Bash Scripting Questions

Welcome to the Bash Scripting Lab Questions File! This document contains a collection of practical exercises and challenges designed to help you develop and enhance your Bash scripting skills.

01 - Create a Bash script that takes two numbers as input from the user and prints their sum.

Answer Here

```
#!/bin/bash

# Prompt the user to enter the first number
echo "Enter the first number:"
read num1

# Prompt the user to enter the second number
echo "Enter the second number:"
read num2

# Calculate the sum
sum=$((num1 + num2))

# Print the sum
echo "The sum of $num1 and $num2 is: $sum"
```

02 - Write a Bash script to create a new directory named "backup" if it doesn't already exist.

Answer Here

```
#!/bin/bash

backup_dir="backup"

# Check if the directory exists
```

```
if [ ! -d "$backup_dir" ];
then
    # If it doesn't exist, create it
    mkdir "$backup_dir"
    echo "Backup directory created."
else
    echo "Backup directory already exists."
fi
```

03 - Create a Bash script that finds and prints all files larger than 1MB in the current directory. (Print any messages)

Answer Here

```
#!/bin/bash
echo "Finding files larger than 1MB in the current directory:"
find . -type f -size +1M -exec ls -lh {} \;
```

04 - Write a Bash script that prompts the user to enter a directory name and then prints the number of files in that directory.

Answer Here

```
#!/bin/bash
echo "Enter the directory name:"
read directory

# Check if the directory exists
if [ -d "$directory" ];
then
    # Count the number of files in the directory and print the count
    num_files=$(find "$directory" -maxdepth 1 -type f | wc -l)
    echo "Number of files in '$directory': $num_files"
else
    echo "Directory '$directory' does not exist."
fi
```

05 - Develop a Bash script that takes a string as input and prints the number of words in the string.

Answer Here

```
#!/bin/bash

# Prompt the user to enter a string
echo "Enter a string:"
read input_string

# Count the number of words in the string
word_count=$(echo "$input_string" | wc -w)

# Print the number of words
echo "Number of words in the string: $word_count"
```

06 - Develop a Bash script that takes a filename as input and counts the number of occurrences of a specific word in the file.

Answer Here

```
#!/bin/bash

# Prompt the user to enter a filename
echo "Enter the filename:"
read filename

# Check if the file exists
if [ ! -f "$filename" ];
then
    echo "File '$filename' not found."
    exit 1
fi

# Prompt the user to enter a word to search for
echo "Enter the word to count:"
read word
```

```
# Count the number of occurrences of the word in the file
word_count=$(grep -o "\<$word\>" "$filename" | wc -l)

# Print the number of occurrences
echo "Number of occurrences of '$word' in '$filename': $word_count"
```

07 - Write a Bash script to find and print all symbolic links in the / directory

Answer Here

```
#!/bin/bash

echo "Symbolic links in the / directory:"

find / -type l -print
```

08 - Create a Bash script that takes a directory name as input and prints the total size of all files in that directory.

Answer Here

```
#!/bin/bash

# Prompt the user to enter a directory name
echo "Enter the directory name:"
read dir

# Check if the directory exists
if [ ! -d "$dir" ];
then
    echo "Directory '$dir' not found."
    exit 1
fi
# Calculate the total size of all files in the directory
total_size=$(du -sh "$dir" | cut -f1)
```

```
# Print the total size
echo "Total size of all files in '$dir': $total_size"
```

09 - Write a Bash script that takes two filenames as input and compares their contents. If they are identical, print "Files are identical"; otherwise, print "Files are different."

Answer Here

```
#!/bin/bash

# Prompt the user to enter the first filename
echo "Enter the first filename:"
read filename1
# Prompt the user to enter the second filename
echo "Enter the second filename:"
read filename2

# Check if the files exist
if [ ! -f "$filename1" ] || [ ! -f "$filename2" ];
then
    echo "One or both files do not exist."
    exit 1
fi
# Compare the contents of the files
if cmp -s "$filename1" "$filename2";
then
    echo "Files are identical."
else
    echo "Files are different."
fi
```

10 - Write a Bash script that takes a filename as input and prints the file's permissions in symbolic notation (e.g., -rwxr-xr--).

Answer Here

```
#!/bin/bash

# Prompt the user to enter a filename
echo "Enter the filename:"
read filename

# Check if the file exists
if [ ! -f "$filename" ];
then
    echo "File '$filename' not found."
    exit 1
fi

# Get the file's permissions in symbolic notation
permissions=$(stat -c "%A" "$filename")
# Print the file's permissions
echo "Permissions of '$filename': $permissions"
```

11 - Write a Bash script that takes a list of filenames as arguments and concatenates their contents into a single file.

Answer Here

```
#!/bin/bash

# Check if there are no arguments
if [ $# -eq 0 ];
then
    echo "Usage: $0 <filename1> <filename2> ..."
    exit 1
fi

# Name of the concatenated file
output_file="concatenated_file.txt"

# Concatenate the contents of all files into the output file
cat "$@" > "$output_file"
```

```
echo "Contents of the following files have been concatenated into '$output_file':"
echo "$@"
```

12 - Write a Bash script that prompts the user to enter a directory name and then creates subdirectories named "Monday" to "Friday" within that directory.

Answer Here

```
#!/bin/bash

# Prompt the user to enter a directory name
echo "Enter the directory name:"
read directory_name

# Check if the directory already exists
if [ -d "$directory_name" ];
then
    echo "Directory '$directory_name' already exists."
else
    # Create the directory
    mkdir "$directory_name"
    echo "Directory '$directory_name' created."

    # Create subdirectories for each day of the week
    for day in Monday Tuesday Wednesday Thursday Friday;
    do
        mkdir "$directory_name/$day"
        echo "Subdirectory '$day' created."
    done
fi
```

13 - Develop a Bash script that takes a number as input from the user and prints "Positive" if the number is greater than zero, "Negative" if it's less than zero, and "Zero" if it's equal to zero

Answer Here

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number:"
read number

# Check if the number is greater than zero
if ((number > 0));
then
    echo "Positive"
# Check if the number is less than zero
elif ((number < 0));
then
    echo "Negative"
# If neither of the above conditions is met, the number is zero
else
    echo "Zero"
fi
```

14 - Develop a Bash script that takes a filename as input and checks if the file is executable. If it is, print "File is executable"; otherwise, print "File is not executable."

Answer Here

```
#!/bin/bash

# Prompt the user to enter a filename
echo "Enter the filename:"
read filename

# Check if the file exists
if [ ! -e "$filename" ];
then
    echo "File '$filename' does not exist."
    exit 1
```

```
fi

# Check if the file is executable
if [ -x "$filename" ];
then
    echo "File '$filename' is executable."
else
    echo "File '$filename' is not executable."
fi
```

15 - Write a Bash script that prompts the user for their age and prints "You are an adult" if their age is 18 or older; otherwise, print "You are a minor."

Answer Here

```
#!/bin/bash
# Prompt the user to enter their age
echo "Enter your age:"
read age

# Check if the age is greater than or equal to 18
if [ "$age" -ge 18 ];
then
    echo "You are an adult."
else
    echo "You are a minor."
fi
```

16 - Create a Bash script that checks if a given username exists in the system. If it does, display "User exists"; otherwise, display "User does not exist."

Answer Here

```
#!/bin/bash
# Prompt the user to enter a username
echo "Enter the username:"
read username

# Check if the user exists
if id "$username" &>/dev/null;
then
    echo "User '$username' exists."
else
    echo "User '$username' does not exist."
fi
```

17 - Develop a Bash script that checks if the SSH service is running on the system. If it's running, display "SSH service is running"; otherwise, start the service and print "SSH service started."

Answer Here

```
#!/bin/bash

# Check if the SSH service is running
if systemctl is-active --quiet sshd;
then
    echo "SSH service is running."
else
    # Start the SSH service
    sudo systemctl start sshd
    echo "SSH service started."
fi
```

18 - Write a Bash script that checks if a package named "nginx" is installed on the system. If it's installed, print "Nginx is installed"; otherwise, install the package and print "Nginx installed successfully."

Answer Here

```
#!/bin/bash

# Check if nginx is installed
if rpm -q nginx &>/dev/null; then
    echo "Nginx is installed."
else
    # Install nginx
    sudo dnf install nginx -y
    echo "Nginx installed successfully."
fi
```

19 - Create a Bash script that checks the disk usage of the root filesystem. If the disk usage exceeds 90%, print "Disk usage high"; otherwise, print "Disk usage normal."

Answer Here

```
#!/bin/bash

# Set the threshold for disk usage
threshold=90
# Get the current disk usage of the root filesystem
disk_usage=$(df -h / | awk 'NR==2 {print $5}' | cut -d'%' -f1)

# Check if the disk usage exceeds the threshold
if [ "$disk_usage" -gt "$threshold" ];
then
    echo "Disk usage high"
else
    echo "Disk usage normal"
fi
```

20 - Develop a Bash script that checks if a user named "admin" exists on the system. If the user exists, print "User exists"; otherwise, create the user and print "User created successfully."

Answer Here

```
#!/bin/bash

# Check if the user "admin" exists
if id "admin" &>/dev/null;
then
    echo "User 'admin' exists."
else
    # Create the user "admin"
    sudo useradd -m admin
    echo "User 'admin' created successfully."
fi
```

21 - Write a Bash script that checks if the "sudo" command is configured for the current user. If it's configured, print "User has sudo access"; otherwise, print "User does not have sudo access."

Answer Here

```
#!/bin/bash

# Check if the current user has sudo access
if sudo -n true 2>/dev/null;
then
    echo "User has sudo access."
else
    echo "User does not have sudo access."
fi
```

22 - Develop a Bash script that checks the system load average. If the load average exceeds a predefined threshold (e.g., 2.0), print a warning message; otherwise, print "System load normal."

Answer Here

```
#!/bin/bash

# Define the threshold for the load average
threshold=2.0

# Get the current 1-minute load average
load_average=$(uptime | awk -F 'load average: ' '{print $2}' | cut -d, -f1)

# Check if the load average exceeds the threshold
if [ $(echo "$load_average > $threshold" | bc) -eq 1 ];
then
    echo "Warning: System load average ($load_average) exceeds threshold ($threshold)."
else
    echo "System load normal."
fi
```

23 - Develop a Bash script that checks the available disk space in the "/home" directory. If the available space is less than 1GB, print a warning message; otherwise, print "Disk space is sufficient."

Answer Here

```
#!/bin/bash

# Define the threshold for available disk space (in GB)
threshold_gb=1
# Get available disk space in /home directory (in GB)
available_space_gb=$(df -BG /home | awk 'NR==2 {print $4}' | sed 's/G//')

# Check if available space is less than the threshold
if [ "$available_space_gb" -lt "$threshold_gb" ];
then
    echo "Warning: Available disk space in /home is less than 1GB."
else
    echo "Disk space is sufficient."
fi
```

24 - Write a Bash script that checks if a user named "john" is logged in to the system. If the user is logged in, display "User is logged in"; otherwise, print "User is not logged in."

Answer Here

```
#!/bin/bash

# Check if the user "john" is logged in
if who | grep -q "\<john\>";
then
    echo "User is logged in."
else
    echo "User is not logged in."
fi
```

25 - Create a Bash script that checks the system uptime. If the uptime is less than 1 hour, print "System recently started"; otherwise, print "System has been running for a while."

Answer Here

```
#!/bin/bash

# Get the system uptime in seconds
uptime_seconds=$(cut -d' ' -f1 /proc/uptime)

# Calculate the uptime in hours
uptime_hours=$(bc <<< "scale=2; $uptime_seconds / 3600")

# Check if the uptime is less than 1 hour
if (( $(bc <<< "$uptime_hours < 1") ));
then
    echo "System recently started"
else
    echo "System has been running for a while"
fi
```

26 - Develop a Bash script that checks if a network interface (e.g., "enp0s3") is up and running. If it's up, print "Interface is up"; otherwise, print "Interface is down."

Answer Here

```
#!/bin/bash

# Define the network interface to check
interface="enp0s3"

# Check if the interface is up
if ip link show "$interface" &> /dev/null;
then
    echo "Interface is up"
else
    echo "Interface is down"
fi
```

27 - Write a Bash script that checks the current user's home directory. If it's located in "/home", print "Home directory standard"; if it's located elsewhere, print "Home directory non-standard."

Answer Here

```
#!/bin/bash

# Get the current user's home directory
home_directory=$(eval echo "~")

# Check if the home directory is located in /home
if [[ "$home_directory" == "/home"* ]];
then
    echo "Home directory standard"
else
    echo "Home directory non-standard"
fi
```

Revision #2
Created 2026-03-19 10:13:55 UTC by Admin
Updated 2026-03-19 10:25:57 UTC by Admin