Bash Scripting

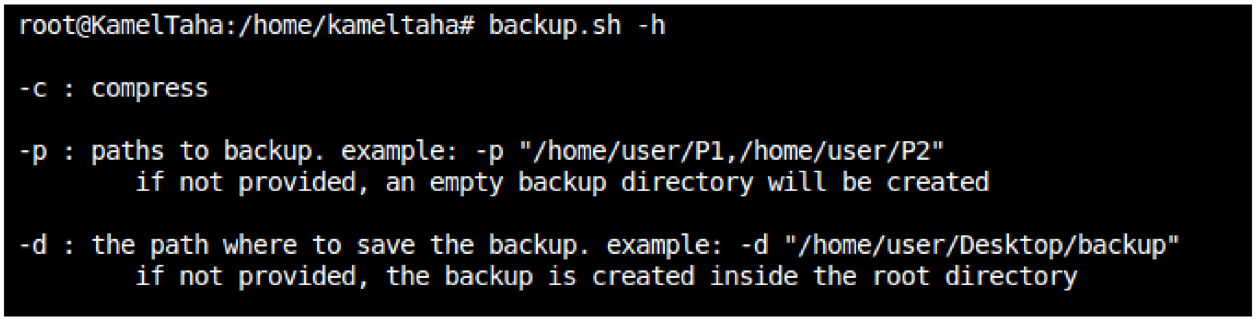
Bash is a very powerful scripting language. It is one out of many Linux scripting languages. It gives us the ability to automate tasks, commands and many more. Let’s say we need to write 50 commands every day at some o’clock, we can write them inside a bash script and schedule them at that time. Also, we can create some logic utilizing if statements and the different types of loops. In this report I will discuss two scripts, the first one is a backup script that takes the paths of the files to backup, and the destination where to save the backup, and gives you the option to compress the files to be backed-up. The second script is the health script, it displays system-related information such as storage usage, memory usage, running services and recent system updates.

Backup script

The purpose of this script is to create a copy of user-specified files, this is useful when you have either a server or a personal desktop. If you have a server then this might be a critical mission that must be done on a daily basis. For that reason, I took server scheduled backups into consideration when I created the script. I did that by allowing the user to pass the paths to backup, the destination, and the compress option, as parameters with the command itself. Instead of asking the user for them after running the command, this helps a lot to automate the process of backup. Also, any questions after executing the command can be answered by yes or no, which can be automated by using the “yes” command before the backup command, separated by the pipeline operator.

Another nice thing about the script, which helps to make it work better with servers, is that it logs all errors to a log file saved in the logs directory under backup logs.

Options that can be passed with the command:



```
root@KamelTaha:/home/kameltaha# backup.sh -h

-c : compress

-p : paths to backup. example: -p "/home/user/P1,/home/user/P2"
        if not provided, an empty backup directory will be created

-d : the path where to save the backup. example: -d "/home/user/Desktop/backup"
        if not provided, the backup is created inside the root directory
```

- -h: to show the help section.
- -c: if provided, the backup will be compressed.
- -p: the paths to backup, comma separated inside a double quotation.
- -d: the destination path where to save the backup.

Notes:

- The -h parameter must be provided alone to work properly, otherwise an error will be thrown.
- If no destination is provided, the backup will be created at the root directory.
- If no paths are provided to backup, an empty backup will be created.

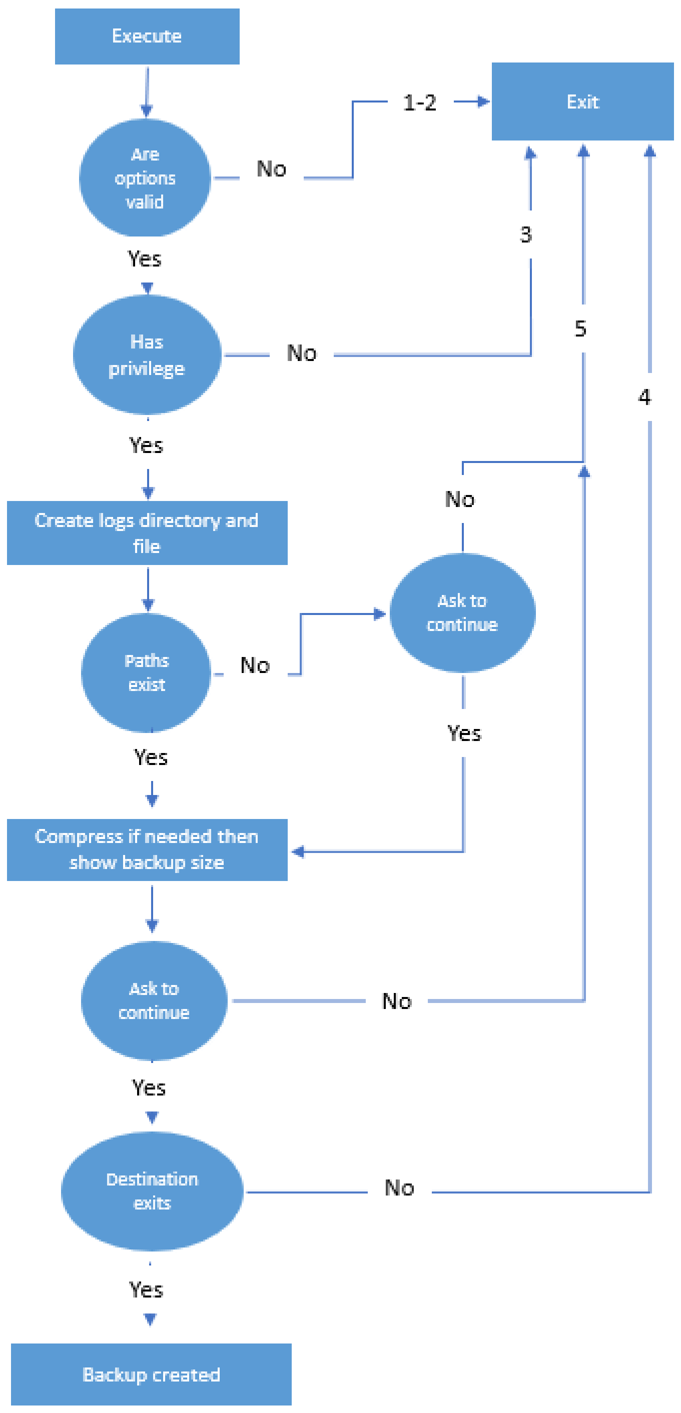
Exception handling:

- The script needs root privileges to work properly since it needs to create directories and create files or copy them to another directory. If no root privileges are available an error will be thrown.
- The -h must be provided alone as mentioned above.
- If an option is provided with the command that is not listed from the options above, an error is thrown.
- Throwing an error if the destination directory does not exist (another technique might be creating that directory).
- If the files size before compression is less than the available storage minus a threshold, an error is thrown.

Exit codes:

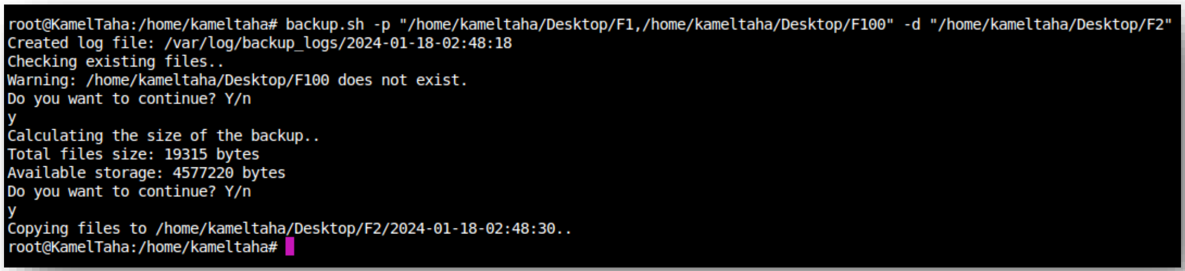
- 0: success.
- 1: provided an invalid option.
- 2: help was provided as an option along with other options.
- 3: the script requires root privileges.
- 4: destination was not found.
- 5: exited by user choosing not to continue.

The flowchart below explains the flow of the script:

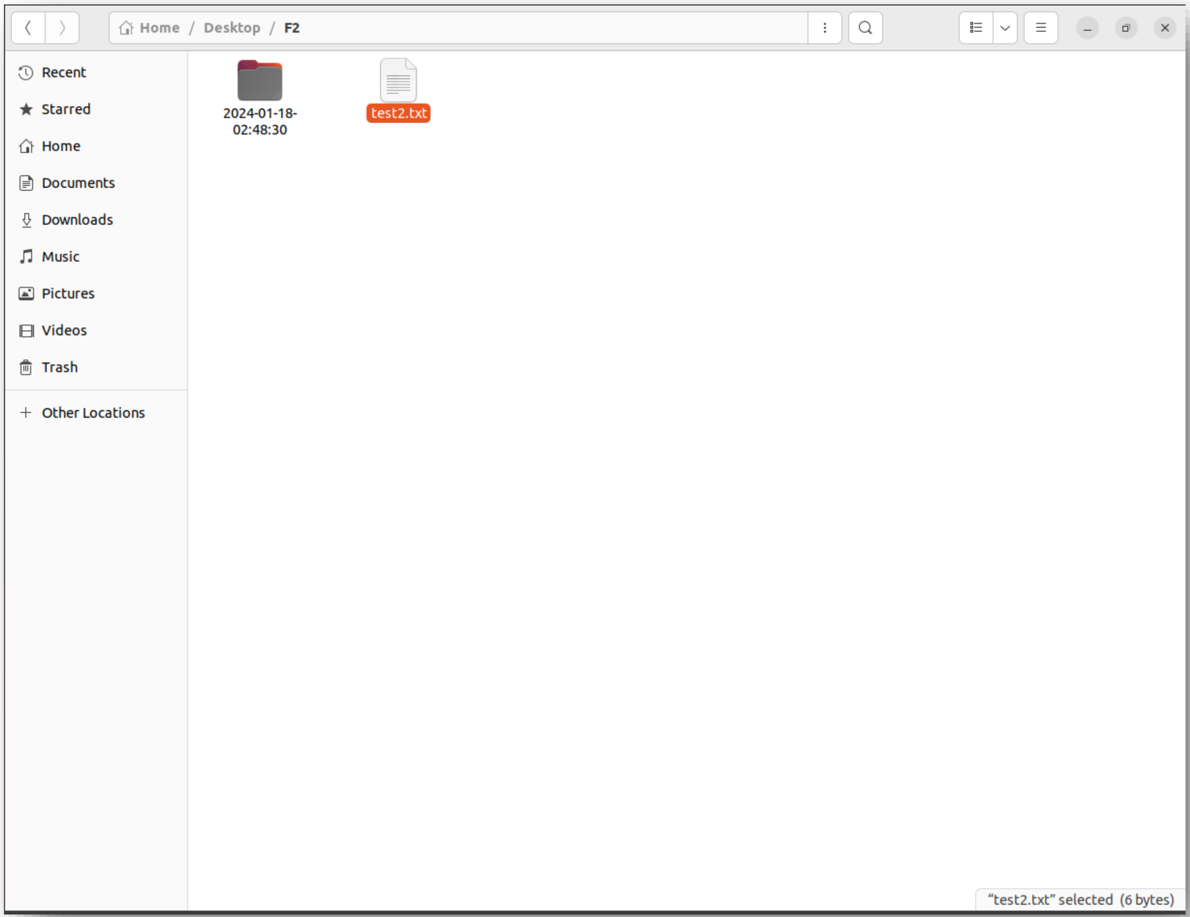


In the script, I tried to use modular functions as much as possible, but I have noticed that it is difficult to return values (other than exit codes) from a function, global variables might be used, but this is not good practice in my opinion since the variable may be modified from multiple places while it is intended to be generated only by the function itself.

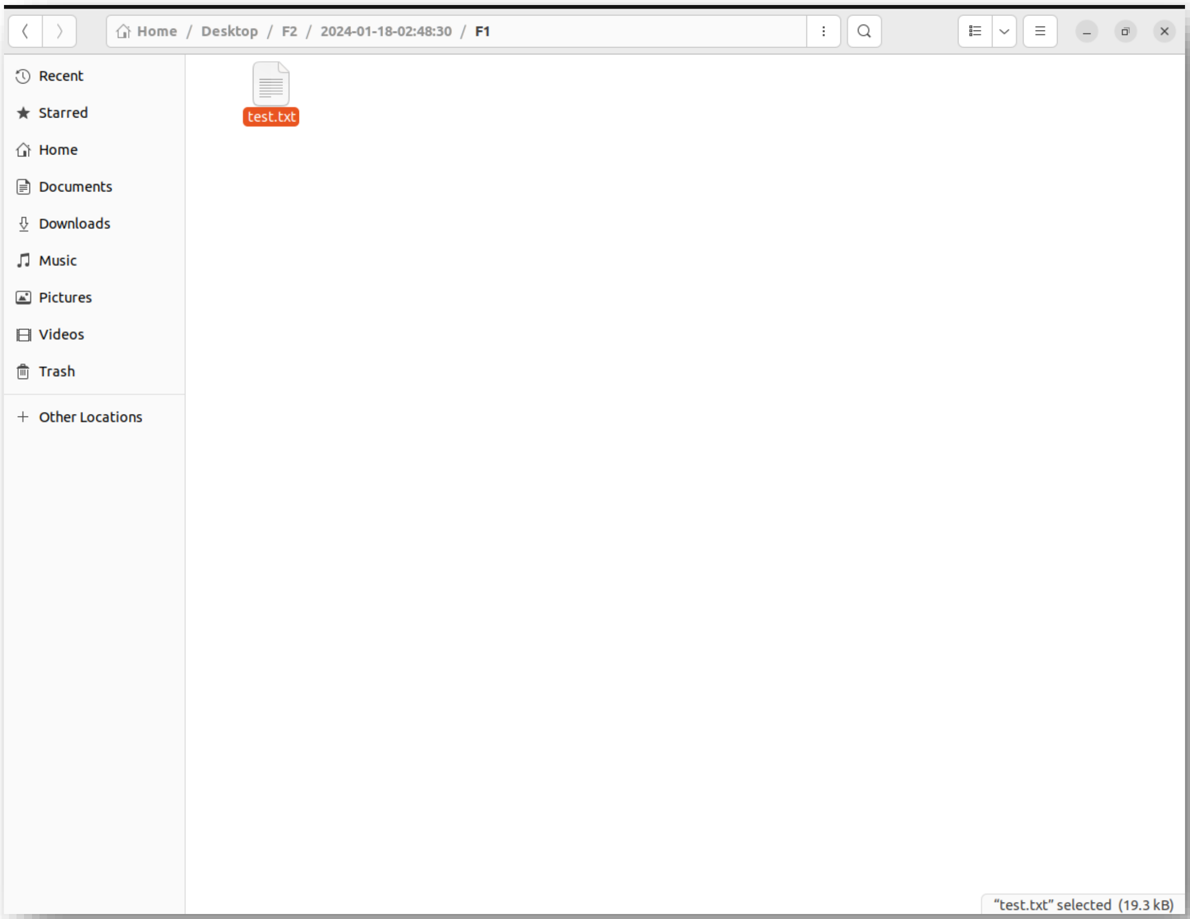
Sample runs:



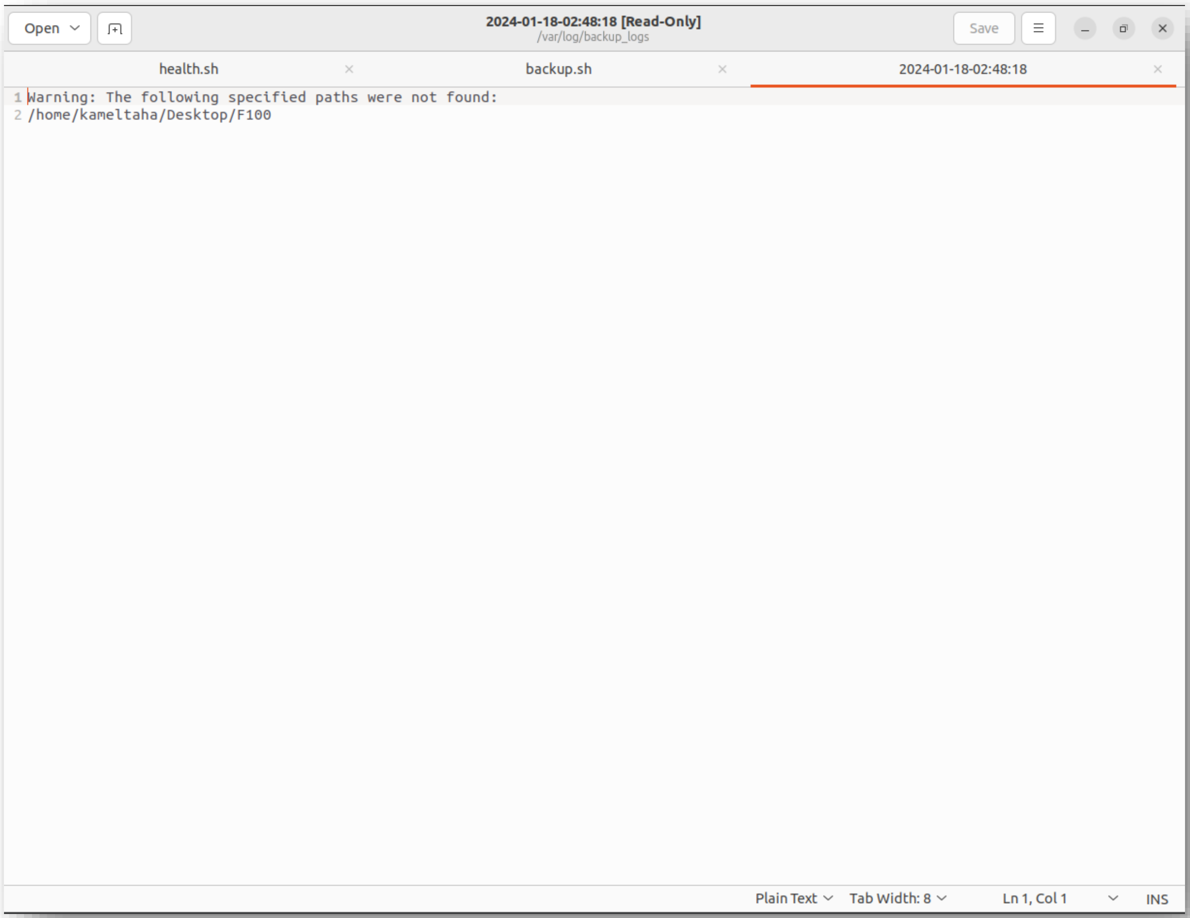
```
root@KamelTaha:/home/kameltaha# backup.sh -p "/home/kameltaha/Desktop/F1,/home/kameltaha/Desktop/F100" -d "/home/kameltaha/Desktop/F2"
Created log file: /var/log/backup_logs/2024-01-18-02:48:18
Checking existing files..
Warning: /home/kameltaha/Desktop/F100 does not exist.
Do you want to continue? Y/n
y
Calculating the size of the backup..
Total files size: 19315 bytes
Available storage: 4577220 bytes
Do you want to continue? Y/n
y
Copying files to /home/kameltaha/Desktop/F2/2024-01-18-02:48:30..
root@KamelTaha:/home/kameltaha#
```



Please note the path:

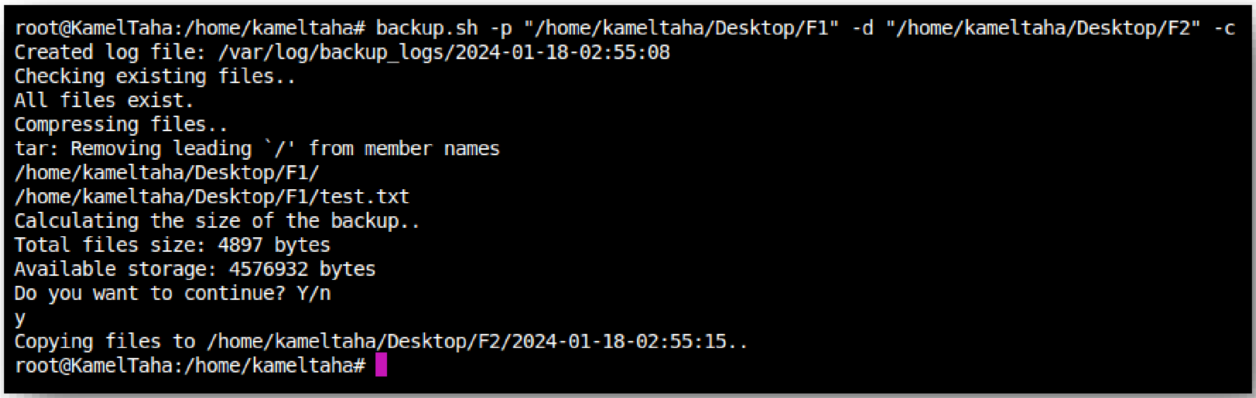


Log file:

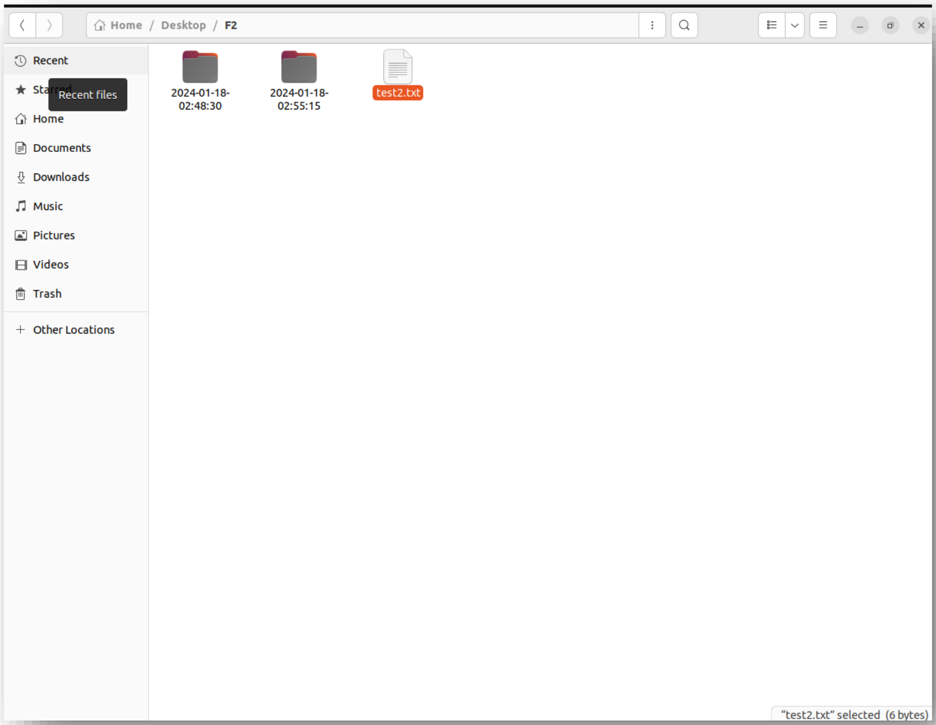


```
Warning: The following specified paths were not found:
/home/kameltaha/Desktop/F100
```

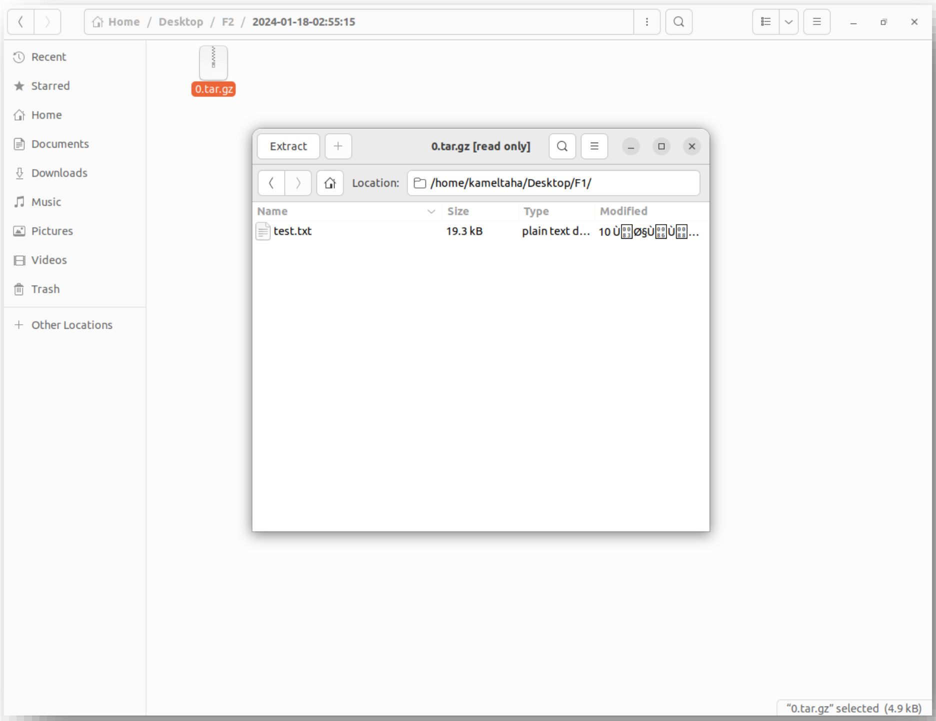
Now let’s try compression:



```
root@KamelTaha:/home/kameltaha# backup.sh -p "/home/kameltaha/Desktop/F1" -d "/home/kameltaha/Desktop/F2" -c
Created log file: /var/log/backup_logs/2024-01-18-02:55:08
Checking existing files..
All files exist.
Compressing files..
tar: Removing leading `/' from member names
/home/kameltaha/Desktop/F1/
/home/kameltaha/Desktop/F1/test.txt
Calculating the size of the backup..
Total files size: 4897 bytes
Available storage: 4576932 bytes
Do you want to continue? Y/n
y
Copying files to /home/kameltaha/Desktop/F2/2024-01-18-02:55:15..
root@KamelTaha:/home/kameltaha#
```



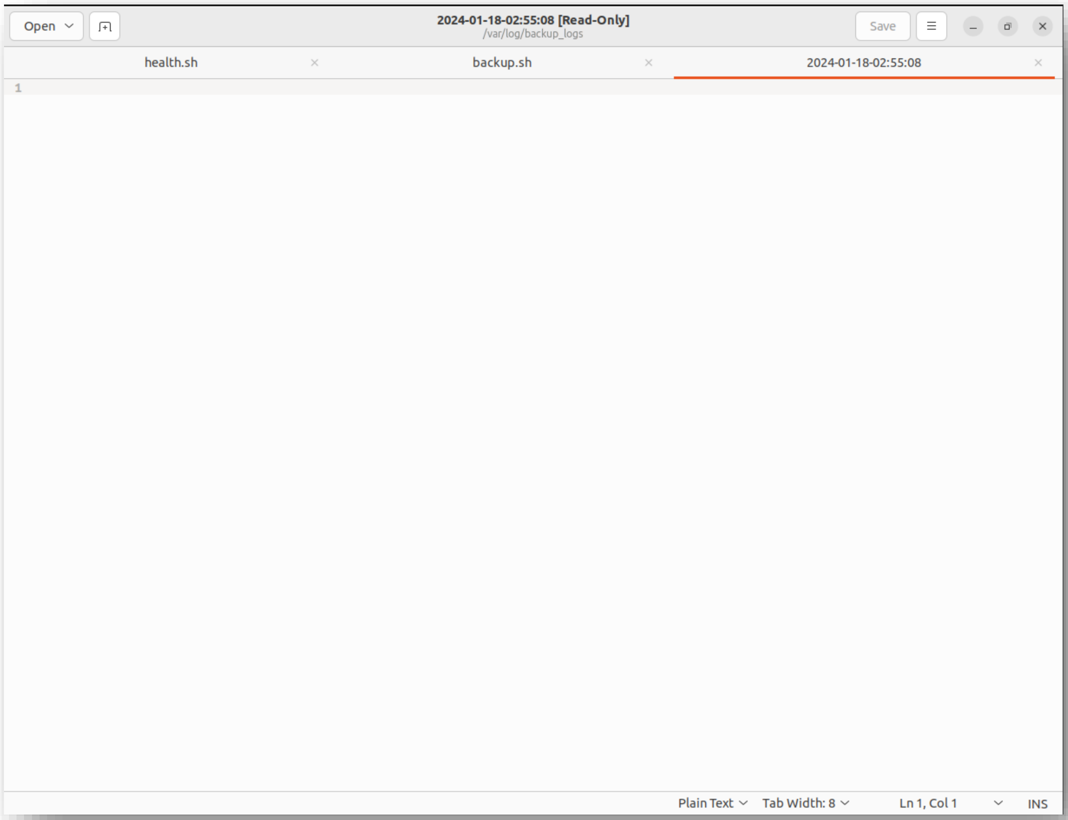
"test2.txt" selected (6 bytes)



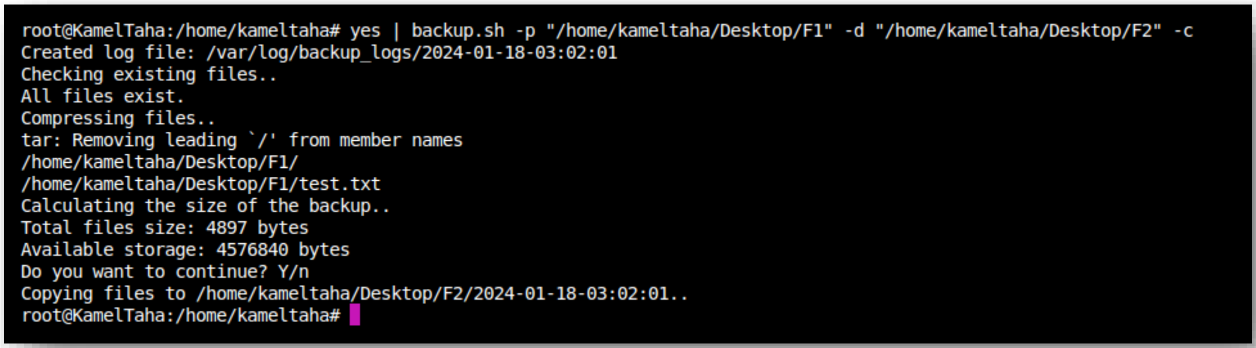
Name	Size	Type	Modified
test.txt	19.3 kB	plain text d…	10 Ù⍰øşÙ⍰Ù⍰…

"0.tar.gz" selected (4.9 kB)

And the log file is empty since there are no errors.



We can use the yes command to automate answering the yes or no questions:

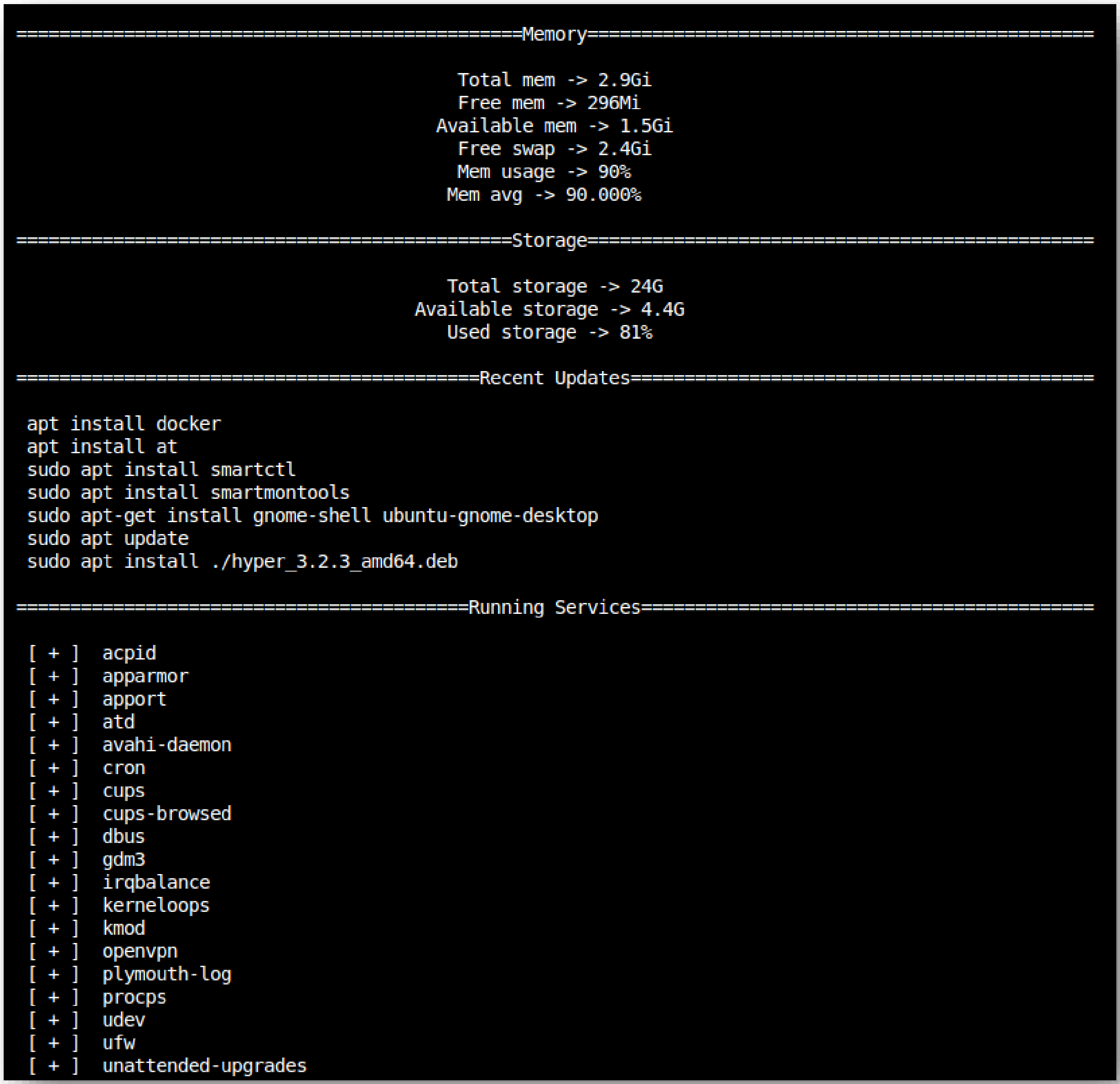


```
root@KamelTaha:/home/kameltaha# yes | backup.sh -p "/home/kameltaha/Desktop/F1" -d "/home/kameltaha/Desktop/F2" -c
Created log file: /var/log/backup_logs/2024-01-18-03:02:01
Checking existing files..
All files exist.
Compressing files..
tar: Removing leading `/' from member names
/home/kameltaha/Desktop/F1/
/home/kameltaha/Desktop/F1/test.txt
Calculating the size of the backup..
Total files size: 4897 bytes
Available storage: 4576840 bytes
Do you want to continue? Y/n
Copying files to /home/kameltaha/Desktop/F2/2024-01-18-03:02:01..
root@KamelTaha:/home/kameltaha# 
```

Health script

The purpose of this script is to display device-related information such as storage usage, memory usage, running services, and recent system updates. This script is a little bit different than the first one since it keeps running infinitely until we enter “CTRL + C”. It keeps updating the information mentioned above every 2 seconds.

The script also may introduce some recommendations based on storage or memory usage.



```
===============================================Memory==============================================

                                        Total mem -> 2.9Gi
                                        Free mem -> 296Mi
                                      Available mem -> 1.5Gi
                                        Free swap -> 2.4Gi
                                        Mem usage -> 90%
                                       Mem avg -> 90.000%

==============================================Storage==============================================

                                       Total storage -> 24G
                                    Available storage -> 4.4G
                                       Used storage -> 81%

===========================================Recent Updates===========================================

 apt install docker
 apt install at
 sudo apt install smartctl
 sudo apt install smartmontools
 sudo apt-get install gnome-shell ubuntu-gnome-desktop
 sudo apt update
 sudo apt install ./hyper_3.2.3_amd64.deb

==========================================Running Services==========================================

 [ + ]  acpid
 [ + ]  apparmor
 [ + ]  apport
 [ + ]  atd
 [ + ]  avahi-daemon
 [ + ]  cron
 [ + ]  cups
 [ + ]  cups-browsed
 [ + ]  dbus
 [ + ]  gdm3
 [ + ]  irqbalance
 [ + ]  kerneloops
 [ + ]  kmod
 [ + ]  openvpn
 [ + ]  plymouth-log
 [ + ]  procps
 [ + ]  udev
 [ + ]  ufw
 [ + ]  unattended-upgrades
```

Regarding memory, please note the following:

- Total mem: the total memory of the device.
- Free mem: the completely free memory on the device.
- Available mem: the memory that is not currently free but can be freed if needed.
- Free swap: swap is when the memory is full, we borrow some storage to act as the memory. So, this value represents how much we can borrow from the storage.
- Mem usage: the percentage of memory usage.
- Mem avg: the average of memory usage since the command started executing.

Regarding storage, please note the following:

- Total storage: the total storage of the device.
- Available mem: the free storage on the device.
- Used storage: the percentage of storage usage.

Regarding storage, please note that it prints the last 15 updates that occurred on the system. It works for both Ubuntu/Debian based systems, and Red Hat systems. Since it is done by checking for any “apt” or “yum” commands in the history file utilizing the grep command.

And at the end of the input, we can see all the running services on the system.

This assignment was useful, it showed me how powerful Linux is. I have searched for a lot of things and learnt a lot to be able to solve the assignment. While searching I empowered my knowledge and got more convinced about how much Linux is important, and how it can handle servers’ jobs in a very convenient way.