Linux Assignment

Atypon / wiley

Hadeel Abunassar

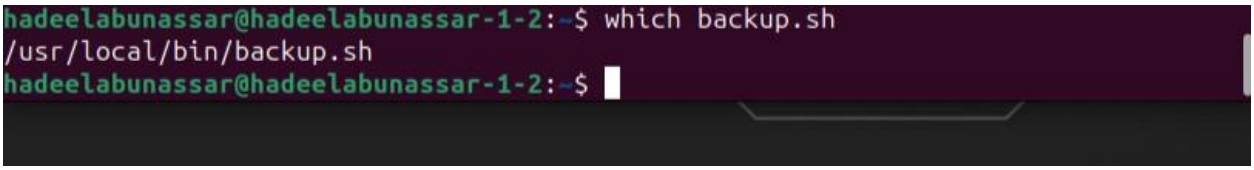
1- Backup Script

The purpose of this script focuses mainly to preform a backup for the user specified directories, the user will pass the source directories and destination directory as an argument when he invokes the executable backup.sh file.

At first, I'm going to ask the user if he wants this backup to be one time, scheduled. or if the user wants to list all previous scheduled backup and ask the user if he wants to delete one.

Let's do a simple demo!

First , I moved the backup to the bin folder , since bin is part of PATH where all Linux command executable files exists , so now, our script is executable form anywhere!



Now , let's run our script with considering the case the user wants it to be a one time backup , I listed dir4 -destination- before and after to see the backup results.

```
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ cd dir4
hadeelabunassar@hadeelabunassar-1-2:~/Desktop/dir4$ ls
hadeelabunassar@hadeelabunassar-1-2:~/Desktop/dir4$ cd ../
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ backup.sh dir1 dir2 dir4
Do you want to perform a one-time backup, schedule it continuously (daily, weekl
y, monthly), list and delete scheduled backups? (0- one-time  1- continuous  2-
list and delete) 0
Backup completed successfully.
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ cd dir4
hadeelabunassar@hadeelabunassar-1-2:~/Desktop/dir4$ ls
backup_2024-01-17           snapshot_dir1.txt
backup_log_2024-01-17.txt   snapshot_dir2.txt
hadeelabunassar@hadeelabunassar-1-2:~/Desktop/dir4$
```

Now, trying the scheduled backup, and then execute the crontab -l to see the result of the backup.(crontab will show the scheduled jobs in the system)

```
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ backup.sh dir1 dir2 dir4
Do you want to perform a one-time backup, schedule it continuously (daily, weekly, monthly), list and delete scheduled
backups? (0- one-time  1- continuous  2- list and delete) 1
You want the backup to be scheduled daily, weekly, or monthly: daily
Backup scheduled daily.
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ crontab -l
0 0 * * * backup.sh "dir1" "dir3"
0 0 * * * backup.sh "dir1 dir2" "dir3"
0 0 * * 0 backup.sh dir1 dir2
0 0 * * 0 backup.sh dir1 dir2 dir3
0 0 * * * backup.sh dir1 dir2 dir3
0 0 * * * backup.sh dir1 dir2 dir3
0 0 * * * backup.sh dir1 dir2 dir4
```

Finally , lets try to list and delete any scheduled backup and check the crontab

```
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ backup.sh dir1 dir2 dir4
Do you want to perform a one-time backup, schedule it continuously (daily, weekly, monthly), list and delete scheduled
backups? (0- one-time  1- continuous  2- list and delete) 2
List of scheduled backups:
     1  0 0 * * * backup.sh "dir1" "dir3"
     2  0 0 * * * backup.sh "dir1 dir2" "dir3"
     3  0 0 * * 0 backup.sh dir1 dir2
     4  0 0 * * 0 backup.sh dir1 dir2 dir3
     5  0 0 * * * backup.sh dir1 dir2 dir3
     6  0 0 * * * backup.sh dir1 dir2 dir3
     7  0 0 * * * backup.sh dir1 dir2 dir4
Do you want to delete a scheduled backup? (y/n): y
Enter the line number of the scheduled backup you want to delete: 7
Scheduled backup deleted.
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$ crontab -l
0 0 * * * backup.sh "dir1" "dir3"
0 0 * * * backup.sh "dir1 dir2" "dir3"
0 0 * * 0 backup.sh dir1 dir2
0 0 * * 0 backup.sh dir1 dir2 dir3
0 0 * * * backup.sh dir1 dir2 dir3
0 0 * * * backup.sh dir1 dir2 dir3
hadeelabunassar@hadeelabunassar-1-2:~/Desktop$
```

Process:

-When choosing one time backup a ‘perform backup’ function will be invoked.

- When choosing the continuous backup and the deciding the period, when it’s the time of the script to be executed, at first Linux will check if this is a cron process, because I don’t want the script to be interactive, I just want to perform the backup. This is done in this piece of code :

```
if [[ -z "$SHELL" ]]; then
   perform_backup
```

where , it checks if the $SHELL environment variable is empty, which is generally an indication that the script is not being run from an interactive shell session. – invoked form the crontab-

This piece of code is before the else, where the user will be asked if he wants the backup to be one time / continuous. etc.

- When choosing the list and delete option, I will invoke a function to list the crontab jobs, then in this function a delete function will be invoked if the user enters [y/Y]

Here I used cat -n to print the number of the lines of the output – beside each job-
Then I used sed -e "${line_number}d" to delete the job if the user wants to

About the script:

1- Error Handling:
   - Checking if all command used in the script is installed in the user system, Where the code have a for loop iterates over the commands , then check the existence of each command using command -v and redirect the output to dev/null because we are not interested in this.
   - Checking the existence of the source and destination directory
   - Checking the success of the compression process
   - Checking the success of retrieving the backup size

2- Compression:
   The tar command Is used to perform the compression
   - A new archive will be created
   - It should be a compressed using gzip,where --gzip flag instructs tar to compress the backup files using the potent Gzip algorithm.
   - The created file will be depended on: 1-source name 2-the time of the backup
   - The compression is incremental, to save the resources , this is done by --listed-incremental option , and a snapshot file for each directory that will be stored in the destination directory to help in the incremental backup process.
   - Before performing the previous operations, we will be moved to the destination directory to save the changes there, the -C option will make linux temporarily switch its working directory to the specified path before executing its archiving tasks.
   - Finally , the name of the directory being archived is taken form the basename of the source directory.

3- Logging
   I have a log function that will take as a parameter the log massage from different places in the script , all errors that are mentioned previously in the Error Handling section are being logged.

2- System Health Script:

This script purpose is to show general system health information , it includes a function for each check , all function are listed below.

```
echo "                    System Info"
 echo "-------------------------------------------------------"
get_system_info
echo "-------------------------------------------------------"
echo "                    Network Connectivity"
 echo "-------------------------------------------------------"
CheckNetworkConnectivity
echo "-------------------------------------------------------"
echo "                    processes"
 echo "-------------------------------------------------------"
get_process_wait_time
top3proc
echo "-------------------------------------------------------"
echo "                    Memory"
 echo "-------------------------------------------------------"
check_memory_usage
check_virtual_memory_usage
check_swap_usage
echo "-------------------------------------------------------"
echo "                  Disk Space - File System"
 echo "-------------------------------------------------------"
CheckDiskSpace
echo "-------------------------------------------------------"
echo "                    System updates"
 echo "-------------------------------------------------------"
show_system_update_info
```

1 – get_system_info()

- Print host name using hostname command
- Retrieve kernel version using uname -r , uname Is a command to get system info and the - r option is to get the kernel version
- Last reboot time using who -b

2 - CheckNetworkConnectivity()

By checking the [ip route get 8.8.8.8] command , It will return 0 if the packets in the network reach 8.8.8.8 – which is a commonly used test IP address –.

3- get_process_wait_time()

1- We will get the average number of processes in the run queue over the last minute from the loadavg file
2- Retrieve number of process form cpuinfo file by grep ^processor that represent each processor.
3- Dividing avg waiting time by #No of CPUs to get the waiting time.

4 - top3proc()

This function depends mainly in:

1- ps aux: This command acts like a system detective, scanning all running processes on your Linux machine and collecting details about them.
2- --sort -rss and --sort -pcpu: These options guide the detective's focus. They instruct it to sort the processes based on their memory usage (RSS) or CPU usage (%CPU), respectively.

Then I used the 'NR>1 && NR<=4 To filter the top 3 processes.

5,6,7-

check_memory_usage()

check_virtual_memory_usage()

check_swap_usage()

to check the Ram, virtual memory, and swap memory of the system.

In all three functions I am going to grep the free memory and the total memory form meminfo file, grep Mem for RAM ,Vm for the virtual and swap for swap.

Then I calculate the memory usage of all three of them by (total-free)/total *100%.

8- CheckDiskSpace()

By iterating through df -h command to get the free space in the file system and print it in user friendly way.

9- show_system_update_info()

1. Retrieving Last Update Time:

stat -c %y /var/lib/apt/lists/*: This to extract the last modification time (%y) of each file within the /var/lib/apt/lists/ directory, which includes package lists essential for tracking available software , then the output will be sort by the timestamp , and we will get the tail to get the last recent update

2. Simulating Upgrade for Available Updates:

apt-get upgrade: This command invokes the apt-get package manager where it will analyzes the system's installed packages and available updates, generating a comprehensive list of potential upgrades without executing any actual installation actions.

grep -oP '\d+ upgraded' | head -n 1: grep to get the line containing the number of upgradable packages. The -oP options ensure the extraction of the numerical value representing the count of eligible upgrades. The head -n 1 command extracts only the first line, because multiple lines might contain the "upgraded" keyword.