Sathyanarayana Ramesh

Shell Scripting for System Administration

PHASE 1: SPECIFICATION:

A. Creating a user information script

This script's primary goal is to display user information, comprising the user's full name, home directory, and shell type.

```
# ./userinfo.sh test
User: test
Full Name: test acc
Home Directory: /home/test
Shell Type: /bin/bash
```

B. Creating a system health check script

This script's primary goal is to report system information, comprising current date and time, system uptime, number of logged users, memory usage and disk usage.

```
# ./healthcheck.sh
Health Check Report
-------------------
Date and Time: 2023-11-19 20:05:48
Uptime:  20:05:48 up  1:04,  1 user,  load average: 0.00, 0.00, 0.00
Logged in Users: 1
Memory Usage: Mem:        3.7Gi     479Mi     2.9Gi     3.0Mi     387Mi     3.1Gi
Disk Usage: Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc       1007G  1.1G  955G   1% /
```

C. Creating a directory backup script

This script's primary goal is to create compressed backup of the provided directory by appending the current date into the given destination directory.

```
# pwd
/home/user/src
# ls -lrt
-rw-r--r-- 1 user user 15 Nov 19 20:31 file1.txt
-rw-r--r-- 1 user user 20 Nov 19 20:31 file2.txt
# ./backupdir.sh /home/user/src /home/user/dst
Backup completed. The backup file is stored as: /home/user/dst/backup_2023_11_19.tar.gz
# cd dst
# ls -lrt
-rw-r--r-- 1 user user 198 Nov 19 20:38 backup_2023_11_19.tar.gz
```

D. Creating a batch file rename script

This script's primary goal is to rename all the files present in the provided directory, renamed file names would begin with the supplied prefix argument.

```
$ ./rename.sh /home/user/src new
Renamed: file1.txt to new_file1.txt
Renamed: file2.txt to new_file2.txt
File renaming completed with prefix 'new'
```

E. Creating a process monitor script
This script's primary goal is to monitor the process provided as an argument. If the process is not running script would start the process and log the event. If the process is found running or currently executing it's memory and CPU usage would be logged.

```
# ./processmonitor.sh bash
Process 'bash' is running.
Memory Usage: 0.0
0.0%
CPU Usage: 0.0
0.0%
# ./processmonitor.sh date
Process 'date' is not running. Starting the process...
Sun Nov 19 21:02:41 EST 2023
```

Testing the shell scripts using the following steps:

A. userinfo.sh expects 'user_name' to be provided in the command line.
B. healthcheck.sh expects no argument and provides health status when it is run.
C. backupdir.sh expects two arguments, first argument is the absolute path of the source directory followed by the path for destination directory where the backup has to be stored.
D. rename.sh expects two arguments, the path of the directory followed by the prefix for renaming the files.
E. Processmonitor.sh expects process name as a single argument.

<u>PHASE 2 DESIGN:</u>

2.1 Scripts and commands

This program consists of 5 scripts

A. **userinfo.sh**
The script takes a username as a command-line argument, retrieves information about that user using the getent passwd command, and then displays relevant details such as the full name, home directory, and shell type. If the user doesnot exist or the username not provided, script throws the error. Below are the commands used in the script.
getent passwd username
getent passwd username is used to retrieve information about a specific user from the passwd database.

cut -d delimiter -f fields
The cut command is used to extract specific portions of text from each line of a file or from piped input.
-d delimiter: Specifies the delimiter used to separate fields

-f fields: Specifies which fields to extract.

getent passwd john | cut -d: -f5

In the script, cut is used to extract information from the output of the getent passwd command.

B. healthcheck.sh

This script prints to the console a health check report with the gathered information, including the current date and time, system uptime, number of logged-in users, memory usage, and disk usage. Below are the commands used in the script.

(date +"%Y-%m-%d %H:%M:%S")

The date command is used to display the current date and time.

The format specified with +"%Y-%m-%d %H:%M:%S" ensures that the output is in the desired format (YYYY-MM-DD HH:MM:SS).

uptime

The uptime command provides information about the system's uptime (how long the system has been running) and the average system load.

(who | wc -l)

The who command shows information about currently logged-in users.

The output of who is then piped (|) to the wc -l command, which counts the number of lines in the output which in-turn gives the number of currently logged user.

(free -h | grep Mem)

The free command is used to display information about system memory usage.

The -h option is used to display the output in a human-readable format.

The output is piped to 'grep Mem' to filter out information related to memory.

(df -h /)

The 'df' command is used to display information about disk space usage.

The '-h' option is used to display output in a human readable format.

The '/' specifies that the disk space information for the root filesystem.

C. backupdir.sh

The script checks if a valid directory path is provided, generates a unique backup filename with the current date, specifies the backup location, and then creates a compressed backup of the specified directory using the tar command.

tar -czf "${backup_location}/${backup_filename}" -C "$(dirname "$input_directory")" "$(basename "$input_directory")"

tar command with options:

-czf: Options for creating a compressed archive file using gzip compression.

"${backup_location}/${backup_filename}": Specifies the destination path for the backup file.

-C "$(dirname "$input_directory")": Changes to the parent directory of the input directory.

"$(basename "$input_directory")": Specifies the base name of the input directory to be included in the backup.

D. rename.sh

The script checks if the correct number of arguments is provided, ensures that the specified source directory exists, throws error if the destination directory is not present, navigates to the source directory, and then renames each file in the directory by adding a specified prefix.

The script loops through the list of files in the source directory and rename using the below command.

mv "$file" "$new_name"

The mv command in Linux is used for moving or renaming files and directories. When mv command filename if it follows by pathname then the file would be moved. Instead of path if filename is provided then mv renames the file.

E. processmonitor.sh

The script checks if a process is running, and if not, it starts the process, logs the event, and provides information about the process's memory and CPU usage if it is running. The script is designed to monitor a specific process based on the provided process name.

pgrep -x "$process_name" > /dev/null

The pgrep command searches for a process by name and returns its process ID (PID).
-x ensures an exact match for the process name.
> /dev/null redirects the standard output to null, suppressing any output.
The condition checks if the process is running based on the exit status.

ps -e

The ps command provides information about processes.
-e lists information about all processes.

process_monitor.log

Logs events to a file named process_monitor.log, including the date and time when a process is started.

2.2 Shell Script / Batch Script:

1. userinfo.sh

```
#!/bin/bash
# Check if the username argument is provided
if [ $# -ne 1 ]; then
  echo "No Arguments Provided"
  exit 1
fi
# Get user information
username=$1
user_info=$(getent passwd "$username")
# Check if the user is found
if [ -z "$user_info" ]; then
  echo "User not found: $username"
  exit 1
fi
# Extract user details
full_name=$(echo "$user_info" | cut -d: -f5 | cut -d, -f1)
home_directory=$(echo "$user_info" | cut -d: -f6)
shell_type=$(echo "$user_info" | cut -d: -f7)
# Display user information
echo "User: $username"
echo "Full Name: $full_name"
echo "Home Directory: $home_directory"
echo "Shell Type: $shell_type"
```

2. healthcheck.sh

```
#!/bin/bash
# Current date and time
current_date_time=$(date +"%Y-%m-%d %H:%M:%S")
#current_date_time=$(date)
# System uptime
uptime_info=$(uptime)
# Total number of users currently logged in
logged_in_users=$(who | wc -l)
# Memory usage
memory_info=$(free -h | grep Mem)
# Disk usage
disk_info=$(df -h /)
# Display system information
echo "Health Check Report"
echo "-------------------"
echo "Date and Time: $current_date_time"
echo "Uptime: $uptime_info"
echo "Logged in Users: $logged_in_users"
echo "Memory Usage: $memory_info"
echo "Disk Usage: $disk_info"
```

3. ***backupdir.sh***

```
#!/bin/bash
# Check if the directory path argument is provided
if [ $# -eq 0 ]; then
  echo "No Arguments provided"
  exit 1
fi
if [ "$#" -ne 2 ]; then
  echo "Script expects two arguments but only one was provided"
  exit 1
fi
# Input directory path
input_directory="$1"
# Check if the input is a directory
if [ ! -d "$input_directory" ]; then
  echo "Error: '$input_directory' is not a directory."
  exit 1
fi
# Get the current date
current_date=$(date +"%Y_%m_%d")

# Backup file name with current date
backup_filename="backup_${current_date}.tar.gz"
# Backup location
backup_location="$2"  # Change this to your desired backup location
# Create a compressed backup of the entire directory
tar -czf "${backup_location}/${backup_filename}" -C "$(dirname "$input_directory")" "$(basename
"$input_directory")"
echo "Backup completed. The backup file is stored as: ${backup_location}/${backup_filename}"
```

4. ***rename.sh***

```
#!/bin/bash
# Check if the correct number of arguments is provided
if [ "$#" -lt 2 ]; then
    echo "Script expects 2 arguments: path_to_dir prefix : missing arguments "
  exit 1
fi
# Directory path
directory_path="$1"
# Prefix
prefix="$2"
# Check if the directory exists
if [ ! -d "$directory_path" ]; then
  echo "Error: Directory '$directory_path' not found."
  exit 1
fi
# Check if a prefix is provided
if [ -z "$prefix" ]; then
  echo "Error: No prefix provided."
  exit 1
fi
# Navigate to the directory
cd "$directory_path" || exit 1
# Rename files by adding the prefix
for file in *; do
  if [ -f "$file" ]; then
    new_name="${prefix}_${file}"
    mv "$file" "$new_name"
    echo "Renamed: $file to $new_name"
  fi
done
echo "File renaming completed with prefix '$prefix'"
```

5. ***processmonitor.sh***

```
#!/bin/bash
# Check if the process name argument is provided
if [ $# -eq 0 ]; then
  echo "Script expects one argument but none provided"
  exit 1
fi
# Process name
process_name=$1
# Check if the process is running
if pgrep -x "$process_name" > /dev/null; then
  # Process is running, get memory and CPU usage
  memory_usage=$(ps -e -o pid,%mem,cmd | awk -v process="$process_name" '$NF==process {print
$2}')
  cpu_usage=$(ps -e -o pid,%cpu,cmd | awk -v process="$process_name" '$NF==process {print $2}')
  echo "Process '$process_name' is running."
  echo "Memory Usage: $memory_usage%"
  echo "CPU Usage: $cpu_usage%"
else
  # Process is not running, start it and log the event
```

```
    echo "Process '$process_name' is not running. Starting the process..."
    "$process_name" &
    # Log the event
    echo "$(date): Process '$process_name' started." >> process_monitor.log
fi
```

PHASE 3: RISK ANALYSIS

No risk.

PHASE 4: VERIFICATION

In order to successfully run the userinfo.sh the provided user should already exist. adduser.sh creates new user ‘test’ which can then verified using userinfo.sh. healthcheck.sh doesn’t expects any argument. backupdir.sh source and destination directory both should be present. rename.sh requires minimum of one file to be present in the source directory. Processmonitor.sh doesn’t expects process name as an argument. All the scripts has to be provided +x permission to test and executes successfully.

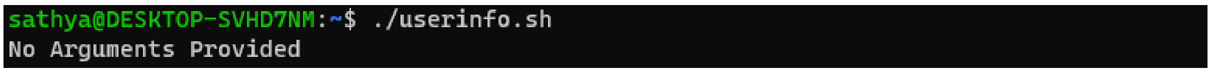
PHASE 5: CODING

The code has been pushed to the git hub, and the link has been provided below.

https://github.com/akgWMU/pa5-shell-scripting-Sathya-Ramesh

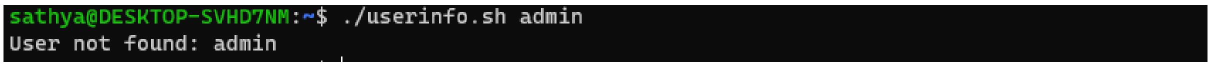
PHASE 6: TESTING

1) Testing of userinfo.sh
   When no user is passed.



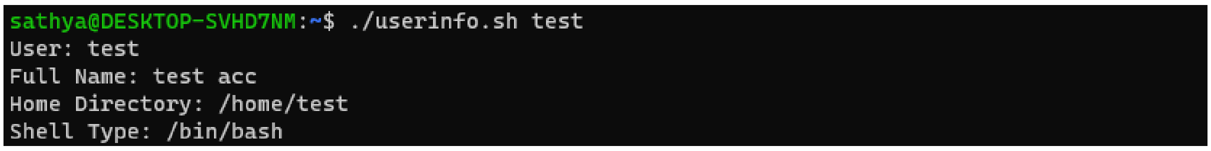
```
sathya@DESKTOP-SVHD7NM:~$ ./userinfo.sh
No Arguments Provided
```

   When user is not present.



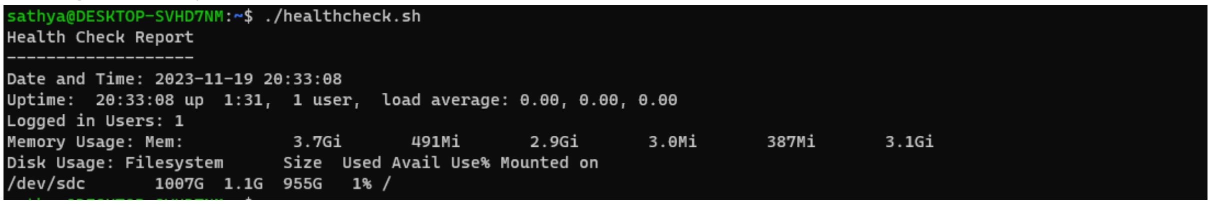
```
sathya@DESKTOP-SVHD7NM:~$ ./userinfo.sh admin
User not found: admin
```

   When user is present.



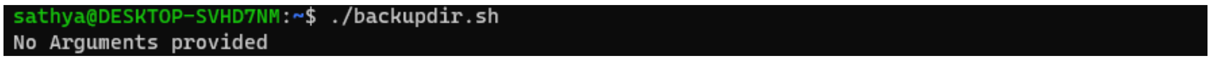
```
sathya@DESKTOP-SVHD7NM:~$ ./userinfo.sh test
User: test
Full Name: test acc
Home Directory: /home/test
Shell Type: /bin/bash
```

2) Testing healthreport.sh



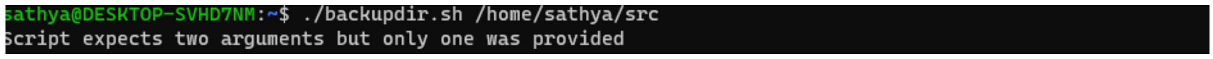
```
sathya@DESKTOP-SVHD7NM:~$ ./healthcheck.sh
Health Check Report
-------------------
Date and Time: 2023-11-19 20:33:08
Uptime:  20:33:08 up  1:31,  1 user,  load average: 0.00, 0.00, 0.00
Logged in Users: 1
Memory Usage: Mem:           3.7Gi       491Mi       2.9Gi       3.0Mi       387Mi       3.1Gi
Disk Usage: Filesystem      Size  Used Avail Use% Mounted on
/dev/sdc       1007G  1.1G  955G   1% /
```

3) Testing backupdir.sh
   When no arguments passed.



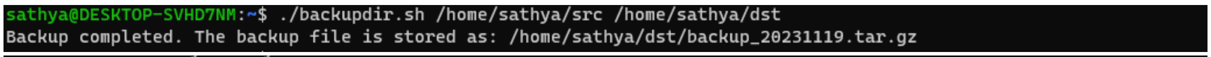
```
sathya@DESKTOP-SVHD7NM:~$ ./backupdir.sh
No Arguments provided
```

   When destination path is not provided.

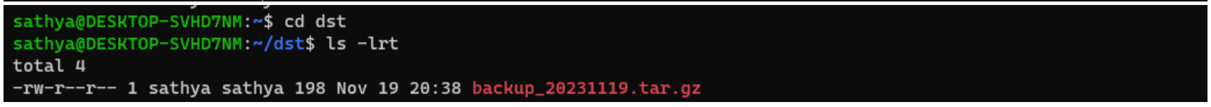


```
athya@DESKTOP-SVHD7NM:~$ ./backupdir.sh /home/sathya/src
cript expects two arguments but only one was provided
```

   When the destination is provided.

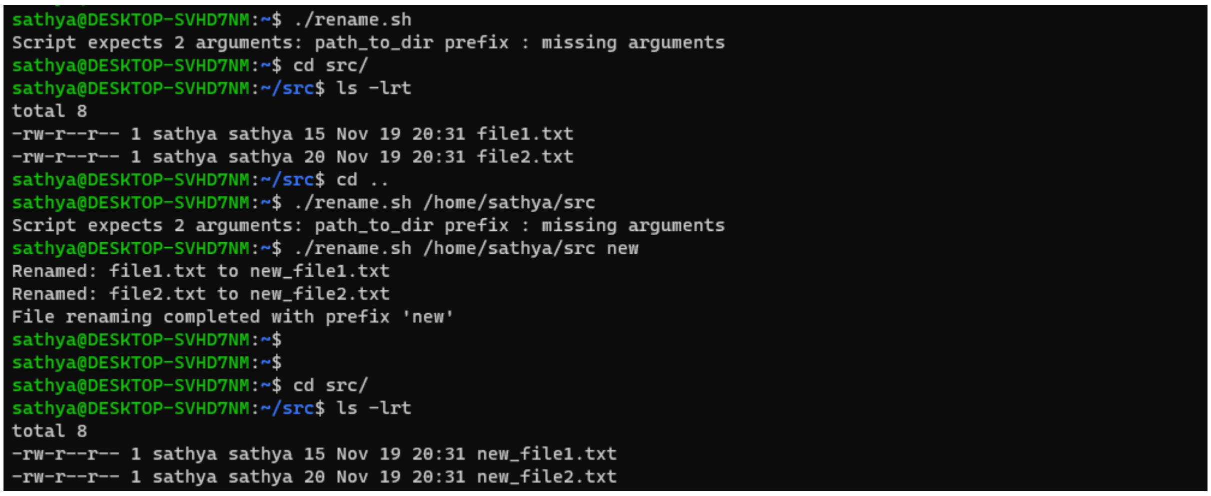


```
sathya@DESKTOP-SVHD7NM:~$ ./backupdir.sh /home/sathya/src /home/sathya/dst
Backup completed. The backup file is stored as: /home/sathya/dst/backup_20231119.tar.gz
```



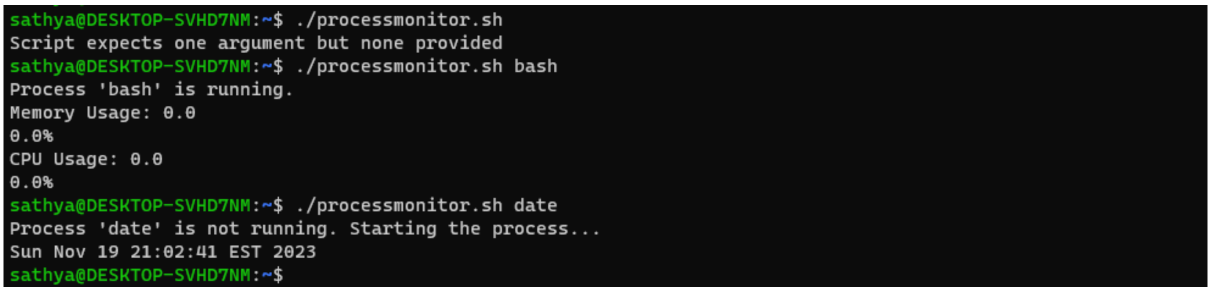
```
sathya@DESKTOP-SVHD7NM:~$ cd dst
sathya@DESKTOP-SVHD7NM:~/dst$ ls -lrt
total 4
-rw-r--r-- 1 sathya sathya 198 Nov 19 20:38 backup_20231119.tar.gz
```

4) Testing rename.sh



```
sathya@DESKTOP-SVHD7NM:~$ ./rename.sh
Script expects 2 arguments: path_to_dir prefix : missing arguments
sathya@DESKTOP-SVHD7NM:~$ cd src/
sathya@DESKTOP-SVHD7NM:~/src$ ls -lrt
total 8
-rw-r--r-- 1 sathya sathya 15 Nov 19 20:31 file1.txt
-rw-r--r-- 1 sathya sathya 20 Nov 19 20:31 file2.txt
sathya@DESKTOP-SVHD7NM:~/src$ cd ..
sathya@DESKTOP-SVHD7NM:~$ ./rename.sh /home/sathya/src
Script expects 2 arguments: path_to_dir prefix : missing arguments
sathya@DESKTOP-SVHD7NM:~$ ./rename.sh /home/sathya/src new
Renamed: file1.txt to new_file1.txt
Renamed: file2.txt to new_file2.txt
File renaming completed with prefix 'new'
sathya@DESKTOP-SVHD7NM:~$
sathya@DESKTOP-SVHD7NM:~$
sathya@DESKTOP-SVHD7NM:~$ cd src/
sathya@DESKTOP-SVHD7NM:~/src$ ls -lrt
total 8
-rw-r--r-- 1 sathya sathya 15 Nov 19 20:31 new_file1.txt
-rw-r--r-- 1 sathya sathya 20 Nov 19 20:31 new_file2.txt
```

5) Testing proccessmonitor.sh



```
sathya@DESKTOP-SVHD7NM:~$ ./processmonitor.sh
Script expects one argument but none provided
sathya@DESKTOP-SVHD7NM:~$ ./processmonitor.sh bash
Process 'bash' is running.
Memory Usage: 0.0
0.0%
CPU Usage: 0.0
0.0%
sathya@DESKTOP-SVHD7NM:~$ ./processmonitor.sh date
Process 'date' is not running. Starting the process...
Sun Nov 19 21:02:41 EST 2023
sathya@DESKTOP-SVHD7NM:~$
```

PHASE 7: REFINING THE PROGRAM

The script userinfo.sh works only if user is present, to make it convenience new script adduser.sh has been created, when the scripts get executed by the admin creates a new user.

PHASE 8: PRODUCTION

Folder containing all the scripts & report has been compressed to a zip file and the same has been added to the dropbox submission.

References:

[1] https://unix.stackexchange.com/questions/31414/how-can-i-pass-a-command-line-argument-into-a-shell-script

[2] https://www.digitalocean.com/community/tutorials/linux-commands

[3] https://www.guru99.com/managing-processes-in-linux.html

[4] https://www.howtogeek.com/562941/how-to-use-the-awk-command-on-linux/