a message to indicate the status of the port
  if [ -z "$port_status" ]; then
    echo "Port $port is closed."
  else
    echo "Port $port is open."
  fi
fi
```

The packgeas we needed -> ufw, lighttpd

How the script works:

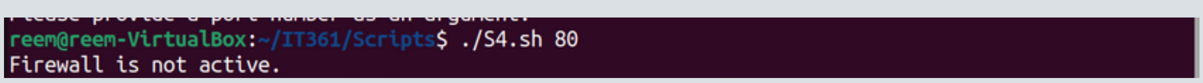
- First, we check whether the port argument is given or not using the **If statement.**
- Then we assign a port argument to get it from the command line.
- Thereafter, we check the status of the firewall to see if it is active or inactive by using the cmd **sudo ufw status | grep "status: active".**
- After checking the firewall status, we check whether the port is allowed in ufw rules or not by using the cmd **sudo ufw status | grep "$port".**
- After checking the port status, we used the if statement to indicate the port status. The cmd **if [-z "varible"] option** is a test to check whether a string is empty; return true or false if it is not.

The command line argument



```
reem@reem-VirtualBox:~/IT361/Scripts$ ./S4.sh
Please provide a port number as an argument.
```

- In the image above, we did not give a port number in cmd.

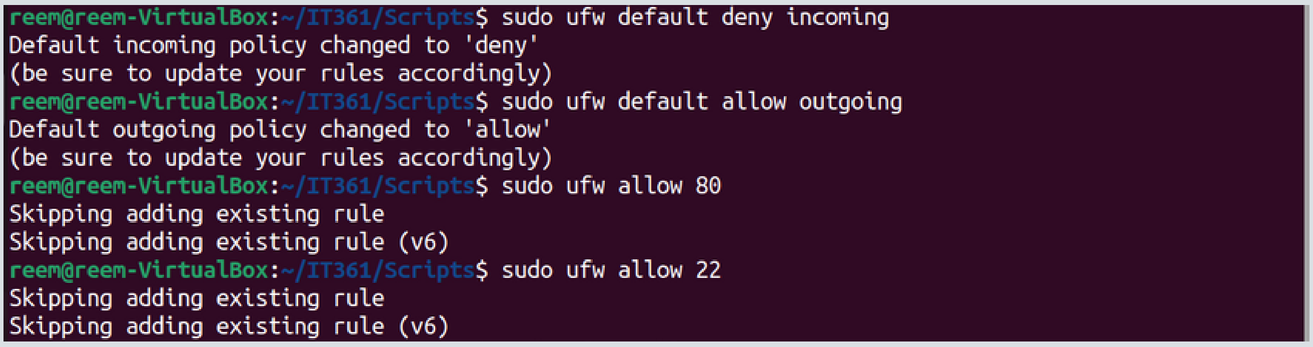


```
reem@reem-VirtualBox:~/IT361/Scripts$ ./S4.sh 80
Firewall is not active.
```

- and here, we did not active the firewall.

continue

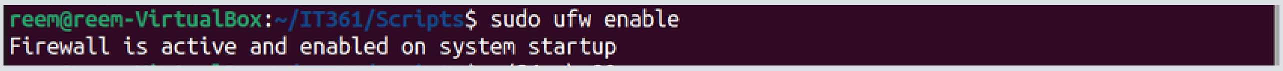
The output



```
reem@reem-VirtualBox:~/IT361/Scripts$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
reem@reem-VirtualBox:~/IT361/Scripts$ sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
reem@reem-VirtualBox:~/IT361/Scripts$ sudo ufw allow 80
Skipping adding existing rule
Skipping adding existing rule (v6)
reem@reem-VirtualBox:~/IT361/Scripts$ sudo ufw allow 22
Skipping adding existing rule
Skipping adding existing rule (v6)
```

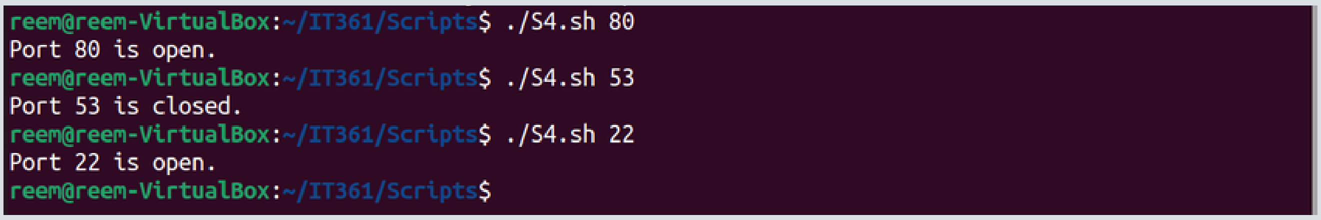
- Therefore, in order to activate the firewall, we must first confirm that we have downloaded the necessary packages, after which we must define its rules by **permitting** all outgoing connections and **rejecting** all incoming ones.
- However, if we enable the firewall now, it will reject all incoming connections, so before we activate it, we have to allow the connection we want the server to respond to, like what's clear in the picture above.

- Here in picture below , the firewall is **activated on.**



```
reem@reem-VirtualBox:~/IT361/Scripts$ sudo ufw enable
Firewall is active and enabled on system startup
```

- and this shows some port's current state.



```
reem@reem-VirtualBox:~/IT361/Scripts$ ./S4.sh 80
Port 80 is open.
reem@reem-VirtualBox:~/IT361/Scripts$ ./S4.sh 53
Port 53 is closed.
reem@reem-VirtualBox:~/IT361/Scripts$ ./S4.sh 22
Port 22 is open.
reem@reem-VirtualBox:~/IT361/Scripts$
```

CONCLUSION

In conclusion, the scripts presented in this report show the actual use of scripting in different system administration tasks. The first script creates encrypted copies of the main partition and sends them to the specified email address, automating the vital backup process. From the authentication log file, the second script effectively extracts and logs user-related activity. The third script finds and lists the top 5 memory-intensive processes connected to a certain user in order to prioritize system resource management. Finally, the fourth script improves the system's firewall settings by assisting with network security by verifying if a certain port is open or closed. Together, these scripts demonstrate how automation and modification can improve system functionality and make administrative duties simpler.

REFERENCES

[1] "File Encryption in Bash," in FastSitePHP, [Online]. Available: https://www.fastsitephp.com/en/documents/file-encryption-bash. [Accessed: November 3, 2023].
[2] "An Introduction to Cron Jobs," in Computer Hope, [Online]. Available: https://www.computerhope.com/unix/ucrontab.htm. [Accessed: November 3, 2023].
[3] "How to Use Cron to Automate Tasks on a VPS," in DigitalOcean, [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-on-a-vps. [Accessed: November 3, 2023].
[4] "GnuPG - The GNU Privacy Guard," in GnuPG, [Online]. Available: https://www.gnupg.org/gph/en/manual/x110.html. [Accessed: November 3, 2023].
[5] "How to Use GPG to Encrypt and Sign Messages," in DigitalOcean, [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-use-gpg-to-encrypt-and-sign-messages. [Accessed: November 3, 2023].
[6]“AWK - Quick Guide.” [Online]. Available: https://www.tutorialspoint.com/awk/awk_quick_guide.htm
[7]“ps command in Linux with Examples,” GeeksforGeeks, Jan. 10, 2022. [Online]. Available: https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/
[8]“Security - Firewall | Ubuntu,” Ubuntu. [Online]. Available: https://ubuntu.com/server/docs/security-firewall
[9]S. Verma, “What does -z flag mean in shell script used in if condition,” Linux.org, May 26, 2018. [Online]. Available: https://www.linux.org/threads/what-does-z-flag-mean-in-shell-script-used-in-if-condition.17899/#:~:text=The%20%2Dz%20flag%20causes%20test,the%20value%20returned%20by%20test.
[10]B. Boucheron, “How To Set Up a Firewall with UFW on Ubuntu 20.04,” DigitalOcean, May 04, 2020. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-20-04