Operating System Lab

Name: Tonmoy Biswas

Roll No: 002110501133

Class: BCSE 3rd Year 1st Sem

Section: A3

Assignment No: 1

1. Write a shell script that has 2 user created variables, uv1 and uv2. Ask for the values of the variables from the user and take in any values (real/integer/character) for the 2 variables. Test the program for different types of uv1 and uv2.

(a) Print them as:

(i) value of uv1 followed by value of uv2 separated by a comma and

(ii) value of uv2 followed by value of uv1 separated by the word “and”.

(b) Print the variables in reverse order [If uv1 is 1234, then output should be 4321]

**Solution**: 2 numbers are taken as input. Using for loop, the string is reversed; though “**rev**” function could be used to reverse the string.

**Code**:

#!/bin/bash

echo "Enter the first variable"

read uv1

echo "Enter the second variable"

read uv2

echo "$uv1,$uv2"

echo "$uv2 and $uv1"

read -p "Enter the value:" str

# read - p "Enter string:" string

len=${#str}

for ((i = $len - 1; i >= 0; i--))

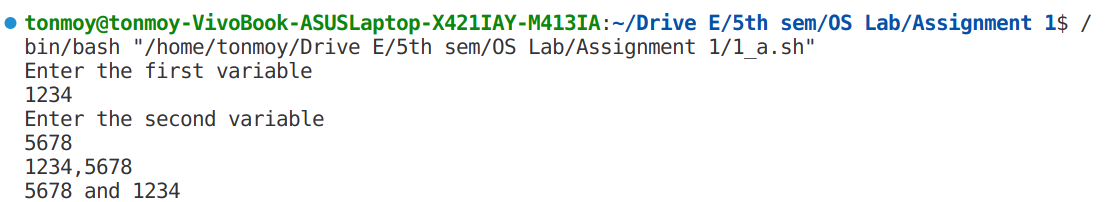
do

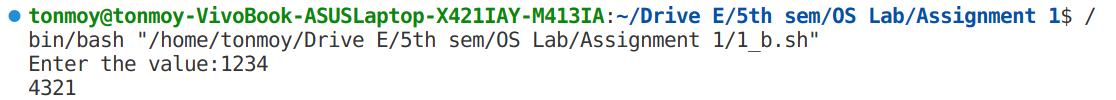
reverse="$reverse${str:$i:1}"

done

echo $reverse

**Output:**





2. Write a shell script to count the number of lines in a file. Test if the file is present. If not, create and write.

**Solution**: Presence of file in the directory is checked using **–f options** inside the if condition. If the file is present, the number of lines can be found using **“wc -l”** command. If the file is not present, the file must be created and input is taken using cat command.

**Code:**

**#!/bin/bash**

read -p "Enter the file name:" file

if ! [[ -f $file ]]

then

touch $file

echo "$file created, write something in file and do Ctrl+d to exit"

cat > $file

fi

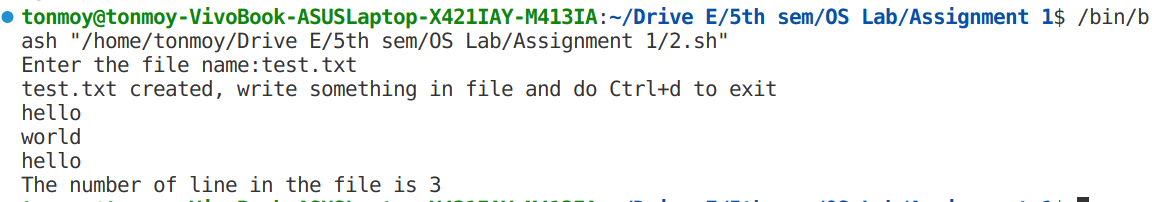
# line\_no=`wc --lines < $file`

line\_no=$(wc -l < $file)

line\_no=$(($line\_no + 1))

echo "The number of line in the file is $line\_no"

**Output**:



3. Write a shell script that counts the number of ordinary files (not directories) in the current working directory and its sub-directories. Repeat the count of files including the subdirectories that the current working directory has.

**Solution**: This can be done by recursively calling a function for all the subdirectories. Two functions were generated, one counted only the files, and another counted the number of files as well as directories.

**Code:**

#!/bin/bash

count\_files()

{

if [ $# -eq 0 ]

then

echo 0

else

PATH="$1"

no\_of\_files=0

for file in "$PATH"/\*

do

if [ -d "$file" ]

then

rec\_return=$(count\_files "$file/")

no\_of\_files=$(($no\_of\_files + $rec\_return))

else

no\_of\_files=$(($no\_of\_files + 1))

fi

done

echo $no\_of\_files

fi

}

count\_files\_with\_dir()

{

if [ $# -eq 0 ]

then

echo 0

else

PATH="$1"

no\_of\_files=0

for file in "$PATH"/\*

do

no\_of\_files=$(($no\_of\_files + 1))

if [ -d "$file" ]

then

rec\_return=$(count\_files\_with\_dir "$file/")

no\_of\_files=$(($no\_of\_files + $rec\_return))

fi

done

echo $no\_of\_files

fi

}

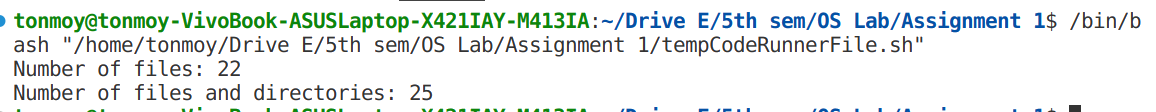
ans=$(count\_files ".")

echo "Number of files: $ans"

ans=$(count\_files\_with\_dir ".")

echo "Number of files and directories: $ans"

**Output**:



4. Write a shell program to duplicate the UNIX **rm** command with the following features:

a. Instead of deleting the files, it will move them to a **my-deleted-files** directory. If the file already exists in the **my-deleted-files** directory, then the existing file (in the **my-deleted-files**) will have the version number zero (0) appended to it and the newly deleted file will have version number one (1) appended to it. Go on incrementing the version nos., if required.

b. The command will have a switch -c that will clear the entire **my-deleted-files** directory after asking for confirmation.

**Solution**: The count of the next file to be printed, is done by by for loop. on using –c, the whole “my-deleted-files” folder is deleted.

**Code**:

#!/bin/bash

deleted\_dir="my-deleted-files"

# Function to move files to the my-deleted-files directory

move\_to\_deleted() {

file="$1"

# base\_name

# extension

version=0

# Extract the base name and extension of the file

base\_name="${file%.\*}"

extension="${file##\*.}"

if [ -e "$deleted\_dir/$base\_name.$extension" ]

then

mv "$deleted\_dir/$file" "$deleted\_dir/$base\_name$version.$extension"

fi

# Check if the file already exists in the my-deleted-files directory

while [ -e "$deleted\_dir/$base\_name$version.$extension" ]

do

((version++))

done

if [ $version -eq 0 ]

then

# Move the file without any version number

mv "$file" "$deleted\_dir/$file"

else

# Move the file to the my-deleted-files directory with the version number

mv "$file" "$deleted\_dir/$base\_name$version.$extension"

fi

}

# Function to clear the entire my-deleted-files directory

clear\_deleted\_directory() {

read -p "Are you sure you want to clear the my-deleted-files directory? (y/n): " confirm

if [ "$confirm" = "y" ]

then

rm -r "$deleted\_dir"/\*

echo "my-deleted-files directory cleared."

else

echo "Operation canceled."

fi

}

# Check if the my-deleted-files directory exists, and create it if not

if [ ! -d "$deleted\_dir" ]

then

mkdir "$deleted\_dir"

fi

# args\_no=$#

if [ $1 == "-c" ]

then

clear\_deleted\_directory

else

# Loop through the remaining arguments (files to be deleted)

for file in "$@"

do

if [ -e "$file" ]

then

move\_to\_deleted "$file"

echo "Moved '$file' to $deleted\_dir."

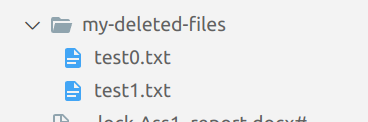
else

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

fi

done

fi**Output**



5. Write a script called birthday\_match.sh that takes two birthdays of the form DD/MM/YYYY (e.g., 15/05/2000) and returns whether there is a match if the two people were born on the same day of the week (e.g., Friday). And then find out the age/s in years/months/days.

**Solution**: ‘date’ commands can be used to retrieve the days, and can also detect the dates that are invalid. A function was created to generate the age from a given date.

**Code**:

#!/bin/bash

# Function to calculate the age

calculate\_age() {

bd\_date="$1"

current\_date="$(date +'%d/%m/%Y')"

# echo "Current data $current\_date"

bd\_year="${bd\_date##\*/}"

current\_year="${current\_date##\*/}"

# echo "Year $bd\_year"

# echo "Year $current\_year"

bd\_month="${bd\_date%/\*}"

current\_month="${current\_date%/\*}"

# echo "month $bd\_month"

# echo "month $current\_month"

bd\_day="${bd\_month%/\*}"

current\_day="${current\_month%/\*}"

# echo "day $bd\_day"

# echo "day $current\_day"

bd\_month="${bd\_month#\*/}"

current\_month="${current\_month#\*/}"

# echo "month $bd\_month"

# echo "month $current\_month"

# temp=$bd\_date

# bd\_date=$bd\_month

# bd\_month=$temp

# temp=$current\_day

# current\_day=$current\_month

# current\_month=$temp

age\_years=$(expr $current\_year - $bd\_year )

# echo "Age year $age\_years"

age\_months=$(expr $current\_month - $bd\_month )

age\_days=$(expr $current\_day - $bd\_day )

# echo "Age day $age\_days"

# Adjust for negative months or days

if [ "$age\_days" -lt 0 ]

then

age\_days=$((age\_days + 30))

age\_months=$((age\_months - 1))

fi

if [ "$age\_months" -lt 0 ]

then

age\_months=$((age\_months + 12))

age\_years=$((age\_years - 1))

fi

echo "$age\_years years, $age\_months months, $age\_days days"

}

# Function to check if two dates fall on the same day of the week

check\_same\_day\_of\_week() {

date1=$(echo "$1" | awk -F'/' '{print $2"/"$1"/"$3}')

date2=$(echo "$2" | awk -F'/' '{print $2"/"$1"/"$3}')

day1="$(date -d "$date1" +'%A')"

day2="$(date -d "$date2" +'%A')"

if [ "$day1" == "$day2" ]

then

echo "Same day of the week ($day1)"

else

echo "Different days of the week ($day1 and $day2)"

fi

}

read -p "Enter the first date of birth: " birthday1

read -p "Enter the second date of birth: " birthday2

# Check if the days of the week match and calculate age

check\_same\_day\_of\_week "$birthday1" "$birthday2"

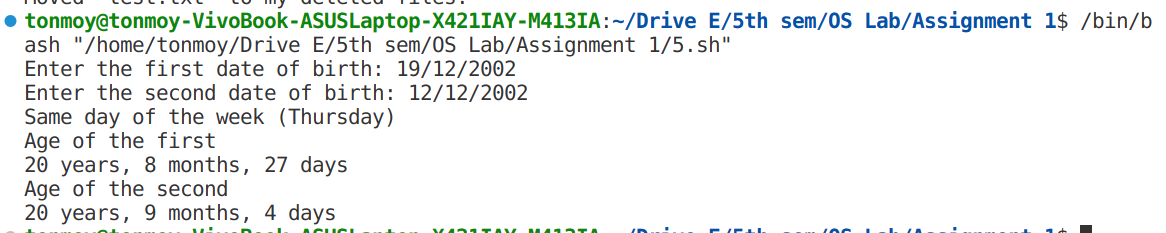
echo "Age of the first"

calculate\_age "$birthday1"

echo "Age of the second"

calculate\_age "$birthday2"

**Output**:



6. Write a shell script that accepts a file name as an input and performs the following activities on the given file. The program asks for a string of characters (that is, any word) to be provided by the user. The file will be searched to find whether it contains the given word. If the file contains the given word, the program will display (a) the number of occurrences of the word. The program is also required to display (b) the line number in which the word has occurred and no. of times the word has occurred in that line (Note: the word may occur more than once in a given line). If the file does not contain the word, an appropriate error message will be displayed.

**Solution**: grep command was used to find the number of words in the file as well as in each line. awk was used to extract out the lines from the file, and count is shown for each line. File name is taken as command line argument

**Code**:

#!/bin/bash

read -p "Enter the file name: " filename

#to check whether the file exist

if [ ! -f "$filename" ]

then

echo "Error: File '$filename' does not exist."

exit 1

fi

read -p "Enter the word: " word

#to count the total number of occurrence and show the error message

total\_count=$(grep -o -w "$word" "$filename" | wc -l)

if [ $total\_count -gt 0 ]

then

echo "Total no of occurrence of $word is $total\_count"

else

echo "Error: The file $filemane doesn't contain word $word"

exit 1

fi

line\_no=1

# To count the number of occurrence of word line by line

while read -r line

do

# echo "Line: $line"

word\_count=$(echo "$line" | grep -o -w "$word" | wc -l)

if [ $word\_count -gt 0 ]

then

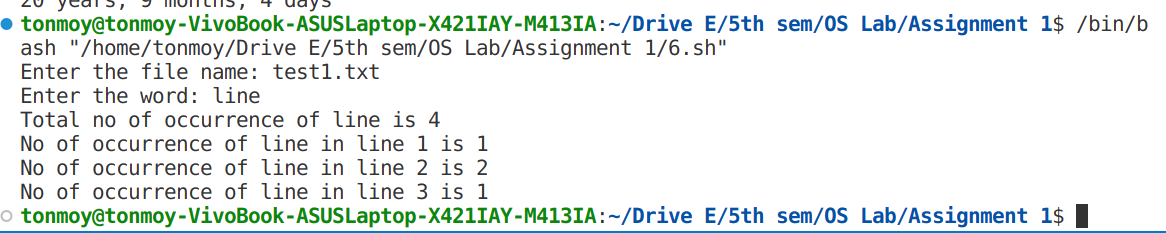
echo "No of occurrence of $word in line $line\_no is $word\_count"

fi

((line\_no+=1))

done < "$filename"

**Output** :



7. Extend the shell script written in (6) to perform the following task: User is asked to enter two different patterns or words. The first pattern will have to be matched with the contents of the file and replaced by the second pattern if a match occurs. If the first pattern does not occur in the file, an appropriate error message will be displayed.

**Solution**: Same as previous, the **–w** option was removed in grep command, since we are not finding whole matching words. Replace is done by “**sed**” command

**Code**:

#!/bin/bash

read -p "Enter the file name: " filename

#to check whether the file exist

if [ ! -f "$filename" ]

then

echo "Error: File '$filename' does not exist."

exit 1

fi

read -p "Enter the word: " word

#to count the total number of occurrence and show the error message

total\_count=$(grep -o -w "$word" "$filename" | wc -l)

if [ $total\_count -gt 0 ]

then

echo "Total no of occurrence of $word is $total\_count"

else

echo "Error: The file $filemane doesn't contain word $word"

exit 1

fi

line\_no=1

# To count the number of occurrence of word line by line

while read -r line

do

# echo "Line: $line"

word\_count=$(echo "$line" | grep -o -w "$word" | wc -l)

if [ $word\_count -gt 0 ]

then

echo "No of occurrence of $word in line $line\_no is $word\_count"

fi

((line\_no+=1))

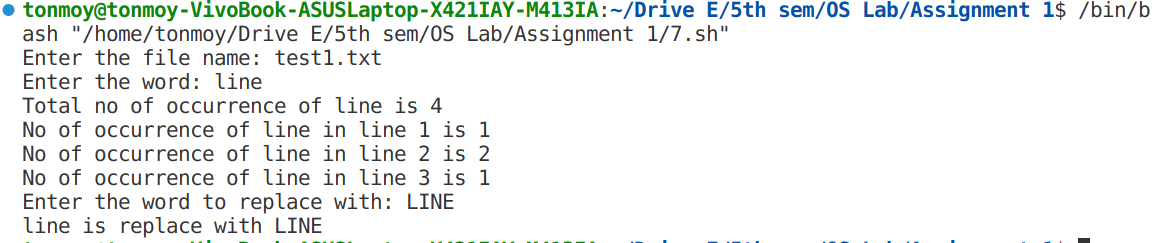
done < "$filename"

read -p "Enter the word to replace with: " new\_word

sed -i "s/$word/$new\_word/g" "$filename"

echo "$word is replace with $new\_word"

**Output**:





**Additional Assignment**

**Code:**

**#!/bin/bash**

filename="logfile.txt"

if [ ! -f $filename ]

then

touch $filename

fi

greeting(){

cur\_time=$(date +"%H")

time="morning"

if [ $cur\_time -gt 3 ]

then

time="evening"

fi

echo

echo "Hello $USER good $time"

echo "Hello $USER good $time" >> $filename

echo

}

show\_disk\_usage(){

disk\_usage=$(df -h)

echo

echo "Disk Usage Information:"

echo "Disk Usage Information:" >> $filename

echo "-----------------------"

echo "-----------------------" >> $filename

echo "$disk\_usage"

echo "$disk\_usage" >> $filename

echo

}

list\_files(){

read -p "Enter the minimum file size in bytes: " min\_size

echo

echo "List of files greater than or equal to ${min\_size} bytes:"

echo "List of files greater than or equal to ${min\_size} bytes:" >> $filename

echo "-----------------------------------------"

echo "-----------------------------------------" >> $filename

for file in "."/\*

do

size=$(du -b "$file")

for i in $size

do

# echo $i

if [ "$i" -ge "$min\_size" ]

then

echo "FileName: $file | Size: $i bytes"

echo "FileName: $file | Size: $i bytes" >> $filename

fi

break

done

done

echo "-----------------------------------------"

echo "-----------------------------------------" >> $filename

echo

}

show\_log\_file(){

echo

echo "The content of $filename"

echo

cat $filename

echo

}

while true

do

echo "1. Display Greeting"

echo "2. List Large files"

echo "3. Disk usage"

echo "4. View Log File"

echo "5. Exit"

read -p "Enter the choice no: " c

case $c in

1 )

echo "$USER%Display Greeting%" >> $filename

greeting ;;

2 )

echo "$USER%List Large files%" >> $filename

list\_files ;;

3 )

echo "$USER%Disk usage%" >> $filename

show\_disk\_usage ;;

4 )

echo "$USER%View Log File%" >> $filename

show\_log\_file;;

5 )

echo "$USER%Exit%" >> $filename

echo

echo "Exiting..."

echo

break ;;

\* )

echo "Error: You enter a wrong key"

break;;

esac

done

**Output**:

tonmoy@tonmoy-VivoBook-ASUSLaptop-X421IAY-M413IA:~/Drive E/5th sem/OS Lab/Assignment 1$ /bin/bash "/home/tonmoy/Drive E/5th sem/OS Lab/Assignment 1/8.sh"

1. Display Greeting

2. List Large files

3. Disk usage

4. View Log File

5. Exit

Enter the choice no: 1

Hello tonmoy good evening

1. Display Greeting

2. List Large files

3. Disk usage

4. View Log File

5. Exit

Enter the choice no: 2

Enter the minimum file size in bytes: 1000

List of files greater than or equal to 1000 bytes:

-----------------------------------------

FileName: ./3.sh | Size: 1275 bytes

FileName: ./4.sh | Size: 1774 bytes

FileName: ./5.sh | Size: 2158 bytes

FileName: ./8.sh | Size: 2371 bytes

FileName: ./Ass1\_report.docx | Size: 429418 bytes

FileName: ./BCSE-OS-Assignments-I-2023[809].pdf | Size: 126270 bytes

FileName: ./dir | Size: 4096 bytes

FileName: ./my-deleted-files | Size: 4114 bytes

FileName: ./q3.sh | Size: 1038 bytes

FileName: ./tempCodeRunnerFile.sh | Size: 1038 bytes

-----------------------------------------

1. Display Greeting

2. List Large files

3. Disk usage

4. View Log File

5. Exit

Enter the choice no: 3

Disk Usage Information:

-----------------------

Filesystem Size Used Avail Use% Mounted on

tmpfs 735M 2.1M 733M 1% /run

/dev/nvme0n1p2 468G 52G 393G 12% /

tmpfs 3.6G 47M 3.6G 2% /dev/shm

tmpfs 5.0M 4.0K 5.0M 1% /run/lock

/dev/nvme0n1p1 511M 6.1M 505M 2% /boot/efi

tmpfs 735M 112K 735M 1% /run/user/1000

1. Display Greeting

2. List Large files

3. Disk usage

4. View Log File

5. Exit

Enter the choice no: 4

The content of logfile.txt

%List Large files%

List of files greater than or equal to 1000 bytes:

-----------------------------------------

FileName: ./4.sh | Size: 2449 bytes

FileName: ./5.sh | Size: 2523 bytes

FileName: ./6.sh | Size: 1579 bytes

FileName: ./7.sh | Size: 1379 bytes

FileName: ./8.sh | Size: 3441 bytes

FileName: ./BCSE-OS-Assignments-I-2023[809].pdf | Size: 126270 bytes

FileName: ./q3.sh | Size: 1025 bytes

-----------------------------------------

%Exit%

tonmoy%Display Greeting%

Hello tonmoy good evening

tonmoy%List Large files%

List of files greater than or equal to 1000 bytes:

-----------------------------------------

FileName: ./3.sh | Size: 1275 bytes

FileName: ./4.sh | Size: 1774 bytes

FileName: ./5.sh | Size: 2158 bytes

FileName: ./8.sh | Size: 2371 bytes

FileName: ./Ass1\_report.docx | Size: 429418 bytes

FileName: ./BCSE-OS-Assignments-I-2023[809].pdf | Size: 126270 bytes

FileName: ./dir | Size: 4096 bytes

FileName: ./my-deleted-files | Size: 4114 bytes

FileName: ./q3.sh | Size: 1038 bytes

FileName: ./tempCodeRunnerFile.sh | Size: 1038 bytes

-----------------------------------------

tonmoy%Disk usage%

Disk Usage Information:

-----------------------

Filesystem Size Used Avail Use% Mounted on

tmpfs 735M 2.1M 733M 1% /run

/dev/nvme0n1p2 468G 52G 393G 12% /

tmpfs 3.6G 47M 3.6G 2% /dev/shm

tmpfs 5.0M 4.0K 5.0M 1% /run/lock

/dev/nvme0n1p1 511M 6.1M 505M 2% /boot/efi

tmpfs 735M 112K 735M 1% /run/user/1000

tonmoy%View Log File%

1. Display Greeting

2. List Large files

3. Disk usage

4. View Log File

5. Exit

Enter the choice no: 5

Exiting...

tonmoy@tonmoy-VivoBook-ASUSLaptop-X421IAY-M413IA:~/Drive E/5th sem/OS Lab/Assignment 1$ ^C