**JOSPHAT WAWERU THUMI**

**SCT211-0003/2022**

**SYSTEMS PROGRAMMING**

**CAT 2**

**1.** **Write the shell script that reads any 6 numbers from the keyboard and print their sum, if non numbers are keyed from the keyboard , it throws an alert to key in numbers only**

*#!/bin/bash*

*sum=0*

*count=0*

*echo "Enter 6 numbers:"*

*while [ $count -lt 6 ]; do*

*read -p "Input number at index $((count + 1)): " number*

*# Check if the input is a valid number*

*if [[ $number =~ ^-?[0-9]+$ ]]; then*

*sum=$((sum + number)) # Add the number to the sum*

*count=$((count + 1)) # Increment the count of valid numbers*

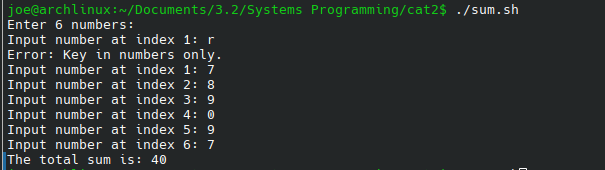
*else*

*echo "Error: Key in numbers only."*

*fi*

*done*

*echo "The total sum is: $sum"*

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

*#!/bin/bash*

*# File to read*

*file="Angukanayo.txt"*

*# Check if the file exists*

*if [[ -f "$file" ]]; then*

*# Count characters including spaces using wc -m*

*characters=$(wc -m < "$file")*

*echo "The file '$file' has $characters characters (including spaces)."*

*else*

*echo "Error: $file does not exist."*

*fi*

**

**2(b) write the script to report if in the text file called Angukanayo.txt has a string Mpangale and the line number on which it occurs within file.**

*#!/bin/bash*

*# File to read*

*file="Angukanayo.txt"*

*# String to search for*

*search\_string="Mpangale"*

*# Check if the file exists*

*if [[ -f "$file" ]]; then*

*# Search for the string and display line numbers*

*grep -n "$search\_string" "$file" > /dev/null 2>&1*

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

*echo "The string '$search\_string' is found in the following lines of '$file':"*

*grep -n "$search\_string" "$file"*

*else*

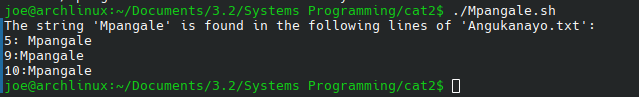
*echo "The string '$search\_string' is not found in the file '$file'."*

*fi*

*else*

*echo "Error: $file does not exist."*

*fi*

**

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

*#!/bin/bash*

*# Prompt for the user's UID*

*read -p "Enter the user's UID: " uid*

*# Get the username associated with the UID*

*username=$(getent passwd "$uid" | cut -d: -f1)*

*# Check if the UID exists in the system*

*if [[ -z "$username" ]]; then*

*echo "Error: No user found with UID $uid."*

*exit 1*

*fi*

*# Use the `who` command to find login information*

*login\_info=$(who | grep -w "$username")*

*if [[ -z "$login\_info" ]]; then*

*echo "User '$username' (UID: $uid) is not currently logged in."*

*else*

*# Count the number of login sessions*

*login\_count=$(echo "$login\_info" | wc -l)*

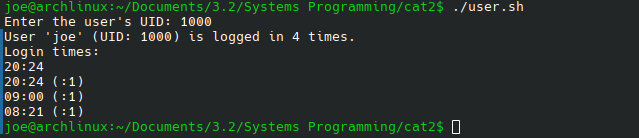
*echo "User '$username' (UID: $uid) is logged in $login\_count times."*

*echo "Login times:"*

*# Display the login times*

*echo "$login\_info" | awk '{print $4, $5}' # Extracts login time and date*

*fi*

****

**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**

*#!/bin/bash*

*# Prompt for the directory to search*

*read -p "Enter the directory path: " dir*

*# Check if the provided directory exists*

*if [[ ! -d "$dir" ]]; then*

*echo "Error: The directory '$dir' does not exist."*

*exit 1*

*fi*

*# Find all empty directories and output their names and creation times*

*empty\_folders=$(find "$dir" -type d -empty)*

*# Check if there are empty directories*

*if [[ -z "$empty\_folders" ]]; then*

*echo "No empty directories found in '$dir'."*

*exit 0*

*fi*

*# Count empty directories*

*count=$(echo "$empty\_folders" | wc -l)*

*# Display the number of empty directories*

*echo "Number of empty directories: $count"*

*# Output the empty directories and their creation time*

*echo "List of empty directories and their creation times:"*

*for folder in $empty\_folders; do*

*# Get the creation time of the directory (using stat command)*

*creation\_time=$(stat --format='%w' "$folder")*

*# If creation time is unavailable, use the last modification time*

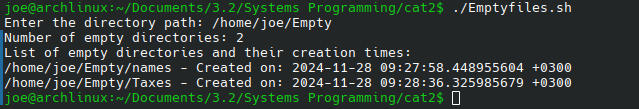
*if [[ "$creation\_time" == "-" ]]; then*

*creation\_time=$(stat --format='%y' "$folder")*

*fi*

*echo "$folder - Created on: $creation\_time"*

*done*



***5.Write a script that will convert all of the .jpg files in a given directory into .tiff 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*

*# Prompt the user to enter the directory path*

*read -p "Enter the directory path: " dir*

*# Check if the directory exists*

*if [[ ! -d "$dir" ]]; then*

*echo "Error: The directory '$dir' does not exist."*

*exit 1*

*fi*

*# Initialize a counter for the number of converted files*

*converted\_count=0*

*# Find all .jpg files in the directory*

*for img in "$dir"/\*.jpg; do*

*# Check if the file exists (in case there are no .jpg files)*

*if [[ -f "$img" ]]; then*

*# Construct the output file name by changing the extension to .tiff*

*output="${img%.jpg}.tiff"*

*# Convert the image from .jpg to .tiff using the magick command (ImageMagick v7)*

*magick "$img" "$output"*

*# Check if conversion was successful*

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

*# Increment the counter if the conversion was successful*

*((converted\_count++))*

*else*

*echo "Error: Failed to convert $img."*

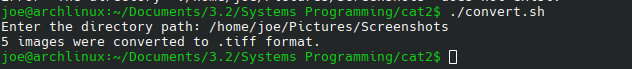
*fi*

*fi*

*done*

*# Output the number of images converted*

*echo "$converted\_count images were converted to .tiff format."*

**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 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 indicating Yes if up.**

*#!/bin/bash*

*# Prompt the user to enter the network IP address or subnet*

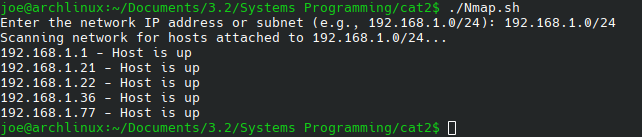
*read -p "Enter the network IP address or subnet (e.g., 192.168.1.0/24): " subnet*

*# Perform a ping sweep on the given subnet to check if hosts are up*

*echo "Scanning network for hosts attached to $subnet..."*

*# Use nmap to perform the scan and identify hosts*

*nmap -sn "$subnet" | grep -B 2 "Host is up" | grep "Nmap scan report" | awk '{print $5, "- Host is up"}'*

 **7. Department Chairman of Computing has heard about your Unix shell scripting Prowess in creating mail merge service, he wants your services to invite students to a webinar hosted by JhubAfrica . The department wants to send the invitation letter below to a webinar addressed to each student personally.**

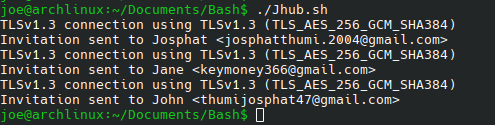
**“Inviting the you as our computing student to our 21st Open Webinar on skills optimization by jhubAfrica scheduled for Friday, 29th November from 5:00 P.M. The Zoom link for joining in on Friday is as given below. We shall also broadcast the webinar on our youtube channel. Youtube link:**

[**https://youtu.be/ONVTA7LKMIs**](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 linux bash and**

**should work with all popular browsers .**

*#!/bin/bash   
  
# Check if required files exist   
if [[ ! -f "students.txt" || ! -f "invitation.txt" ]]; then   
 echo "Required files (students.txt and invitation.txt) not found!"   
 exit 1   
fi   
  
# Loop through each line in the students list   
while IFS=" " read -r name email; do   
 # Create a personalized email body by replacing [NAME] in the invitation templ>   
 email\_body=$(sed "s/\[NAME\]/$name/" invitation.txt)   
  
 # Send the email using Mutt   
 echo "$email\_body" | mutt -s "Invitation to 21st Open Webinar" -e "set from=jo>   
  
 echo "Invitation sent to $name <$email>"   
done < students.txt*****

**8. Write a bash that checks the available disk spaces in your drive C and alert the**

**“the space available is ----Mb”**

*#!/bin/bash*

*# Specify the drive or partition you want to check (e.g., /dev/sda1 or /)*

*drive="/"*

*# Get the available space on the specified partition (in MB)*

*available\_space=$(df -m $drive | awk 'NR==2 {print $4}')*

*# Output the available space*

*echo "The space available is ${available\_space}MB"*



**9.(a) using the wget command , download the following two files Download an example TXT file**

**from**

[**https://github.com/yesinteractive/dadjokes/blob/master/controllers/jokes.txt**](https://github.com/yesinteractive/dadjokes/blob/master/controllers/jokes.txt)

**Download the example JSON file from**

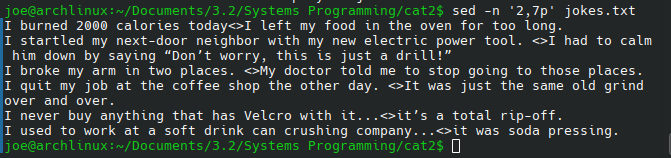
**[https://gist.github.com/rominirani/8235702#file-employees-json](https://gist.github.com/rominirani/8235702" \l "file-employees-json)**

*wget https://raw.githubusercontent.com/yesinteractive/dadjokes/master/controllers/jokes.txt*

*wget https://gist.githubusercontent.com/rominirani/8235702/raw/9db4a51c8f9c76fa2e9d3f3794c07a1d2cc16d93/employees.json*

**(b) Write a shell command to list and show lines 2 to 7 of the file jokes.txt**

*sed -n '2,7p' jokes.txt*

**

**C. Create an MD5 checksum file MD5SUM for the TXT and JSON files above using a shell script**

*#!/bin/bash*

*# Create the MD5 checksum file for jokes.txt and employees.json*

*md5sum jokes.txt > MD5SUM*

*md5sum employees.json >> MD5SUM*

*# Display the checksum file content*

*echo "MD5 checksum file 'MD5SUM' has been created:"*

*cat MD5SUM*

**D. Write a shell command to archive (tar gzipped archive ) the two file TXT and**

**JSON as Johnnie.tar.gz**

*tar -czf Johnnie.tar.gz jokes.txt employees.json*

**(e) Write the shell command that can delete the two files from the downloaded location in (a)**

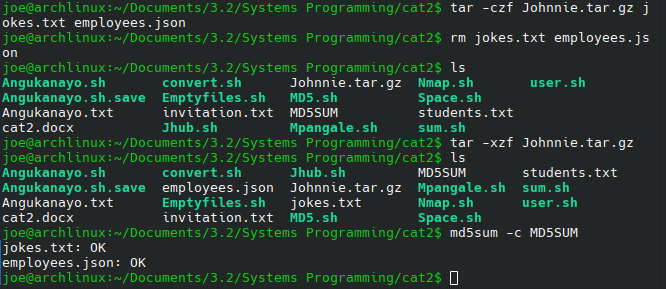
*rm jokes.txt employees.json*

**(f) Write the shell command to recover or restore the above two files:**

*tar -xzf Johnnie.tar.gz*

**(g) Write a shell command to compare the MD5SUM of the recovered files against the deleted ones:**

*md5sum -c MD5SUM*

**