**Operating Systems – Lab 03**

**Name:** Hafsa Salman

**Roll no.** 22K-5161

**Task no. 01**

Code:

#!/bin/bash

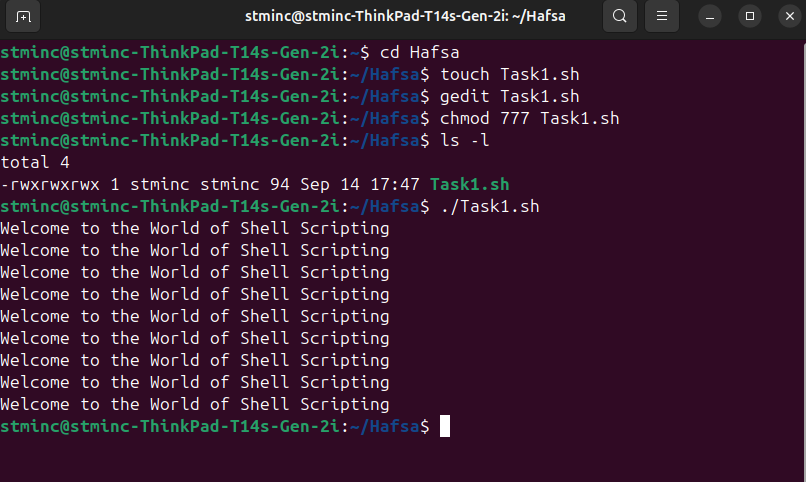
for (( i=1; i<10; i++ ))

do

echo "Welcome to the World of Shell Scripting"

done

Output:



**Task no. 02**

Code:

#!/bin/bash

read -p "Enter Name: " name

read -p "Enter Degree: " degree

read -p "Enter batch no. " batch

read -p "Enter Course Title: " title

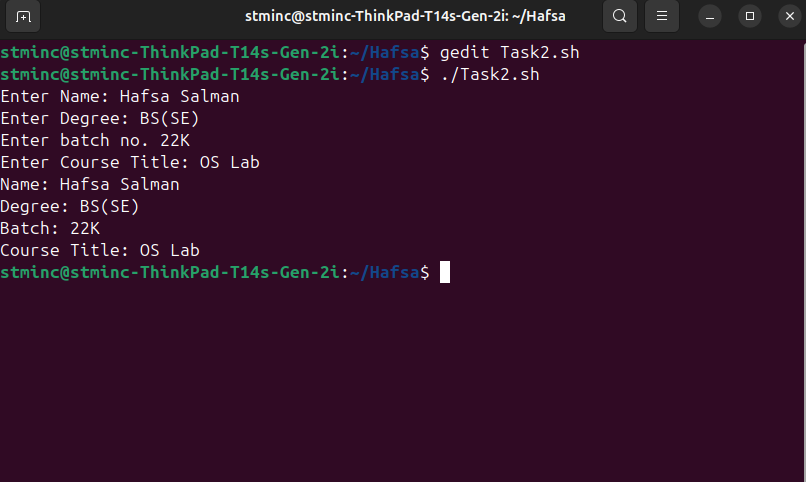
echo "Name: $name"

echo "Degree: $degree"

echo "Batch: $batch"

echo "Course Title: $title"

Output:



**Task no. 03**

Code:

#!/bin/bash

read -p "Enter number: " num

if [[ $num -lt 0 ]];

then

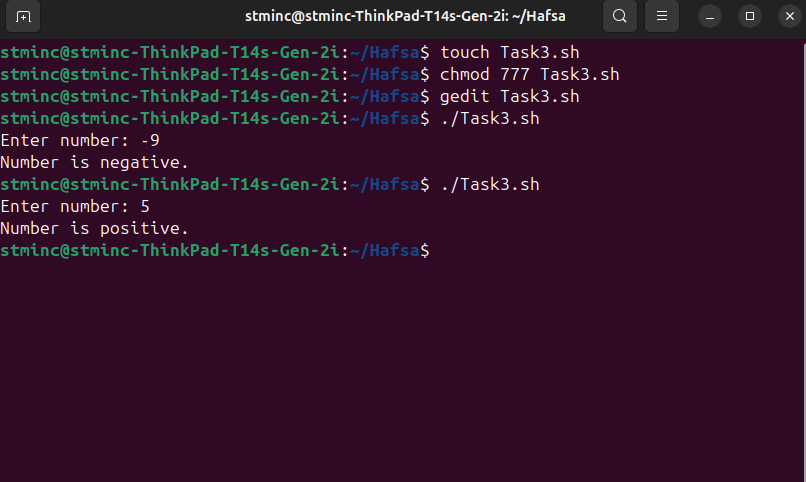
echo "Number is negative."

else

echo "Number is positive."

fi

Output:



**Task no. 04**

Code:

#!/bin/bash

read -p "Enter number: " num

if [[ $num -eq 0 ]];

then

echo "Number is zero."

elif [[ $((num % 2)) -eq 0 ]];

then

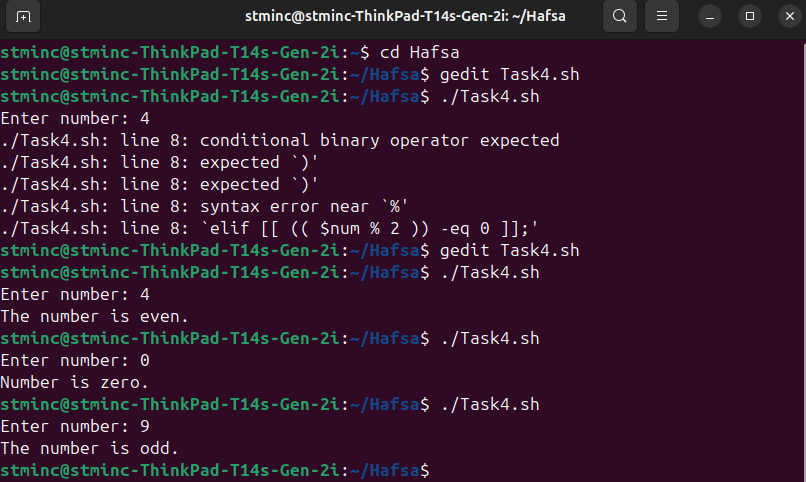
echo "The number is even."

else

echo "The number is odd."

fi

Output:



**Task no. 05**

Code:

#!/bin/bash

read -p "Enter number: " num

prime=0

for (( i=2; i<=num/2; i++ ));

do

if [[ $((num % 2 )) -eq 0 ]];

then

prime=1

break

fi

done

if [[ $prime -eq 0 ]];

then

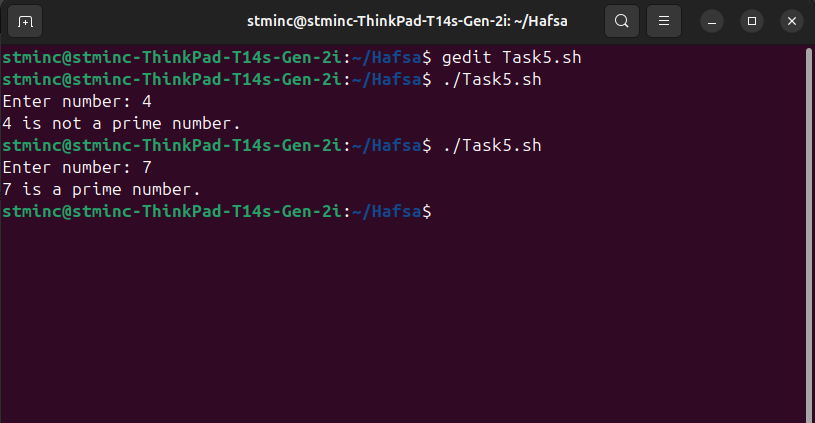
echo "$num is a prime number."

else

echo "$num is not a prime number."

fi

Output:



**Task no. 06**

Code:

read -p "Enter number 1: " num1

read -p "Enter number 2: " num2

read -p "Enter number 3: " num3

read -p "Enter number 4: " num4

read -p "Enter number 5: " num5

max=$num1

min=$num1

if [ $num2 -gt $max ];

then

max=$num2

fi

if [ $num3 -gt $max ];

then

max=$num3

fi

if [ $num4 -gt $max ];

then

max=$num4

fi

if [ $num5 -gt $max ];

then

max=$num5

fi

if [ $num2 -lt $min ];

then

min=$num2

fi

if [ $num3 -lt $min ];

then

min=$num3

fi

if [ $num4 -lt $min ];

then

min=$num4

fi

if [ $num5 -lt $min ];

then

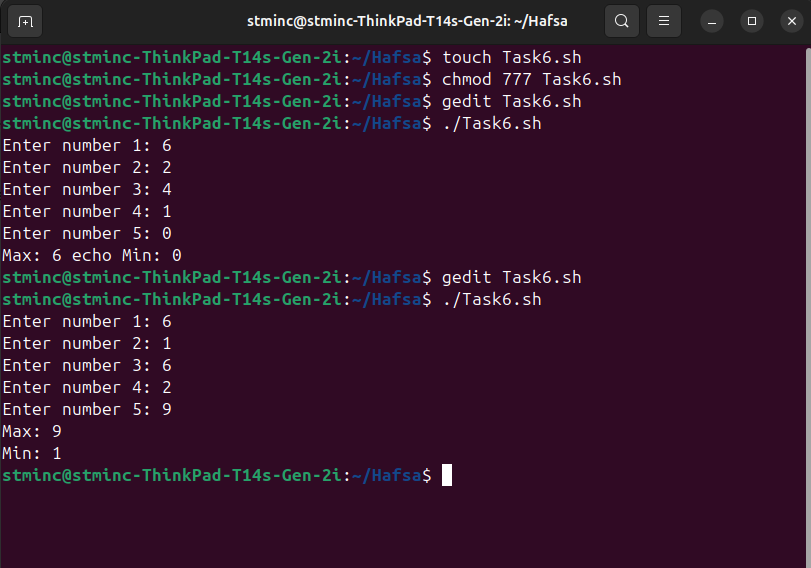
min=$num5

fi

echo "Max: $max"

echo "Min: $min"

Output:



**Task no. 07**

Code:

#!/bin/bash

fibonacci()

{

local n=$1

if [[ $n -eq 0 ]];

then

echo -n "0"

elif [[ $n -eq 1 ]];

then

echo -n "1"

else

local fn1=$(fibonacci $((n-1)))

local fn2=$(fibonacci $((n-2)))

echo -n "$((fn1 + fn2)) "

fi

}

read -p "Enter the number of terms for Fibonacci series: " n

echo -n "Fibonacci Series up to $n terms: "

for (( i=0; i<n; i++ ));

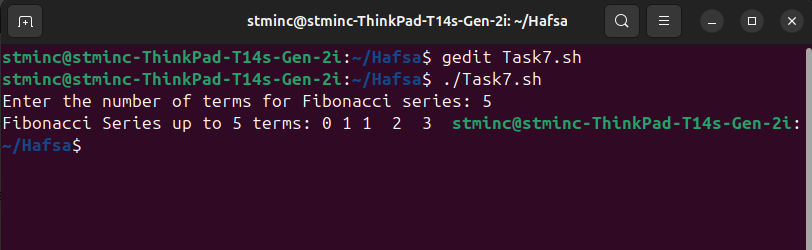
do

fibonacci $i

echo -n " "

done

Output:



**Task no. 08**

Code:

#!/bin/bash

read -p "Enter any number: " num

factorial=1

for (( i=1 ; i<=num ; i++));

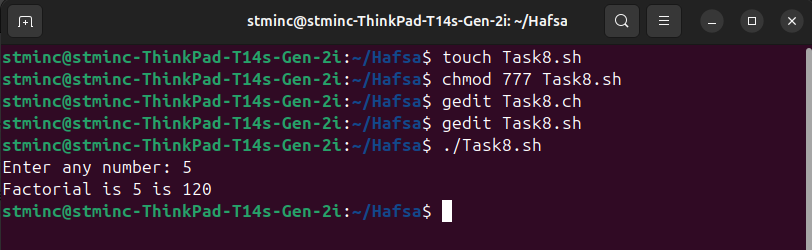
do

factorial=$((factorial \* i))

done

echo "Factorial is $num is $factorial"

Output:



**Task no. 09**

Code:

#!/bin/bash

read -p "Enter name of file to move: " name

if [[ -f "$name" ]];

then

ext="${name##\*.}"

if [[ "$ext" == "sh" ]];

then

mkdir -p shelldir

mv "$name" shelldir/

elif [[ "$ext" == "c" ]];

then

mkdir -p cdir

mv "$name" cdir/

elif [[ "$ext" == "jpg" ]];

then

mkdir -p jpgdir

mv "$name" jpgdir/

else

echo "Unsupported file extension: $ext"

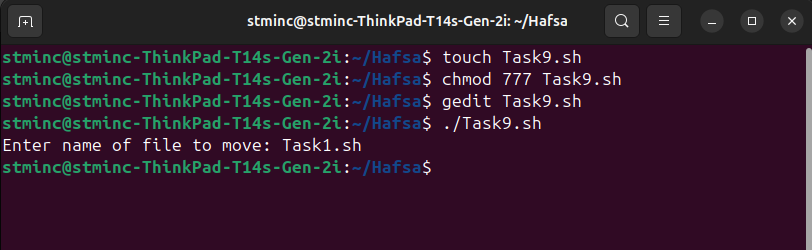
fi

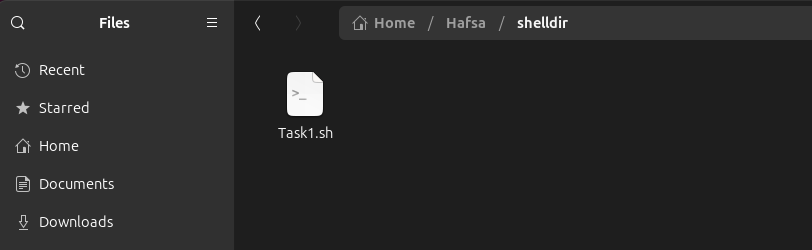
else

echo " File doesnt exist! "

fi

Output:





**Task no. 10**

Code:

#!/bin/bash

echo "Select an option:"

echo "1. Create folders and files"

echo "2. Change file permissions"

echo "3. Search for files"

echo "4. Display and manage processes"

echo "5. Write and run a shell script"

echo "6. Write and run a C program"

read -p "Enter choice: " choice

case $choice in

1)read -p "How many folders to create? " f\_count

for ((i=1; i<=f\_count; i++)); do

read -p "Enter name of folder $i: " fname

mkdir -p "$fname"

echo "Folder '$fname' created."

done

read -p "How many files to create? " fi\_count

for ((i=1; i<=fi\_count; i++)); do

read -p "Enter name of file $i: " finame

touch "$finame"

echo "File '$finame' created."

done;;

2)read -p "Enter file name to change permissions: " filename

if [[ -e "$filename" ]]; then

read -p "Enter permission code (e.g. 755): " perm

chmod "$perm" "$filename"

echo "Permissions updated."

else

echo "File does not exist."

fi;;

3)read -p "Enter file name to search: " sfile

find . -name "$sfile";;

4)ps -e -o pid,cmd

read -p "Enter PID to kill (or press Enter to skip): " pid

if [[ -n "$pid" ]]; then

kill "$pid" && echo "Process $pid killed." || echo "Failed to kill process."

fi;;

5)read -p "Enter name of shell script to create (without .sh): " sname

sname="$sname.sh"

nano "$sname"

chmod +x "$sname"

bash "$sname" &;;

6)read -p "Enter name of C program (without .c): " cname

cfile="$cname.c"

nano "$cfile"

gcc "$cfile" -o "$cname"

if [[ -f "$cname" ]]; then

./"$cname" &

else

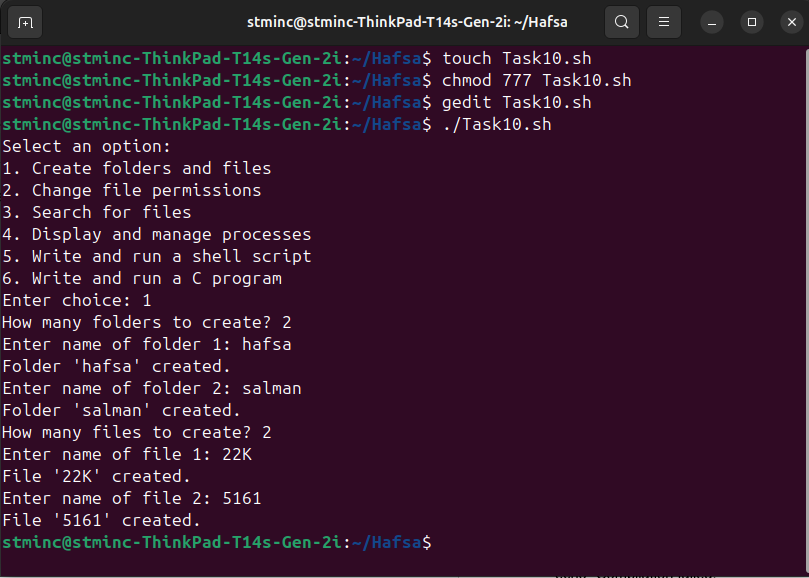
echo "Compilation failed."

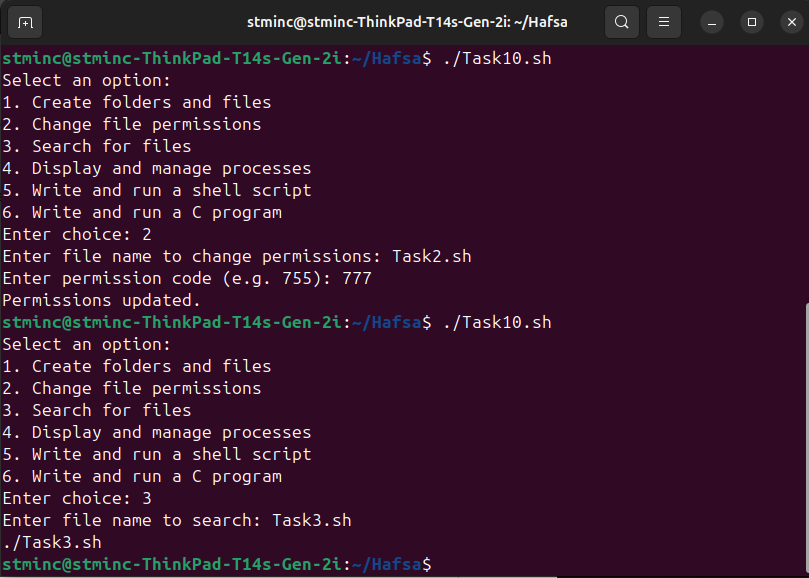
fi;;

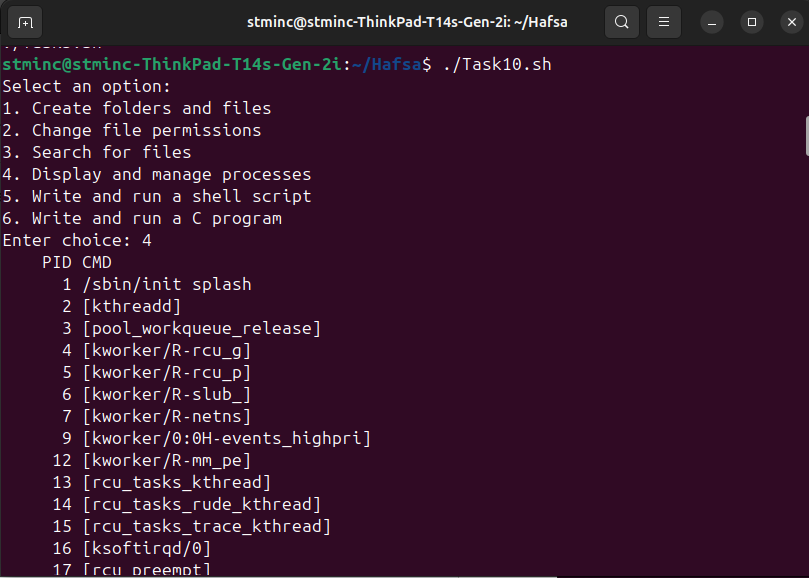
\*)echo "Invalid choice.";;

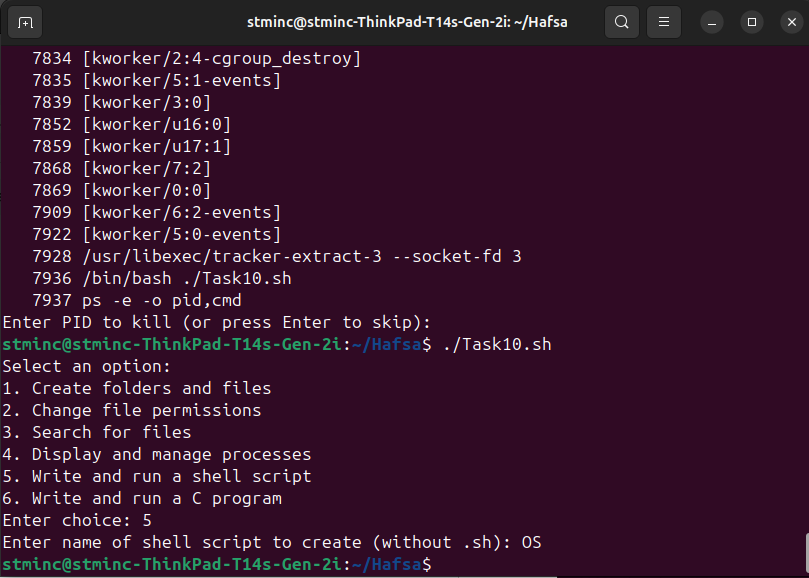
esac

Output:









**Task no. 11**

Code:

#!/bin/bash

if [[ $# -ne 2 ]];

then

echo "Usage: $0 ./Hafsa/Task4.sh TaskFour"

exit 1

fi

dir="$1"

pattern="$2"

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

then

echo "Error: Directory '$dir' does not exist."

exit 1

fi

cd "$dir" || { echo "Error: Cannot access directory."; exit 1; }

count=1

for file in \*;

do

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

then

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

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

if [[ "$file" == "$ext" ]];

then

newname="${pattern}${count}"

else

newname="${pattern}${count}.$ext"

fi

if [[ -e "$newname" ]];

then

echo "Skipping '$file' → '$newname' already exists."

else

mv "$file" "$newname"

echo "Renamed '$file' → '$newname'"

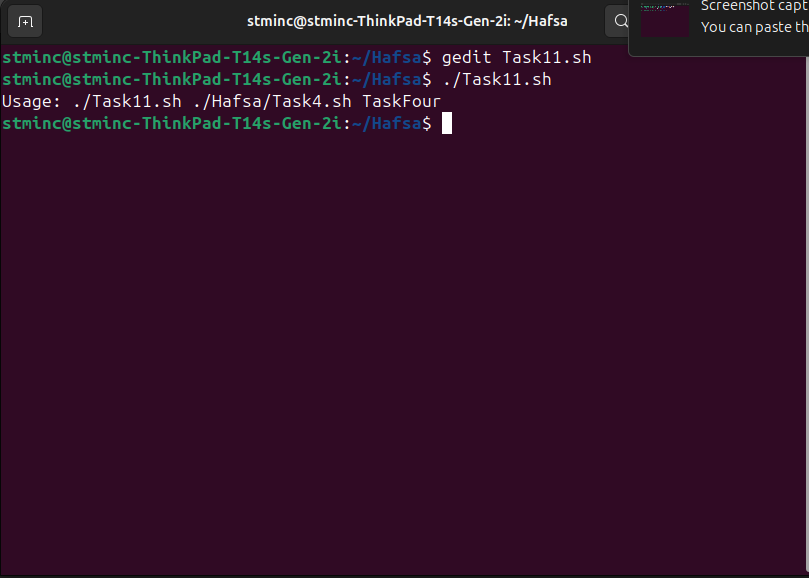
((count++))

fi

fi

done

Output:



**Task no. 12**

Code:

#!/bin/bash

if [[ $# -ne 2 ]];

then

echo "Usage: $0 /home/stminc/Hafsa 30"

exit 1

fi

dir="$1"

days="$2"

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

then

echo "Error: '$dir' is not a valid directory."

exit 1

fi

if ! [[ "$days" =~ ^[0-9]+$ ]];

then

echo "Error: Days must be a non-negative integer."

exit 1

fi

echo "Starting cleanup in '$dir'..."

echo "Removing files older than $days days..."

file\_count=$(find "$dir" -type f -mtime +$days -print | wc -l)

find "$dir" -type f -mtime +$days -print -delete

dir\_count=$(find "$dir" -type d -empty -print | wc -l)

find "$dir" -type d -empty -print -delete

echo "Cleanup complete."

echo "Files deleted: $file\_count"

echo "Empty directories deleted: $dir\_count"

Output:

