Bash Script Examples

1. Check if a file exists in specific directory
2. Schedule BASH script file in CRONTAB
3. Ping whether website/IP is responsive or not
4. List all users one by one from /etc/passwd file
5. Script that backup all the files in example directory and save them all in a .tar archive
6. Adding scripts to your PATH so that are accessible from any folder.
7. Script for start|stop|restart iptables
8. Cleaning files in folders not used last 30 days
9. Organize files or folders per file type into dedicated folders
10. Server Monitor Script – SSH and SCP

Check if a file exists in specific directory.

File Operations:

```
-s    file exists and is not empty
-f    file exists and is not a directory
-d    directory exists
-x    file is executable
-w    file is writable
-r    file is readable
```

```
#!/bin/bash
if [[ -f /home/user/example_file ]]
then
        echo "File exist"
else
        echo "File does not exist"
fi
```

Schedule BASH script file in CRONTAB

Make sure the script is executable:

```
chmod +x /home/username/scripts/myscript.sh
```

Open the crontab file for the current user by running:

```
crontab -e
```

This will open the crontab file in the default text editor.

Schedule the Script - Add a line in the crontab file to specify when you want the script to run. The general format is:

```
* * * * * /path/to/your/script
Or
* * * * * bash /path/to/your/script
```

Here’s what each field represents:

- `minute` (0 - 59)
- `hour` (0 - 23)
- `day of month` (1 - 31)
- `month` (1 - 12)
- `day of week` (0 - 7) (Sunday can be 0 or 7)

Ping whether website/IP is responsive or not

```
#!/bin/bash

ping -c 1 www.abv.bg

if [ $? -eq 0 ]

then

    echo "abv.bg е достъпен."

else

    echo "abv.bg не е достъпен."

fi
```

Специалната променлива `$?` съдържа статус кода на последната изпълнена команда, който се използва за проверка на резултата от `ping`.

List all users one by one from /etc/passwd file

```
#!/bin/bash

i=1
for username in `awk -F: '{print $1}' /etc/passwd`
do
 echo "Username $((i++)) : $username"
done
```

- **`awk`**: A powerful text processing tool in Unix/Linux that scans and processes lines in files based on the given program/script.
- **`-F:`**: This option sets the field separator to a colon (**`:`**). In the **`/etc/passwd`** file, each field in a record (line) is separated by a colon. So, this tells **`awk`** to treat the colon as the delimiter between fields.
- **`'{print $1}'`**: This is the **`awk`** program/script itself, enclosed in single quotes to be treated as a single argument. **`{print $1}`** tells **`awk`** to print the first field of each record it processes. In the context of **`/etc/passwd`**, the first field is the username.
- **`/etc/passwd`**: This specifies the input file for **`awk`** to process. **`/etc/passwd`** is a standard Unix file that contains information about user accounts on the system.

So, what this script does is:

- Use `awk` to extract the first field (username) from each line in the `/etc/passwd` file.
- Iterate over this list of usernames with the `for` loop.
- For each username, execute the body of the loop (which is not shown in your snippet).

Script that backup all the files in example directory and save them all in a .tar archive

```
#!/bin/bash

# Author:

# Created:

# Last Modified:

# Description:

# Creates a backup in the current directory of

# all files in the home directory

currentdir=$(pwd)

echo "Backup file creation in $currentdir"

tar -cf "$currentdir/my_backup_$(date +%d-%m-%Y_%H-%M-%S)".tar $currentdir 2>/dev/null

# tar -cf "path where to store and what will be the name".tar "path - target backup directory" 2>/dev/null

echo "Backup completed successfully"

exit 0
```

In Bash, `2>/dev/null` is used to redirect the standard error (stderr) output to `/dev/null`, effectively discarding any error messages generated by the command. Here's a breakdown of the components:

- `2`: This represents the file descriptor for standard error (stderr). In Unix-like operating systems, each open file is assigned a file descriptor. By convention, 0 is stdin (standard input), 1 is stdout (standard output), and 2 is stderr.
- `>`: This is the redirection operator. It is used to redirect the output from the command on its left to the file or device on its right.
- `/dev/null`: This is a special file that discards all data written to it. Think of it as a black hole for data. When you redirect output to `/dev/null`, you're essentially telling the system to throw it away.

So, when you see a command like this in Bash:

```
some-command 2>/dev/null
```

It means "execute `some-command` and discard any error messages it produces." This is often done in scripts or command lines where error messages from certain commands are not important or where you want to prevent error messages from cluttering the output.

Adding scripts to your PATH so that are accessible from any folder.

Edit your ~/.profile file to add a custom folder to your PATH

- vi ~/.bashrc

Add this line command at the bottom of the file.

- export PATH="$PATH:/path/to/script_directory

Reload the ~/.profile file

- source ~/.bashrc

Add your scripts to the new folder and run like normal commands!

- mv my_script script_directory

You can now run your scripts like regular commands!

- my_script

Note: Scripts must have execute permissions to run.

Script for start|stop|restart iptables

```
#!/bin/bash


start_iptables() {
   echo "start iptables rules"
   iptables -P INPUT DROP
   iptables -P FORWARD DROP
   iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
   iptables -A INPUT -i lo -j ACCEPT
   iptables -A INPUT -p tcp --dport 22 -j ACCEPT
   iptables -A INPUT -p tcp --dport 80 -j ACCEPT
   iptables -A INPUT -p tcp --dport 443 -j ACCEPT
}
stop_iptables() {
   echo " stop iptables rules"
   iptables -F
   iptables -t nat -F
   iptables -t mangle -F
   iptables -X
}
```

```
restart_iptables() {
   echo " restart iptables rules”
   stop_iptables
   start_iptables
}
case "$1" in
   start) start_iptables;;
   stop) stop_iptables;;
   restart) restart_iptables;;
   *) echo "Please use: $0 {start|stop|restart}"
      exit 1
esac
exit 0
```

Cleaning files in folders not used last 30 days

```
#!/bin/bash
# Author: Rosen
echo "Available folders for cleaning are:"
ls -d /*/
# extract all folders into array from the output of a command ls
readarray -t dirs < <(ls -d /*/)
#set 'a' variable for while-loop check of correct folder entered
a=0
while [ $a = 0 ]; do
      read -r -p "Enter folder name in format '/folder/': " folder
      # check whether user input(read command) match an 'item' in the folder array
      for item in "${dirs[@]}"; do
            if [ "$item" = "$folder" ]; then
                  echo "You've selected : $folder"
                  # collect all suspect files in array 'files'
                  readarray -t files < <(find "$folder" -maxdepth 2 -mtime -30 -type f)
                  # check for empthy array
                  if [ "${files}" != '' ]; then
                        echo "Files to be removed are:"
```

```
                    for rmfile in "${files[@]}"; do
                        rm -i "$rmfile"
                    done
                else
                    echo "No files for cleaning in this folder!"
                fi
                a=1
            fi
        done
done
```

----------===---

```
#!/bin/bash
# Author: Ziyad Yehia
# Created: 8th February 2021
# Last Modified: 8th February 2021

# Description:
# Prompts you to remove all files in a specified folder that have not
# been modified within a given number of days

# Usage: ./cruft_remover.sh

read -p "Which folder do you want to remove unmodified files from?: " folder
read -p "How many days is too old?: " days

readarray -t files < <(find $folder -maxdepth 1 -type f -mtime "+$days")

for file in "${files[@]}"; do
    rm -i "$file"
done
```

Organize files or folders per file type into dedicated folders

```
#!/bin/bash

# declare folder with all types of files for the script

folder_path="/home/root/scripts/organiser"

while read -r line; do

        case "$line" in

        *.jpeg | *.png | *.jpg)

                if [ -d "$folder_path"/images ]; then

                        mv "$folder_path"/"$line" "$folder_path"/images/

                else
```

```
                    mkdir "$folder_path"/images
                    mv "$folder_path"/"$line" "$folder_path"/images/
            fi ;;
*.doc | *.docx | *.txt | *.pdf)
        if [ -d "$folder_path"/documents ]; then
                    mv "$folder_path"/"$line" "$folder_path"/documents/
            else
                    mkdir "$folder_path"/documents
                    mv "$folder_path"/"$line" "$folder_path"/documents/
            fi ;;
*.xls | *.xlsx | *.csv)
        if [ -d "$folder_path"/spreadsheets ]; then
                    mv "$folder_path"/"$line" "$folder_path"/spreadsheets/
            else
                    mkdir "$folder_path"/spreadsheets
                    mv "$folder_path"/"$line" "$folder_path"/spreadsheets/
            fi ;;
*.sh)
        if [ -d "$folder_path"/scripts ]; then
                    mv "$folder_path"/"$line" "$folder_path"/scripts/
            else
                    mkdir "$folder_path"/scripts
                    mv "$folder_path"/"$line" "$folder_path"/scripts/
            fi ;;
*.zip | *.tar | *.tar.gz | *.tar.bz2)
        if [ -d "$folder_path"/archives ]; then
                    mv "$folder_path"/"$line" "$folder_path"/archives/
            else
                    mkdir "$folder_path"/archives
                    mv "$folder_path"/"$line" "$folder_path"/archives/
            fi ;;
*.ppt | *.pptx)
```

```
        if [ -d "$folder_path"/presentations ]; then
                mv "$folder_path"/"$line" "$folder_path"/presentations/
          else
                mkdir "$folder_path"/presentations
                mv "$folder_path"/"$line" "$folder_path"/presentations/
          fi ;;
    *.mp3)
        if [ -d "$folder_path"/audio ]; then
                mv "$folder_path"/"$line" "$folder_path"/audio/
          else
                mkdir "$folder_path"/audio
                mv "$folder_path"/"$line" "$folder_path"/audio/
          fi ;;
    *.mp4)
        if [ -d "$folder_path"/video ]; then
                mv "$folder_path"/"$line" "$folder_path"/video/
          else
                mkdir "$folder_path"/video
                mv "$folder_path"/"$line" "$folder_path"/video/
          fi ;;
    *) echo "Left in $folder_path" ;;
    esac
done < <(ls "$folder_path")
```

---=

```
#!/bin/bash

# Author: Ziyad Yehia
# Created: 8th February 2021
# Last Modified: 8th February 2021

# Description:
# Keeps a folder specified by the user clean by moving files into
# folders based on their file extensions
# Usage: ./folder_organiser.sh

read -p "Which folder do you want to organise?: " folder

while read filename; do
    case "$filename" in
        *.jpg|*.jpeg|*.png)
            subfolder="images" ;;
        *.doc|*.docx|*.txt|*.pdf)
            subfolder="documents" ;;
        *.xls|*.xlsx|*.csv)
            subfolder="spreadsheets" ;;
        *.sh)
            subfolder="scripts" ;;
        *.zip|*.tar|*.tar.gz|*.tar.gz.bz2)
            subfolder="archives" ;;
        *.ppt|*.pptx)
            subfolder="presentations" ;;
        *.mp3)
            subfolder="audio" ;;
        *.mp4)
            subfolder="video" ;;
        *)
            subfolder="." ;;
    esac

    if [ ! -d "$folder/$subfolder" ]; then
       mkdir "$folder/$subfolder"
    fi

    mv "$filename" "$folder/$subfolder"
done < <(ls "$folder")
```

Server Monitor Script – SSH and SCP

```
#!/bin/bash

# performance_checker.sh


date >> performance.log


ping -c 1 google.com &> /dev/null

if [ "$?" -eq 0 ]; then

  echo "Internet: Connected" >> performance.log

else

  echo "Internet: Disconnected" >> performance.log

fi


echo "RAM Usages :" >> performance.log
```

```
free -h | grep "Mem" >> performance.log


echo "Swap Usages :" >> performance.log

free -h | grep "Swap" >> performance.log


echo "Disk Usages :" >> performance.log

df -h >> performance.log

echo ""
```

=====-

Automate the script on the remote server:

```
crontab -e

15 * * * * ~/performance_checker.sh
```

Automate the downloading of the data to your local system by adding ssh password in the command.

Note: sshpass must be installed / > sudo apt install sshpass

```
Crontab -e

25 * * * * sshpass ‘remote-user-password’ scp root@server-ip:file-location local-file-location-for-storage-remote-file

25 * * * * sshpass ‘Asdgg!22’ scp root@192.168.0.126:~/performance.log ~/remote_data/localdir/
```

Syntax to transfer from remote system to local system

```
cp user@ip:/path/to/file /path/to/destination
```

Syntax to transfer file from local system to remote system

```
scp /path/to/file user@ip:/path/to/destination
```