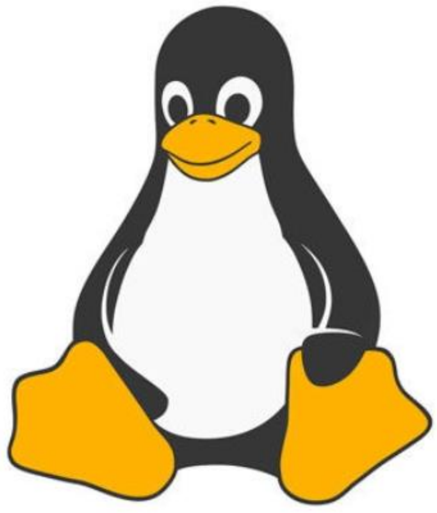


Linux Assignment

Instructor:

Motasem Aldiab

Done by:

Ghassan Yaseen

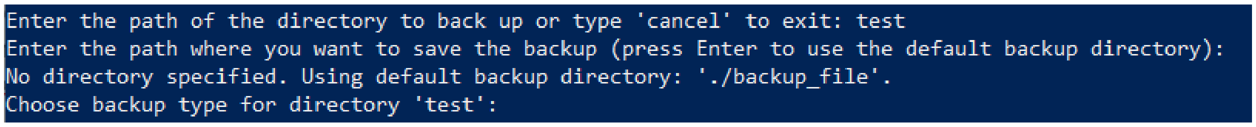
Introduction

This report aims to build a backup script and system health check script using the shell".

Backup Script

The purpose of this script focuses mainly to perform a backup or compress for the user specified directories.

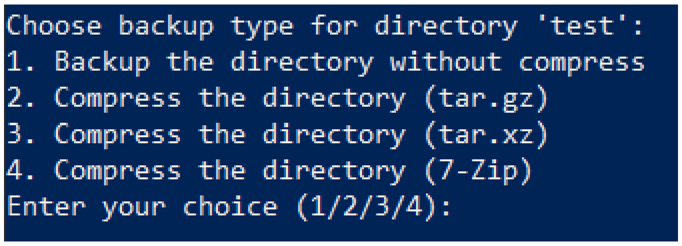
First, I'm going to ask the user what is the directory that he wants to do the backup and I will give a choice to cancel the operation, so if he enters a valid directory, I will ask him where he wants to store the backup directory and if he presses Enter everything will save in the default directory as the figure 1.



```
Enter the path of the directory to back up or type 'cancel' to exit: test
Enter the path where you want to save the backup (press Enter to use the default backup directory):
No directory specified. Using default backup directory: './backup_file'.
Choose backup type for directory 'test':
```

Figure 1

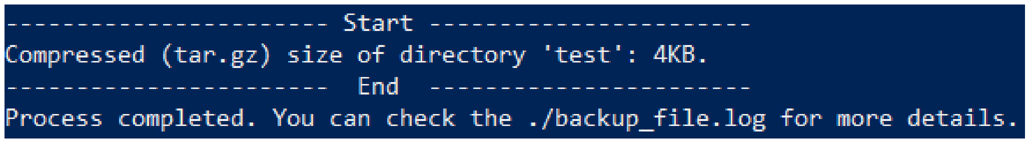
Then, I'm going to ask him if he wants to back up without compress or with compress and what is the type of compress the directory by using the "`user_choice_directory()`" function as figure 2.



```
Choose backup type for directory 'test':
1. Backup the directory without compress
2. Compress the directory (tar.gz)
3. Compress the directory (tar.xz)
4. Compress the directory (7-Zip)
Enter your choice (1/2/3/4):
```

Figure 2

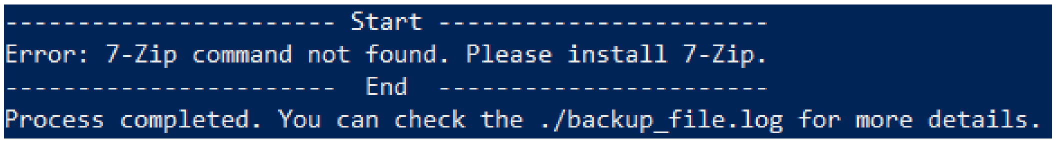
Upon selecting their choice, the user can view the size of the directory that was backed up or compressed. Once the process is complete, a message will be printed informing the user that they can refer to the log file for more details, as figure 3.



```
------------------------ Start ------------------------
Compressed (tar.gz) size of directory 'test': 4KB.
------------------------  End  ------------------------
Process completed. You can check the ./backup_file.log for more details.
```

Figure 3

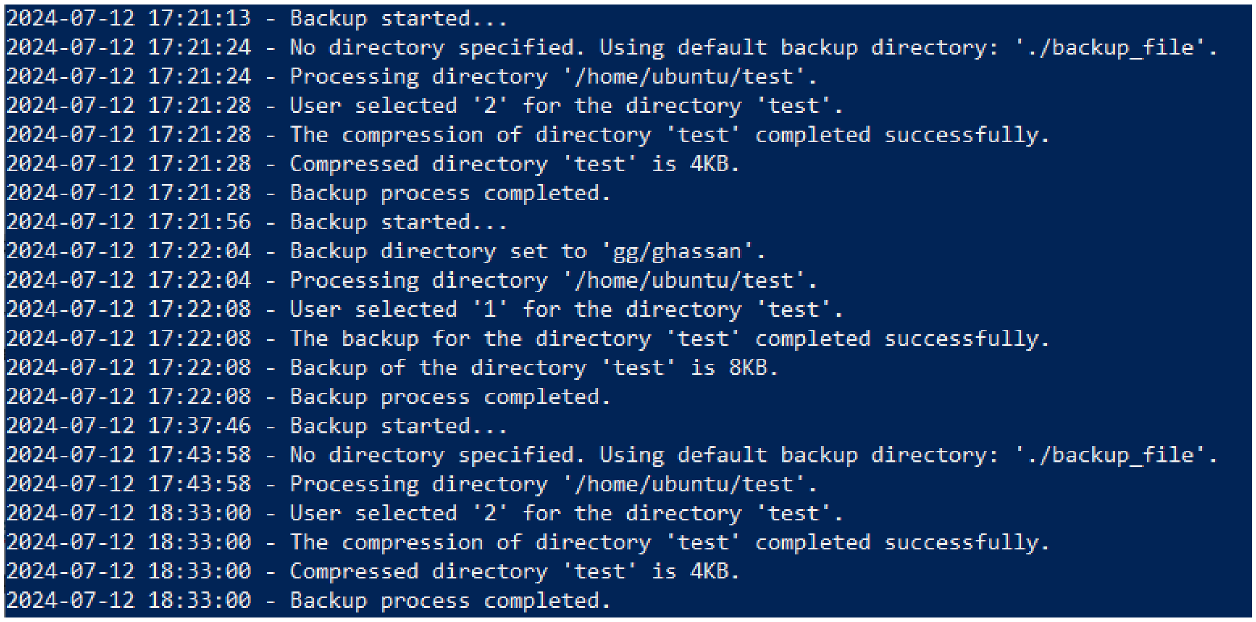
If the necessary compression tool (tar.gz, tar.xz, 7-Zip) is not found, I will inform the user and suggest installing it to resolve any errors, as shown in Figure 4.



```
----------------------- Start -----------------------
Error: 7-Zip command not found. Please install 7-Zip.
-----------------------  End  -----------------------
Process completed. You can check the ./backup_file.log for more details.
```

Figure 4

I will save all system activities in a log file called "backup_file.log", recording the time and a note for each action using the "`log_message()`" function. Additionally, error handling will be implemented to manage incorrect user input or directory issues. Figure 5 will illustrate the contents of the "backup_file.log".

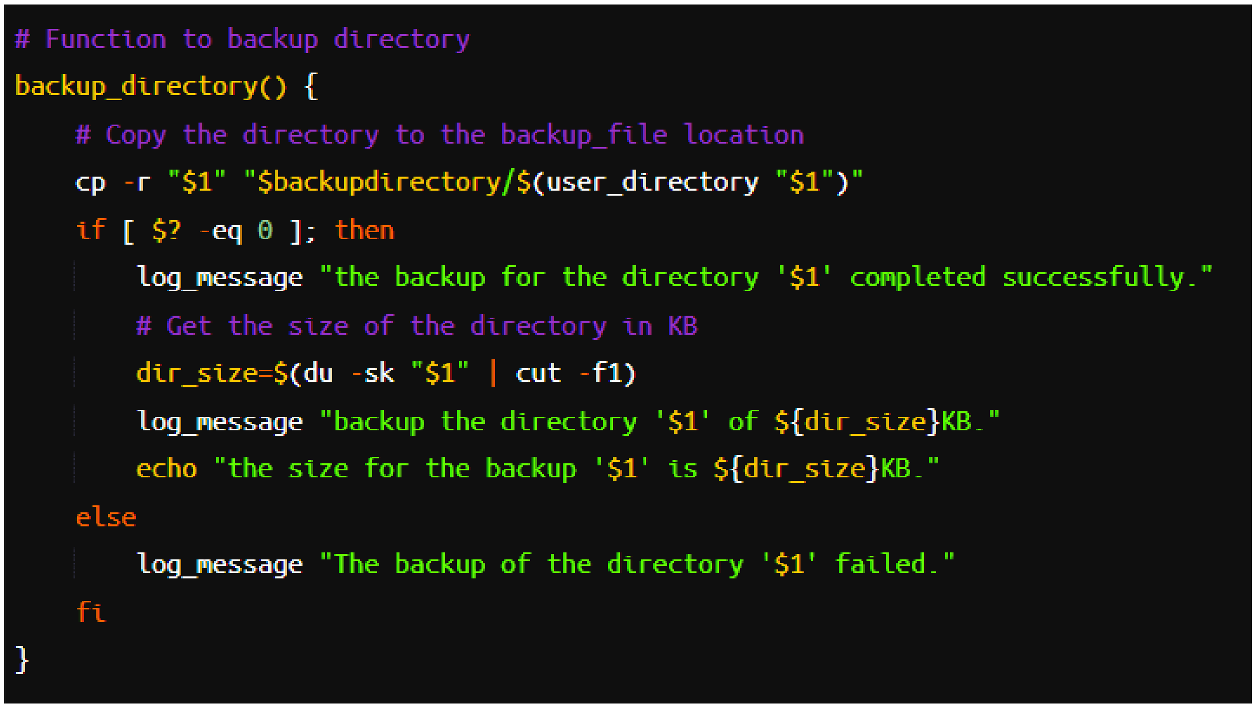


```
2024-07-12 17:21:13 - Backup started...
2024-07-12 17:21:24 - No directory specified. Using default backup directory: './backup_file'.
2024-07-12 17:21:24 - Processing directory '/home/ubuntu/test'.
2024-07-12 17:21:28 - User selected '2' for the directory 'test'.
2024-07-12 17:21:28 - The compression of directory 'test' completed successfully.
2024-07-12 17:21:28 - Compressed directory 'test' is 4KB.
2024-07-12 17:21:28 - Backup process completed.
2024-07-12 17:21:56 - Backup started...
2024-07-12 17:22:04 - Backup directory set to 'gg/ghassan'.
2024-07-12 17:22:04 - Processing directory '/home/ubuntu/test'.
2024-07-12 17:22:08 - User selected '1' for the directory 'test'.
2024-07-12 17:22:08 - The backup for the directory 'test' completed successfully.
2024-07-12 17:22:08 - Backup of the directory 'test' is 8KB.
2024-07-12 17:22:08 - Backup process completed.
2024-07-12 17:37:46 - Backup started...
2024-07-12 17:43:58 - No directory specified. Using default backup directory: './backup_file'.
2024-07-12 17:43:58 - Processing directory '/home/ubuntu/test'.
2024-07-12 18:33:00 - User selected '2' for the directory 'test'.
2024-07-12 18:33:00 - The compression of directory 'test' completed successfully.
2024-07-12 18:33:00 - Compressed directory 'test' is 4KB.
2024-07-12 18:33:00 - Backup process completed.
```

Figure 5

backup_directory

we use the method to copy the directory in the backup location using the command "`cp -r "$1" "$backupdirectory/$( user_directory "$1")"`" , we use the "user_directory" to get the name of the directory from the user, check if it is a valid directory and you take the size of the directory from the memory in KB and add it to the log_file ,the figure 6 show you the code of the method.



```
# Function to backup directory
backup_directory() {
    # Copy the directory to the backup_file location
    cp -r "$1" "$backupdirectory/$(user_directory "$1")"
    if [ $? -eq 0 ]; then
        log_message "the backup for the directory '$1' completed successfully."
        # Get the size of the directory in KB
        dir_size=$(du -sk "$1" | cut -f1)
        log_message "backup the directory '$1' of ${dir_size}KB."
        echo "the size for the backup '$1' is ${dir_size}KB."
    else
        log_message "The backup of the directory '$1' failed."
    fi
}
```

Figure 6

copmress_directory

we use 3 methods to compress the directory in the compress location:

1) compress_directory_tar_gz() using the command "`tar -czf "$backup_file" -C "$1" . 2>> "$log_file"`".
2) compress_directory_tar_xz() using the command "`tar -cJf "$backup_file" -C "$1" . 2>> "$log_file"`".
3) compress_directory_7zip() using the command "`7z a "$backup_file" "$1" &>> "$log_file"`".

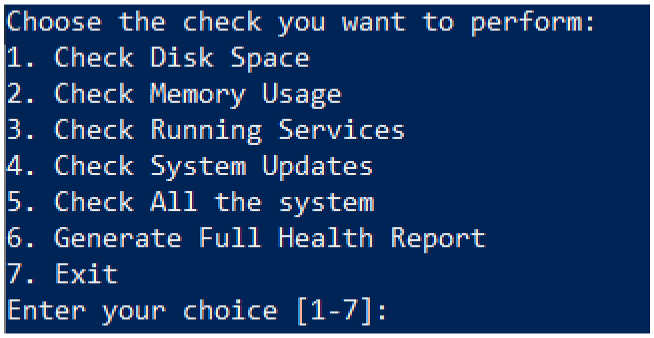
We use the user_directory variable to get the name of the directory from the user and append the date to ensure each file has a unique name. The script checks if it is a valid directory, retrieves its size from memory in KB, and logs this information in the log_file. Each step includes error handling to ensure everything is functioning correctly and to inform the user to install any required compression tools if needed.

Note

It's correct that Linux typically creates backup copies without compression by default, unless specifically instructed by the user to compress them. This approach allows users flexibility in choosing whether to create uncompressed backups or compressed archive files based on their needs, that's why I let my code give the choice to the user.

System Health Check

This script's purpose is to show general system health information, it includes a function for each check and gives the user a choice to choose which part he wants to check and put everything in the final report as figure 7.

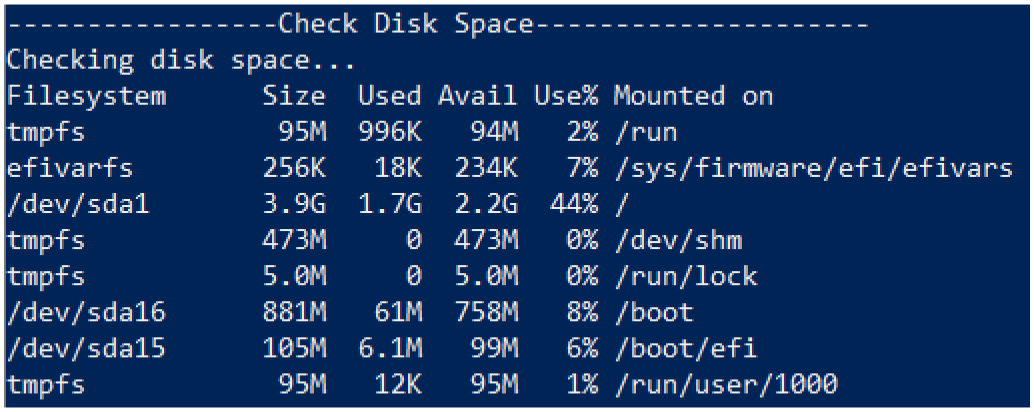


```
Choose the check you want to perform:
1. Check Disk Space
2. Check Memory Usage
3. Check Running Services
4. Check System Updates
5. Check All the system
6. Generate Full Health Report
7. Exit
Enter your choice [1-7]:
```

Figure 7

Check Disk Space

This Function will use the "`df -h`" with a beautiful interface show you the file system name, the size, space that is currently used, space that is available for use, the percentage of the filesystem's total space that is currently used and the directory where the filesystem is mounted every one of them as figure 8:

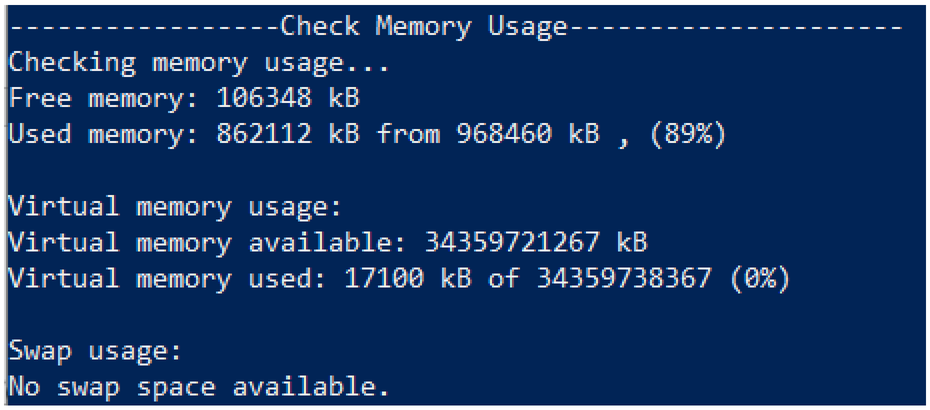


```
-----------------Check Disk Space---------------------
Checking disk space...
Filesystem      Size  Used Avail Use% Mounted on
tmpfs            95M  996K   94M   2% /run
efivarfs        256K   18K  234K   7% /sys/firmware/efi/efivars
/dev/sda1       3.9G  1.7G  2.2G  44% /
tmpfs           473M     0  473M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/sda16      881M   61M  758M   8% /boot
/dev/sda15      105M  6.1M   99M   6% /boot/efi
tmpfs            95M   12K   95M   1% /run/user/1000
```

Figure 8

Check Memory Usage

We have 3 functions 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 grasp the free memory and the total memory from 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% the output will be something as figure 9.

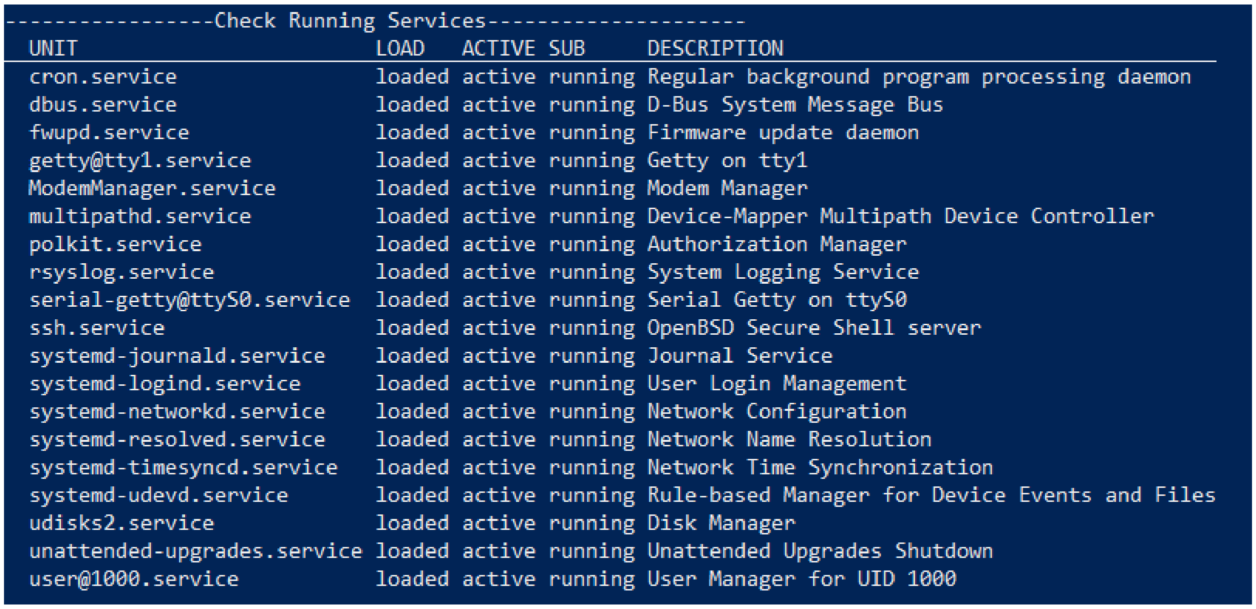


```
-----------------Check Memory Usage---------------------
Checking memory usage...
Free memory: 106348 kB
Used memory: 862112 kB from 968460 kB , (89%)

Virtual memory usage:
Virtual memory available: 34359721267 kB
Virtual memory used: 17100 kB of 34359738367 (0%)

Swap usage:
No swap space available.
```

Figure 9

Check Running Services

This function use to know what the running services is using this command “`systemctl list-units --type=service --state=running`” in this code we choose the service type from the systemctl (tool used to manage and control the systemd system and service manager in Linux systems) with the running state and print those service and if they active or not as figure 10:



```
-----------------Check Running Services---------------------
  UNIT                         LOAD   ACTIVE SUB     DESCRIPTION
  cron.service                 loaded active running Regular background program processing daemon
  dbus.service                 loaded active running D-Bus System Message Bus
  fwupd.service                loaded active running Firmware update daemon
  getty@tty1.service           loaded active running Getty on tty1
  ModemManager.service         loaded active running Modem Manager
  multipathd.service           loaded active running Device-Mapper Multipath Device Controller
  polkit.service               loaded active running Authorization Manager
  rsyslog.service              loaded active running System Logging Service
  serial-getty@ttyS0.service   loaded active running Serial Getty on ttyS0
  ssh.service                  loaded active running OpenBSD Secure Shell server
  systemd-journald.service     loaded active running Journal Service
  systemd-logind.service       loaded active running User Login Management
  systemd-networkd.service     loaded active running Network Configuration
  systemd-resolved.service     loaded active running Network Name Resolution
  systemd-timesyncd.service    loaded active running Network Time Synchronization
  systemd-udevd.service        loaded active running Rule-based Manager for Device Events and Files
  udisks2.service              loaded active running Disk Manager
  unattended-upgrades.service  loaded active running Unattended Upgrades Shutdown
  user@1000.service            loaded active running User Manager for UID 1000
```

Figure 10

Check System Updates

This function is used to know what the recent system update is by taking the last 10 lines from the API history log which are the last system updates (details of installed, upgraded, or removed packages).

Generate Full Health Report

This option will call all the previous functions, print them and put them in a file call system_health_report.txt.