Linux Command Mastery and Scripting Project

Edrees Nabeel Ashab

Table of Contents

This is a report regarding the Linux Command Mastery and Scripting Project assignment. This report has three sections: **The purpose and functionality**, this section explains the purpose of each function of the scripts, and how it works. **Performance analysis**. **Optimizations and their impact**, in order to speed up the scripts processes I had to do some optimizations, I will talk about these optimizations and their impact when using these scripts.

1. The purpose and functionality:

Backup:

a. **validate_directories()**: this function checks the provided directories that the user wants to backup to make sure that they exist using conditional statements "if [[! -d $directory]]", and also if the user doesn't enter any directories at all "if [[-z $directories]]", this functions keeps asking him for a valid input.

```
root@b99c09eeff95:/# ./backup.sh
Enter directories: ooo
Please enter valid directories.
Enter directories: lib usr
```

b. **validate_destination()**: same as **validate_directories()**, but here it checks with the backup destination, if none provided the default destination is "/home/backups."

c. **validate_restore_file()**: validates the provided restore file, which is the backed-up files, and keeps prompting the user to enter a valid backup file.

d. **check_status()**: checks if last used command was successful or not "if [[$? -eq 0]]."

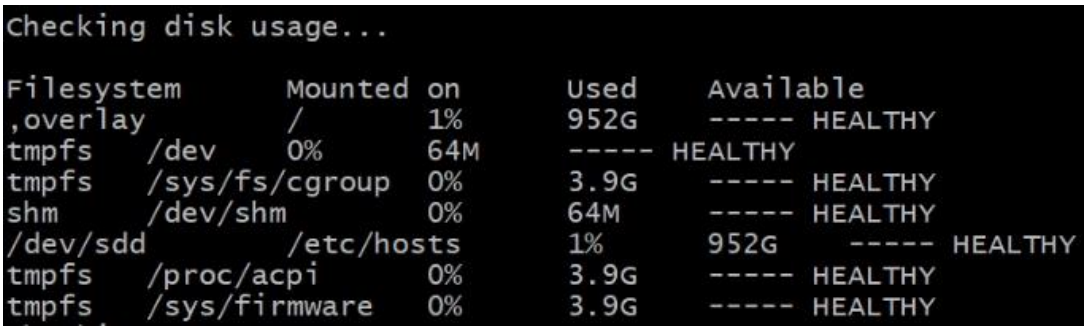
e. **print_log()**: prints log to "backup_log.txt" file, it includes various information such as backup file size, what directories it includes, backup status, etc.

f. **perfom_backup()**: this is where the actual backup happens, it simply takes the provided directories and archives them into one file using "tar -cvf …" command, then compress that file using "pigz …" command, and encrypt it with a password if the user wants "openssl …", then move it to the destination.

g. **perfom_restore()**: restores a backup file by decrypting the file if it was encrypted and the user entered the correct password, and decompress it, then move it to "/home/restore_backup" directory.

System health check:

a. **check_disk()**: checks disk usage using “df -h,” and reports the status of the disks whether they’re “HEALTHY” if its usage is less than %85, and “WARNING” if less than %95 and greater than %85, and “CRITICAL” greater than %95.



b. **check_memory()**: checks memory usage “free -h” and report if any actions should be taken.
c. **check_running_services()**: lists the running services “systemctl … “.
d. **check_recent_sytem_updates()**: prints the most recent system updates from the history.log file in “/var/log/apt/” directory, and checks if there are any upgradable software.

2. Performance analysis:

Backup:

In order to make this script user-friendly and easy to use, I've added a usage message to let the user know how to use it, and also provide useful comments like when the user doesn't provide directories at all or if he provides invalid directories, same goes for restore backup file it prints and prompts the user enter again. This way the user has an interactive script that he can use without knowing too many details or experiencing any unwanted behavior.

```
This is a script to do backups for user-specified directories.
It performs compression, encryption if needed, backup restoration, and error handling.
The backup status will be reported and saved in 'backup_log.txt'.

Usage: ./backup.sh [-r|--restore <backup_file>] [-e|--encrypt] [-d|--directories <directory_list>] [-D|--destination <backup_destination>] [-c|--c
omments <backup_comments>] [-h|--help]
Options:
  -r, --restore <backup_file>   Restore from the specified backup file.
  -e, --encrypt                 Enable encryption for the backup.
  -d, --directories <directory_list>
                                Specify a space-separated list of directories to backup.
  -D, --destination <backup_destination>
                                Specify the destination for the backup.
  -c, --comments <backup_comments>
                                Add comments about the backup.
  -h, --help                    Display this help message.
```

System health check:

In this script as well I added the usage message, and made it as simple as possible, the user just have to provide the flag for the wanted check, and it will generate the report and print it as well.

Report example:

```
-----------------------------------

*****************************
     System health check
*****************************
Current date & time: Wed Jan 17 19:07:38 UTC 2024

Hostname: b99c09eeff95
Uptime: up 7 hours, 17 minutes

Checking disk usage...

Filesystem      Mounted on      Used    Available
,overlay        /       1%      952G    ----- HEALTHY
tmpfs   /dev    0%      64M     ----- HEALTHY
tmpfs   /sys/fs/cgroup  0%      3.9G    ----- HEALTHY
shm     /dev/shm        0%      64M     ----- HEALTHY
/dev/sdd        /etc/hosts      1%      952G    ----- HEALTHY
tmpfs   /proc/acpi      0%      3.9G    ----- HEALTHY
tmpfs   /sys/firmware   0%      3.9G    ----- HEALTHY
Checking memory usage...
               total        used        free      shared  buff/cache   available
Mem:           7.7Gi       967Mi       4.7Gi       6.0Mi       2.0Gi       6.5Gi
Swap:          2.0Gi          0B       2.0Gi

Checking running services...
apache2.service loaded active running   The Apache HTTP Server

1 loaded units listed.

Checking recent system updates...
Start-Date: 2024-01-14  10:51:21
Commandline: apt-get upgrade
Upgrade: libc6:amd64 (2.35-0ubuntu3.5, 2.35-0ubuntu3.6), libsqlite3-0:amd64 (3.37.2-2ubuntu0.1, 3.37.2-2ubuntu0.3), libc-bin:amd64 (2.35-0ubuntu3.
5, 2.35-0ubuntu3.6)
End-Date: 2024-01-14  10:51:26
```

3. Optimizations and their impact:

perform_backup(): I've decided to use (pigz) to compress the backup files, because it's faster. It's a multi-threaded version of (gzip).
When using (gzip) for a 208MB archived directories it took about 8 seconds to compress, and with (pigz) the time it takes to backup the same directories takes 1.4s and the compressed size is almost same.

I've also used pipes which can be more efficient in terms of time and resource utilization compared to writing intermediate files to disk, because pipes allow the output of one command to be used as the input for another command without storing the data on disk, which can reduce disk I/O and save storage space.

I utilized the "awk" command in my scritps to process strings, it is designed to handle large datasets efficiently. It doesn't load the entire file into memory at once, which is particularly useful for processing large log files or other large text datasets.

```
usage="$(echo "$line" | awk '{ print $5 }' | sed 's/%//')"
disk="$(echo "$line" | awk '{ print $1 $6 }' | sed 's/%//')"
report+="$(echo "$line" | awk '{print $1"\t"$6"\t"$5"\t"$4}')"
```