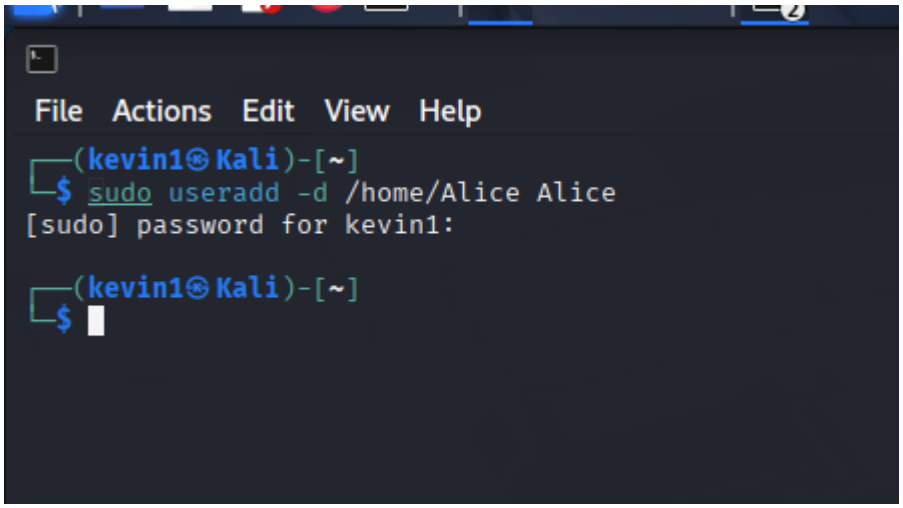
CYSE 270: Linux System for Cybersecurity

Assignment-9

Task A - Backup your system (Using crontab) [100 points]

Scenario: Performing system backup can be time-consuming, and the process is often overlooked. For this scenario:

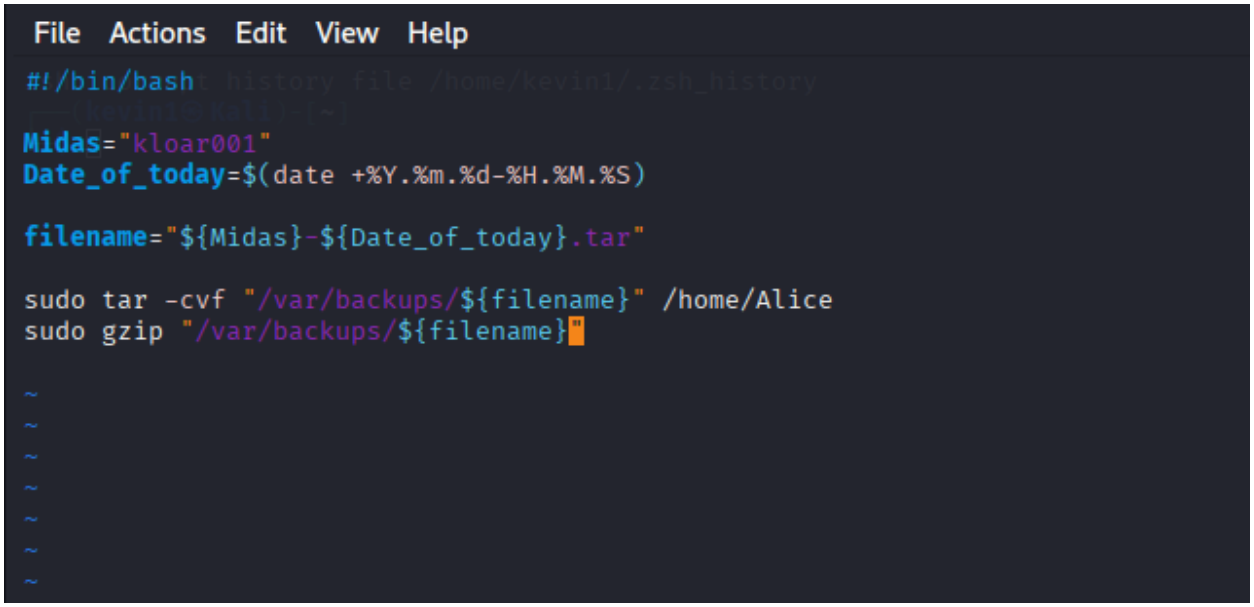
1. **(10 Points)** Create a new user **Alice (with home directory**).



As root, the command "useradd" with "-d" creates the new user with the home directory.

2. **(50 Points) Write a shell script that backups Alice's home directory by creating a tar file (tape archive), using the following steps:**
   - **a.** Do the following:
     - Take **2 inputs** with their values- your **MIDAS** name and **current date (for example, midas=Mohammed)**.
     - Create a variable named as **filename** that should be assigned the value as **MIDAS-date** (example output after executing the script would be like, **Mohammed-2024.11.04-22.08.01.tar.gz**).
     - Using **tar** command, create a tape archive for Alice's home directory (/home/Alice) and the **filename** created above (in step-2-ii). (Please learn about tar command in Linux for its usage)
   - **b.** Move the tape archive file/tar file (created in step 2-iii) to /var/backups/ directory using correct command in linux.

c. To optimize the disk usage, pick a compression algorithm (bz2, gzip, or xv) to compress the tar file you created in /var/backups/ in the previous step-2b.



The first part defines the variables by using “Midas” & “Date_of_today”. For the date variable, I made sure to use the correct format to display the order of the date correctly. Next, the variable “filename” is created, and I used the previous variables to create the tar file. As root user, the “tar” will merge files into a single archive file, and the cvf will create archive, verbose, and archive file name. At the final line, the filename is then zipped using gzip in the /var/backups/.

3. **(30 Points) Create a crontab file to keep the scheduled task running for 3 minutes**, then check the contents in the /var/backups directory. Your output should be look similar to the following:

```
┌──(kevin1㉿Kali)-[~]
└─$ sudo crontab -e
no crontab for root - using an empty one
Select an editor.  To change later, run select-editor again.
  1. /bin/nano        <──── easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny

Choose 1-3 [1]: 2
crontab: installing new crontab

┌──(kevin1㉿Kali)-[~]
└─$
```

```
*/3 * * * * /home/kevin1/backup_Alice.sh
~
~
~
~
~
~
```

Using crontab, I set the correct format to set the scheduled task to run for 3 minutes.

4. **(10 Points)** Cancel the crontab jobs.

```
#*/3 * * * * /home/kevin1/backup_Alice.sh
~
~
```

To cancel it, I went back into the file and added the # to end the task.

TASK B: SYSTEM CLEANUP (EXTRA CREDIT) [20 Points]

Scenario: In the above scenario, your system disk will be filled up eventually without cleaning up the old backups. Therefore, in this optional task, create a script that checks the number of backups you created in Task A. If the number of the backup file is more than a pre-defined threshold, the script will delete the old archives to maintain the backups under a reasonable size.

This script should do the following:

1. Count the number of backups created in Task A and determine if this number is larger than 3.

2. Nothing should happen if the number of backups is less than the threshold, 3.

3. If more backup archives are detected, calculate the number of backups to delete. Then delete the old archives.

```
#!/bin/bash

threshold=3

count=$(ls /var/backups/*.tar.gz | wc -l)

if [ "$count" -gt "$threshold" ]; then
        echo "Number of backups exceeds threshold ($threshold). Cleaning up..."

        delete=$((count - threshold))

        ls -t /var/backups/*.tar.gz | tail -n  $delete | xargs sudo rm -f
        echo "Deleted $delete old backups."
else
        echo "Number of backups is under the threshold ($threshold)."
fi
~
~
~
```

First, I set the variables for both threshold and count. The threshold is set to 3. Then for the count, it reads in the /var/backups for the repeated tar.gz file. Then, with the if command, it detects whether the count variable is higher than the threshold. With the delete variable added, if it detects that it is higher than the threshold, it will delete the exceeded count (which would be in the 12th line).

```
┌──(kevin1㉿Kali)-[~]
└─$ sudo ./backup_cleaner.sh
Number of backups is under the threshold (3).

┌──(kevin1㉿Kali)-[~]
└─$
```

```
┌──(kevin1㉿Kali)-[~]
└─$ ls /var/backups
alternatives.tar.0      dpkg.arch.0     dpkg.diversions.0     dpkg.statoverride.0     dpkg.status.0     kloar001-2025.11.04-19.09.01.tar.gz  kloar001-2025.11.04-19.15.01.tar.gz
apt.extended_states.0   dpkg.arch.1.gz  dpkg.diversions.1.gz  dpkg.statoverride.1.gz  dpkg.status.1.gz  kloar001-2025.11.04-19.12.01.tar.gz  kloar001-2025.11.04-19.24.01.tar.gz

┌──(kevin1㉿Kali)-[~]
└─$ sudo ./backup_cleaner.sh
Number of backups exceeds threshold (3). Cleaning up...
Deleted 1 old backups.

┌──(kevin1㉿Kali)-[~]
└─$
```

Note: As the script needs to write contents in the "/var/backups" folder, which is owned by root, you should consider the permission issue properly. (Using **sudo** to create crontab file)

Reference: How to Format Date for Display or Use In a Shell Script: https://www.cyberciti.biz/faq/linux-unix-formatting-dates-for-display/ Reference: How to append date timestamp to filename: https://crunchify.com/shell-script-append-timestamp-to-file-name/

Reflection

After completing the lab, I understood how the use of crontab could be useful in setting up my scripts to perform at a set time. There were challenges when completing this lab such as not correctly setting the variables in the backup_Alice.sh file.