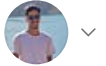


Open in app ↗

Get unlimited access



Search Medium



Fayssal Elaazouzi

Mar 22 · 5 min read · Listen



Save



# Day 5 : Advanced Linux Shell Scripting with User management👍

This is #90DaysOfDevops challenge under the guidance of Shubham Londhe sir.

Day 5 TASK

check this for task:

## 90DaysOfDevOps/tasks.md at master · LondheShubham153/90DaysOfDevOps

This repository is a Challenge for the DevOps Community to get stronger in DevOps. This challenge starts on the 1st...

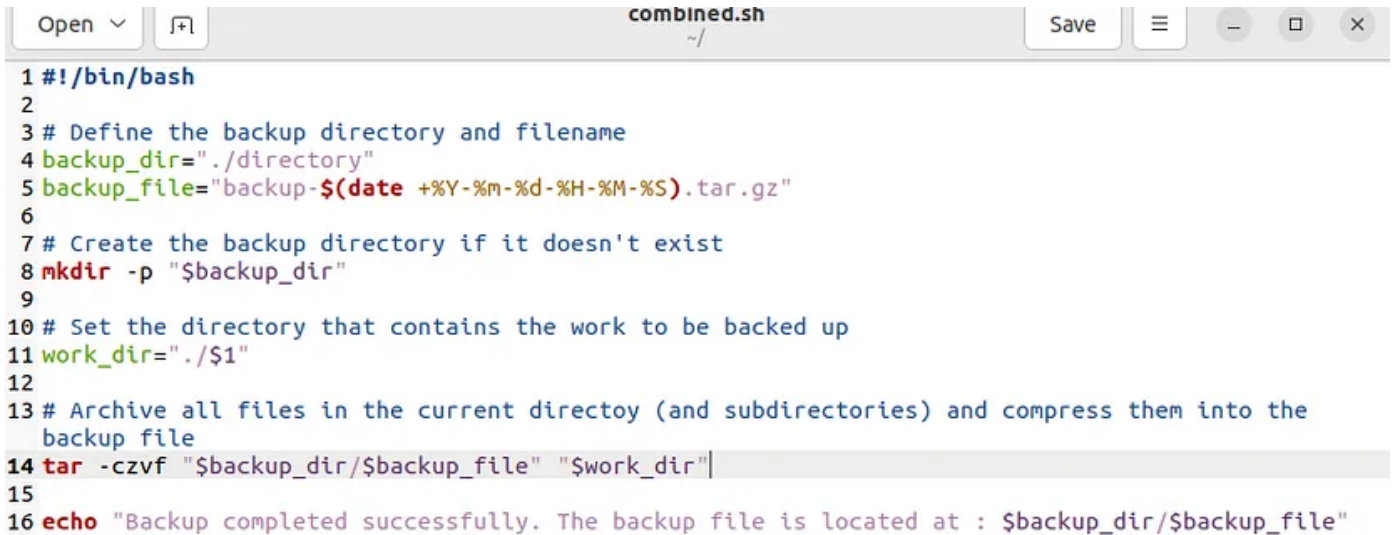
github.com

**1) Write a bash script createDirectoryess1.sh that when the script is executed with three given arguments (one is the directory name and second is the start number of directories and the third is the end number of directories ) it creates a specified number of directories with a dynamic directory name.**

```
Open  [icon] *combined.sh ~/ Save [icon] [icon] [icon]
1 #!/bin/bash
2 mkdir $1
3 for ((i=$2;i<=$3;i++))
4 do
5     mkdir ./$1/$1$i
6 done
7
```

```
fayssal@fayssal-VirtualBox:~$ ./combined.sh day 1 3
fayssal@fayssal-VirtualBox:~$ ls
combined.sh  D1  D2  D3  day  Desktop  Documents  Downloads  Music  my_directory  my_dir
fayssal@fayssal-VirtualBox:~$ ls day
day1  day2  day3
fayssal@fayssal-VirtualBox:~$
```

## 2) Create a Script to back up all your work done till now.



```
combined.sh
~/
Open  Save  [Icons]
1 #!/bin/bash
2
3 # Define the backup directory and filename
4 backup_dir="./directory"
5 backup_file="backup-$(date +%Y-%m-%d-%H-%M-%S).tar.gz"
6
7 # Create the backup directory if it doesn't exist
8 mkdir -p "$backup_dir"
9
10 # Set the directory that contains the work to be backed up
11 work_dir="./$1"
12
13 # Archive all files in the current directory (and subdirectories) and compress them into the
    backup file
14 tar -czvf "$backup_dir/$backup_file" "$work_dir"
15
16 echo "Backup completed successfully. The backup file is located at : $backup_dir/$backup_file"
```

```
fayssal@fayssal-VirtualBox:~$ ./combined.sh day
./day/
./day/day2/
./day/day1/
./day/day3/
Backup completed successfully. The backup file is located at : ./directory/backup-2023-03-22-14-16-53.tar.gz
```

Here's a brief explanation of what the script does:

- The first line ( `#!/bin/bash` ) specifies the shell that should be used to run the script (in this case, bash).
- The `backup_dir` variable specifies the directory where the backup file should be stored.
- The `backup_file` variable specifies the name of the backup file, which includes the current date and time to make it unique.
- The `mkdir -p "$backup_dir"` command creates the backup directory if it doesn't already exist.

- The `tar -czvf "$backup_dir/$backup_file" .` command creates a tar archive of all files in the current directory (and subdirectories) and compresses them using gzip. The resulting archive is saved to the backup directory with the specified filename.
- The `echo` command displays a message to indicate that the backup is complete and specifies the location of the backup file.

### 3) Read About Cron and Crontab, to automate the Script

Cron is a time-based job scheduler in Unix-like operating systems. It allows you to schedule and automate repetitive tasks, such as running a backup script at a specific time and date. The Cron daemon runs in the background and executes scheduled tasks at the specified intervals.

Crontab is a file that contains the scheduling information for the Cron daemon. It specifies the commands or scripts that need to be executed at specific times. You can use the Crontab file to schedule a backup script to run automatically at a specified time.

Here are the basic steps to create a Cron job to automate a backup script:

#### 1. Open the Crontab file by running the following command:

```
crontab -e
```

This will open the Crontab file in your default text editor.

#### 2. Add a new line to the file with the following format:

```
* * * * * /path/to/backup_script.sh
```

The asterisks represent the time and date parameters for the job. They are as follows:

- Minute (0–59)
- Hour (0–23)
- Day of the month (1–31)
- Month (1–12)
- Day of the week (0–6, where Sunday is 0)

In the example above, the asterisks indicate that the job will run every minute of every hour of every day of every month of every day of the week.

The `/path/to/backup_script.sh` is the command or script that you want to run.

### 3. Save and close the Crontab file.

Once you've set up the Cron job, the backup script will run automatically at the specified intervals. You can modify the time and date parameters as needed to customize the schedule.

Note that Cron jobs run as the user who created them. Make sure that the user has the necessary permissions to execute the backup script and access any required files or directories.

## 4) Read about User Management

User management in Linux is an essential task that system administrators need to perform to ensure that users can access the resources they need while maintaining system security. In this context, “user” refers to any individual who needs to access the system, including administrators, developers, and end-users.

Here are some of the key aspects of user management in Linux:

1. **User accounts:** A user account is a unique identity assigned to each user that needs to access the system. User accounts store user-specific information such as username, password, user ID (**UID**), and group ID (**GID**). User accounts can be

created, modified, and deleted using various tools such as the *useradd*, *usermod*, and *userdel* commands.

2. **Passwords:** Passwords are used to authenticate users and ensure that only authorized users can access the system. Password policies can be enforced using tools such as the *pam\_pwquality* module, which can check password complexity, length, and age.
3. **Groups:** Groups are used to organize users and define their permissions. Each group has a unique group ID (GID) and can be assigned specific privileges and access rights. The *groupadd*, *groupmod*, and *groupdel* commands are used to manage groups.
4. **Permissions:** Permissions control the access rights of users and groups to files and directories on the system. The *chmod* command is used to *set file and directory permissions*, while the *chown* and *chgrp* commands are used to *change file and directory ownership*.
5. **Authentication:** Authentication is the process of *verifying the identity of users before granting them access to the system*. Linux uses various authentication mechanisms such as **passwords**, **public key authentication**, and **biometric authentication**.
6. **Access control:** Access control is the process of *restricting access to system resources based on user and group permissions*. Linux provides various tools for access control, such as Access Control Lists (ACLs) and SELinux.

Overall, user management is a critical aspect of Linux administration that helps to maintain system security and ensure that users have the necessary access to the resources they need.

## 5) Create 2 users and just display their Usernames

```
fayssal@fayssal-VirtualBox:~$ sudo useradd Xfayssal
[sudo] password for fayssal:
fayssal@fayssal-VirtualBox:~$ sudo useradd Yfayssal
fayssal@fayssal-VirtualBox:~$ cat /etc/passwd
```

If this post was helpful, please do follow and click the clap 🖐️ button below to show your support 😊

**\_Fayssal** 👍

<https://medium.com/@elaazouzifayssal/day-5-advanced-linux-shell-scripting-with-user-management-8257fa64512c>