SUNNY KUMAR
31 JULY TASK
LSP (LINUX SYSTEM PROGRAMMING)

SHELL SCRIPTING
#!/bin/bash

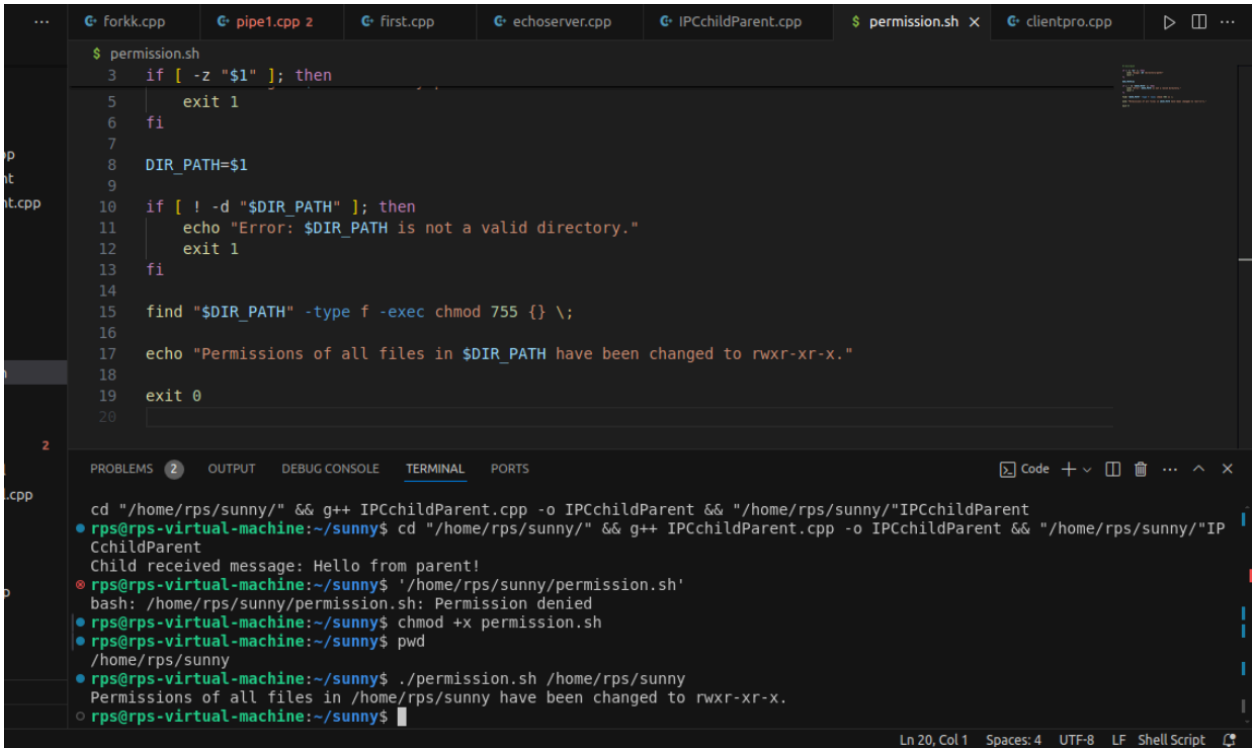
```
HI=Hello

echo HI        # displays HI
echo $HI       # displays Hello
echo \$HI      # displays $HI
echo "$HI"     # displays Hello
echo '$HI'     # displays $HI
echo "$HIalex"  # displays nothing
echo "${HI}Alex" # displays HelloAlex
echo `pwd`     # displays working directory
echo $(pwd)    # displays working directory
```

Q.Change File Permissions
Description: Write a shell script that takes a directory path as an argument and changes the permissions of all files within that directory to read, write, and execute for the owner, and read and execute for the group and others.

Instructions:

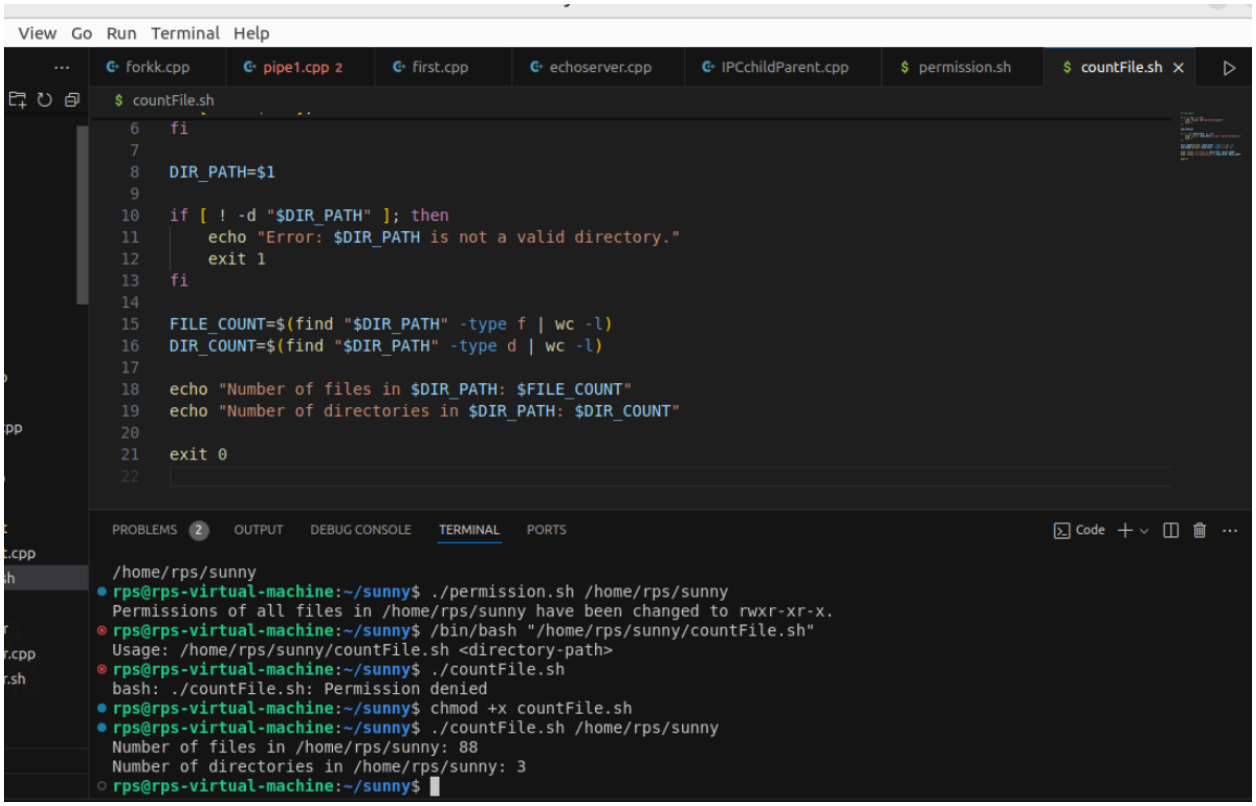
The script should accept one argument, the directory path.
Change permissions of all files in the specified directory to rwxr-xr-x.
Print a message indicating the completion of the permission change.



Q. Problem 2: Count Files and Directories
Description: Write a shell script that counts the number of files and directories in a given directory.

Instructions:

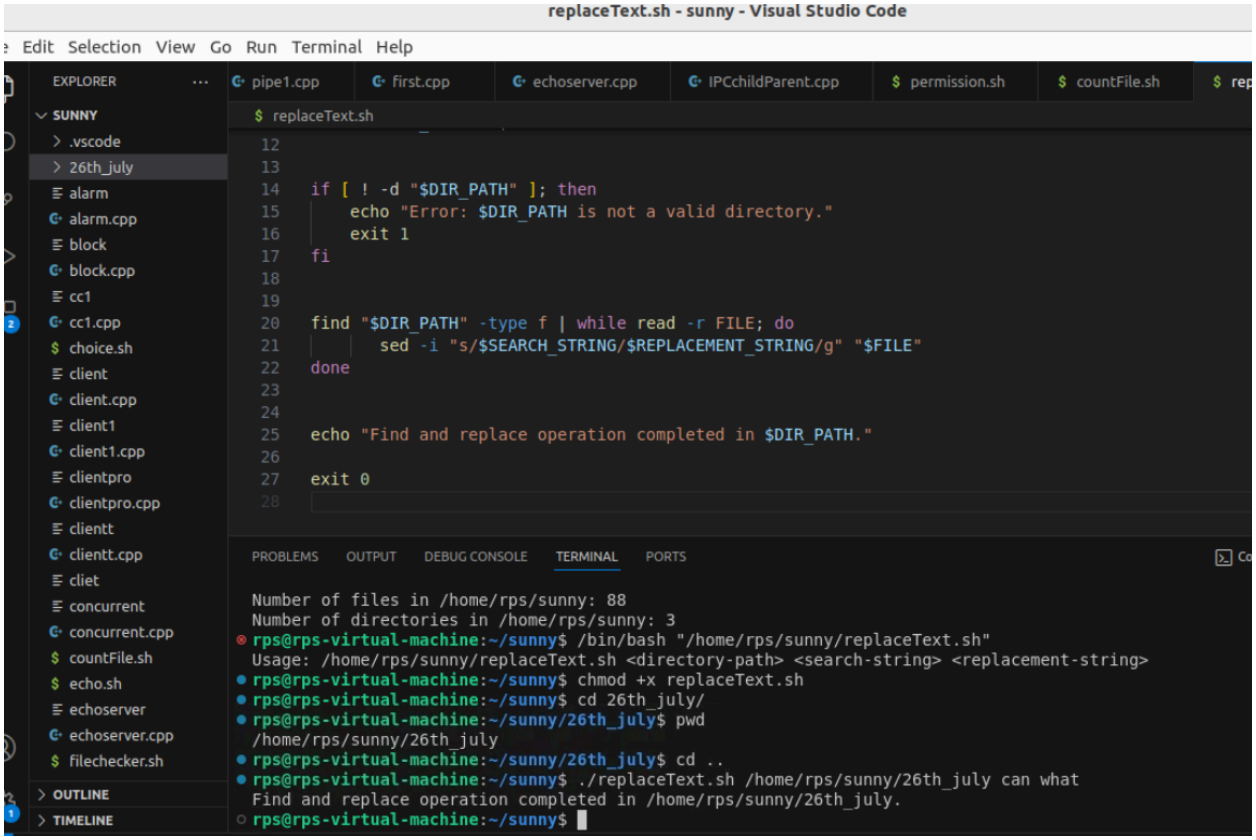
The script should accept one argument, the directory path.
Count the number of files and directories separately.
Print the counts with appropriate labels.
Sample Input:
./count_files_dirs.sh /path/to/directory



Q.Problem 3: Find and Replace Text in Files
Description: Write a shell script to search for a specific text string in all files within a directory and replace it with another string.

Instructions:

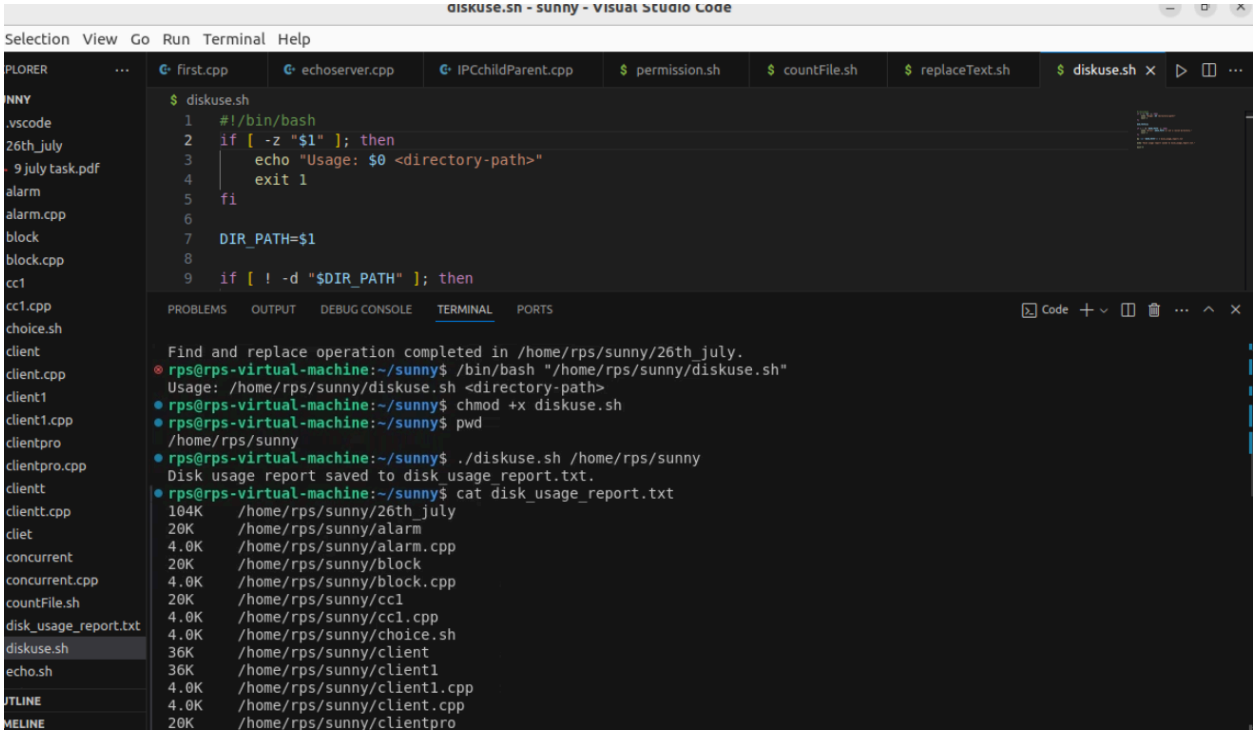
The script should accept three arguments: directory path, search string, and replacement string.
Search for the specified string in all files within the directory.
Replace the string with the given replacement string in all occurrences.
Print a message indicating the completion of the find and replace operation.
Sample Input:
./find_replace.sh /path/to/directory "old_text" "new_text"



Q. Problem 4: Disk Usage Report
Description: Write a shell script that generates a report of disk usage for a specified directory.

Instructions:

The script should accept one argument, the directory path.
Use the du command to generate a disk usage report for the directory.
Save the report to a file named disk_usage_report.txt in the current directory.
Print a message indicating where the report is saved.



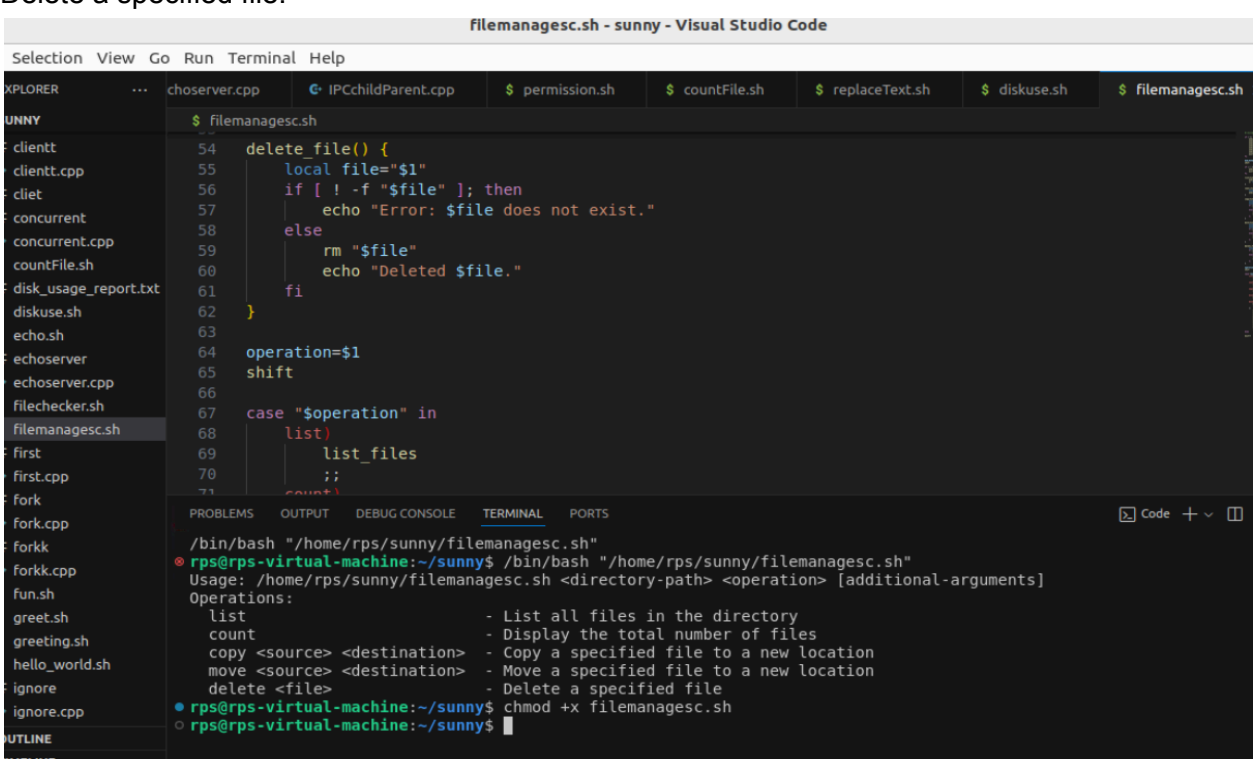
CALCULATOR CODE

```
#!/bin/bash
echo "simple calculator"
sum=0
i="y"
echo "enter first number"
read n1
echo "enter second number"
read n2
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter choice"
read ch
case $ch in
1)sum=$(echo " $n1 + $n2" | bc -l)
echo "Addition is =" $sum;;
2)sum=$(echo "$n1 - $n2" | bc -l)
echo "Sub is =" $sum;;
```

```
3)sum=$(echo "$n1 * $n2" | bc -l)
echo "Mul is =" $sum;;
4)sum=$(echo "$n1 / $n2" | bc -l)
echo "div is =" $sum;;
*)echo "invalid choice"
esac
echo "Do you want to continue"
read i
if [ $i != "y" ]
then
exit
fi
done
```

Q. Problem Statement: File Management Script with Functions and Arguments
Objective
Create a shell script that manages files in a specified directory. The script should include functions to perform the following tasks:

List all files in the directory.
Display the total number of files.
Copy a specified file to a new location.
Move a specified file to a new location.
Delete a specified file.



Q. Problem Statement: File Operations using System Calls in C++
Description:
Write a C++ program that performs various file operations using Linux system calls. The program should create a file, write to it, read from it, and then delete the file. The program should handle errors appropriately and ensure proper resource management (e.g., closing file descriptors).

Instructions:

Create a File:

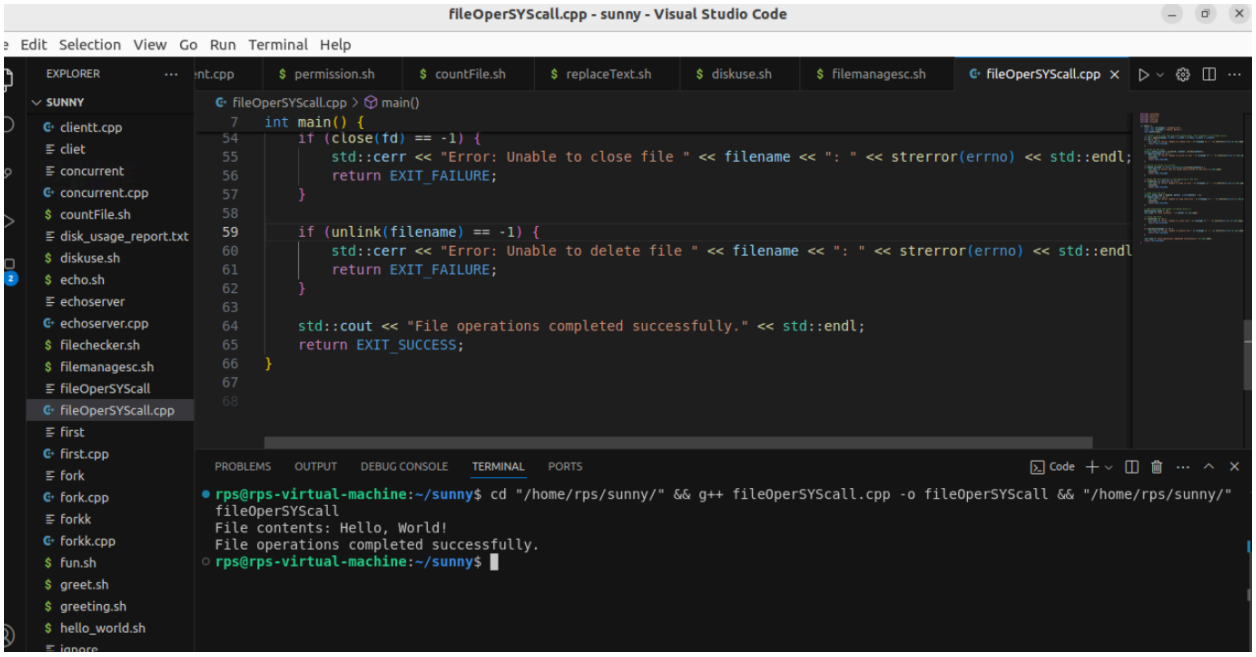
Use the open system call to create a new file named "example.txt" with read and write permissions.
If the file already exists, truncate its contents.
Write to the File:

Write the string "Hello, World!" to the file using the write system call.
Ensure that all bytes are written to the file.
Read from the File:

Use the lseek system call to reset the file pointer to the beginning of the file.
Read the contents of the file using the read system call and store it in a buffer.
Print the contents of the buffer to the standard output.
Delete the File:

Close the file descriptor using the close system call.
Use the unlink system call to delete the file "example.txt".
Error Handling:

Ensure proper error handling for each system call. If a system call fails, print an error message and exit the program with a non-zero status.


fileOperSYScall.cpp - sunny - Visual Studio Code
Edit Selection View Go Run Terminal Help
EXPLORER
SUNNY
clientt.cpp
cliet
concurrent
concurrent.cpp
countFile.sh
disk_usage_report.txt
diskuse.sh
echo.sh
echoserver
echoserver.cpp
filechecker.sh
filemanagesc.sh
fileOperSYScall
fileOperSYScall.cpp
first
first.cpp
fork
fork.cpp
forkk
forkk.cpp
fun.sh
greet.sh
greeting.sh
hello_world.sh
permission.sh
countFile.sh
replaceText.sh
diskuse.sh
filemanagesc.sh
fileOperSYScall.cpp
fileOperSYScall.cpp > main()
int main() {
if (close(fd) == -1) {
std::cerr << "Error: Unable to close file " << filename << ": " << strerror(errno) << std::endl;
return EXIT_FAILURE;
}
if (unlink(filename) == -1) {
std::cerr << "Error: Unable to delete file " << filename << ": " << strerror(errno) << std::endl
return EXIT_FAILURE;
}
std::cout << "File operations completed successfully." << std::endl;
return EXIT_SUCCESS;
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
rps@rps-virtual-machine:~/sunny$ cd "/home/rps/sunny/" && g++ fileOperSYScall.cpp -o fileOperSYScall && "/home/rps/sunny/"
fileOperSYScall
File contents: Hello, World!
File operations completed successfully.
rps@rps-virtual-machine:~/sunny$

Q.