# ICS 2305: Systems Programming Assignment 2: SCT211-0221/2018: Peter Kibuchi

|  |  |
|---|---|
| Created | @October 30, 2023 9:40 AM |
| Last Updated | @October 31, 2023 8:55 PM |
| Tags | 3.1 Assignments |

💡 To run the scripts, first make them executable, i.e., `chmod +x script-name.sh` then run them, i.e., `./script-name.sh`.

1. **Write the shell script that reads 5 numbers from the keyboard and print their sum.**

```
#!/bin/bash

sum=0  # Initialize the sum to 0

echo "Enter 5 numbers:"

# Loop to read 5 numbers
for ((i = 1; i <= 5; i++)); do
    read -p "Enter number $i: " num
    sum=$((sum + num))
done

echo "The sum of the 5 numbers is: $sum"
```

2. **Write a Shell script that reads the text in a text file called `JUJAyetu.txt` and outputs number of characters (exclusive of white spaces) in that text file.**

```
#!/bin/bash

# Check if the file exists
if [ -e "JUJAyetu.txt" ]; then
    # Use tr to remove whitespace characters and count the remaining characters
    char_count=$(cat JUJAyetu.txt | tr -d '[:space:]' | wc -m)

    echo "Number of characters (excluding white spaces) in JUJAyetu.txt: $char_count"
else
    echo "File 'JUJAyetu.txt' does not exist."
fi
```

The script checks if the file `JUJAyetu.txt` exists, and if it does, uses `tr` to remove whitespace characters. It then uses `wc -m` to count the remaining characters and print the count. If the file doesn't exist, it will display an error message.

3. **UNIX allows users to be in groups based on their access levels. Write a shell script that given a person's uid, tells you how many times that person has logged in to the system.**

To accomplish this task we can use the `last` command to retrieve the login history and then filter the results based on the provided UID:

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Usage: $0 <UID>"
    exit 1
fi

uid="$1"

# Use the last command to retrieve login history and filter it by UID
login_count=$(last | grep " $uid " | wc -l)

echo "User with UID $uid has logged in $login_count times."
```

We can then run the script, providing the UID as an argument:

```
./question-3.sh <UID>
```

The script will use `last` to retrieve the login history and then count the number of times the provided UID appears in the output. It will then display the login count for that user.

4. **Write a Shell Script that counts the number of empty folders in a directory (with size=0) and outputs the list of those folders and the time created.**

   To achieve this, we can use the `find` command along with `stat` to retrieve information about each folder:

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Usage: $0 <directory>"
    exit 1
fi

directory="$1"

if [ ! -d "$directory" ]; then
    echo "Error: '$directory' is not a valid directory."
    exit 1
fi

empty_folders=0

find "$directory" -type d -empty | while read -r folder; do
    creation_time=$(stat -c "%y" "$folder")
    echo "Empty folder: $folder, Created on: $creation_time"
    empty_folders=$((empty_folders + 1))
done
```

   We can run the script, providing the directory as an argument:

```
./question-4.sh /path/to/directory
```

   The script will find and list empty folders in the specified directory, along with their creation times, as well as display the total count of empty folders.

5. **Write a script that will convert all of the `.bmp` files in a given directory into `.jpeg` files. The user should be prompted to enter the file path. The output after conversion should give us the number of images converted.**

```
#!/bin/bash

read -p "Enter the directory path containing BMP files: " directory

if [ ! -d "$directory" ]; then
    echo "Error: '$directory' is not a valid directory."
    exit 1
fi

# Check if ImageMagick is installed
if ! command -v convert &> /dev/null; then
    echo "Error: ImageMagick is not installed. Please install it to use this script."
    exit 1
fi

# Convert BMP files to JPEG files
bmp_files=$(find "$directory" -type f -name "*.bmp")
converted_count=0

for bmp_file in $bmp_files; do
    jpeg_file="${bmp_file%.bmp}.jpeg"
    convert "$bmp_file" "$jpeg_file"
    converted_count=$((converted_count + 1))
done

echo "Converted $converted_count BMP files to JPEG."
```

   We need to install ImageMagick before we can run this script:

```
sudo apt install imagemagick
```

   When we run the script will prompt us to enter the directory path containing the `.bmp` files we want to convert. It will then use ImageMagick's `convert` command to perform the conversion and display the number of images converted.

6. **`Nmap` is a network mapper software that can show data about network traffic such as showing the IP address of all devices your machine is connected to. This can be done using a bash script as well. Write a bash Script that scan network for hosts attached to an IP address. The script should show if the host is up.**

   We can use the `nmap` command to scan a network for hosts attached to a specific IP address and check if the hosts are up:

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Usage: $0 <IP_address>"
    exit 1
fi

ip_address="$1"
```

```
# Use nmap to scan the specified IP address
nmap -sn "$ip_address" | grep "Host is up" | cut -d " " -f 2
```

Nmap needs to be installed before we run the script:

```
sudo apt install nmap
```

The script uses `nmap` with the `-sn` option to perform a simple host discovery scan, and then it filters the output to display the hosts that are up.

7. **Department Chairman of Computing has heard about your Unix shell scripting Prowess in creating mail-merge services. He wants your services to invite students to a 3rd Year Projects webinar. The department wants to send the invitation letter below to a webinar addressed to each student personally:**

> **"Inviting you as our computing student to our 3rd year Projects on innovation & incubation scheduled Monday, 23rd October from 10:00 A.M. The Zoom link for joining in on Friday is as given below. We shall also broadcast the webinar on our youtube channel. Youtube link: https://youtu.be/ONVTA7LKMIs**

**The chairman has a text file that contains the names of the persons and the email of persons they want to send to. Implement this mail merge in bash that works with all popular browsers.**

To accomplish this task we can use a combination of shell scripting and the command-line email client `mail`.

First, we'll create a text file containing the invitation letter template, i.e., invitation_template.txt:

```
Dear %NAME%,

Inviting you as our computing student to our 3rd Year Projects on innovation & incubation scheduled Monday, 23rd October from 10:00 A.M

We shall also broadcast the webinar on our YouTube channel.

Zoom link: https://zoom.us/j/zoom_link
YouTube link: https://youtu.be/ONVTA7LKMIs

Best regards,
Chairman, Computing Department
```

Next, we create another text file, i.e., student_list.txt, that contains the names and email addresses of the students. The format should be one student per line, with the name and email separated by a space:

```
John Doe john.doe@example.com
Jane Smith jane.smith@example.com
```

Now, we can create a Bash script to perform the mail merge:

```
#!/bin/bash

template="invitation_template.txt"
student_list="student_list.txt"

if [ ! -f "$template" ]; then
    echo "Error: Template file not found."
    exit 1
fi

if [ ! -f "$student_list" ]; then
    echo "Error: Student list file not found."
    exit 1
fi

while read -r line; do
    name=$(echo "$line" | cut -d ' ' -f 1)
    email=$(echo "$line" | cut -d ' ' -f 2)

    # Read the template and replace placeholders with student-specific information
    invitation_text=$(sed "s/%NAME%/$name/g" "$template")

    # Send the invitation email
    echo "$invitation_text" | mail -s "Invitation to 3rd Year Projects Webinar" "$email"

    echo "Invitation sent to $name ($email)"
done < "$student_list"
```

We have to install `mailutils` before running the script:

```
sudo apt install mailutils
```

The script reads the student list, customizes the invitation letter for each student, and sends the emails using the `mail` command.