

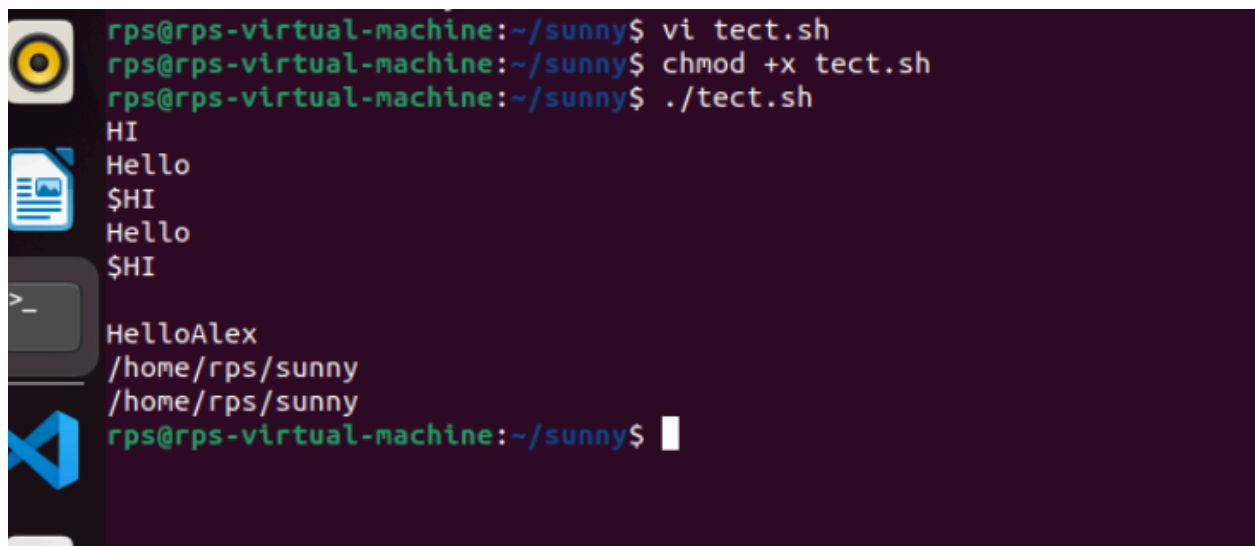
SUNNY KUMAR  
31 JULY TASK  
LSP (LINUX SYSTEM PROGRAMMING)

**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
```



```
rps@rps-virtual-machine:~/sunny$ vi tect.sh
rps@rps-virtual-machine:~/sunny$ chmod +x tect.sh
rps@rps-virtual-machine:~/sunny$ ./tect.sh
HI
Hello
$HI
Hello
$HI

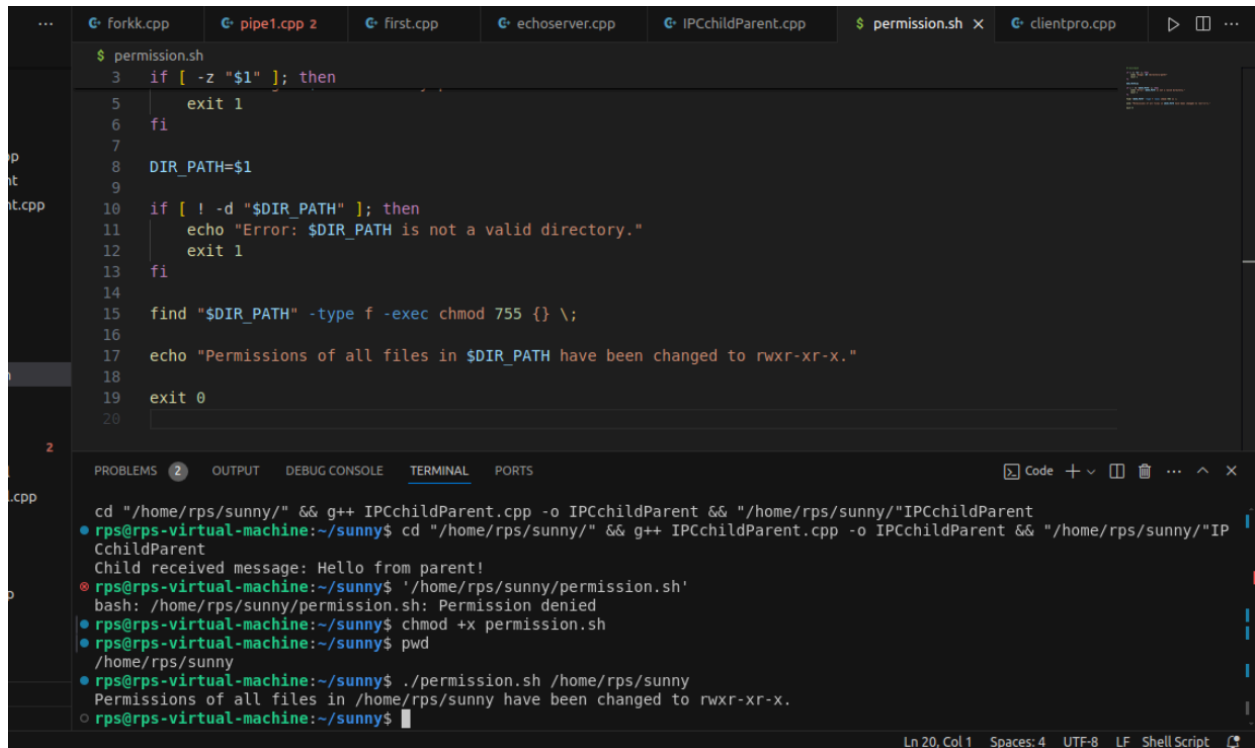
HelloAlex
/home/rps/sunny
/home/rps/sunny
rps@rps-virtual-machine:~/sunny$
```

**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.



```
$ permission.sh
3  if [ -z "$1" ]; then
5      exit 1
6  fi
7
8  DIR_PATH=$1
9
10 if [ ! -d "$DIR_PATH" ]; then
11     echo "Error: $DIR_PATH is not a valid directory."
12     exit 1
13 fi
14
15 find "$DIR_PATH" -type f -exec chmod 755 {} \;
16
17 echo "Permissions of all files in $DIR_PATH have been changed to rwxr-xr-x."
18
19 exit 0
20
```

```
cd "/home/rps/sunny/" && g++ IPCchildParent.cpp -o IPCchildParent && "/home/rps/sunny"/IPCchildParent
rps@rps-virtual-machine:~/sunny$ cd "/home/rps/sunny/" && g++ IPCchildParent.cpp -o IPCchildParent && "/home/rps/sunny"/IPCchildParent
Child received message: Hello from parent!
rps@rps-virtual-machine:~/sunny$ '/home/rps/sunny/permission.sh'
bash: /home/rps/sunny/permission.sh: Permission denied
rps@rps-virtual-machine:~/sunny$ chmod +x permission.sh
rps@rps-virtual-machine:~/sunny$ pwd
/home/rps/sunny
rps@rps-virtual-machine:~/sunny$ ./permission.sh /home/rps/sunny
Permissions of all files in /home/rps/sunny have been changed to rwxr-xr-x.
rps@rps-virtual-machine:~/sunny$
```

## 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:

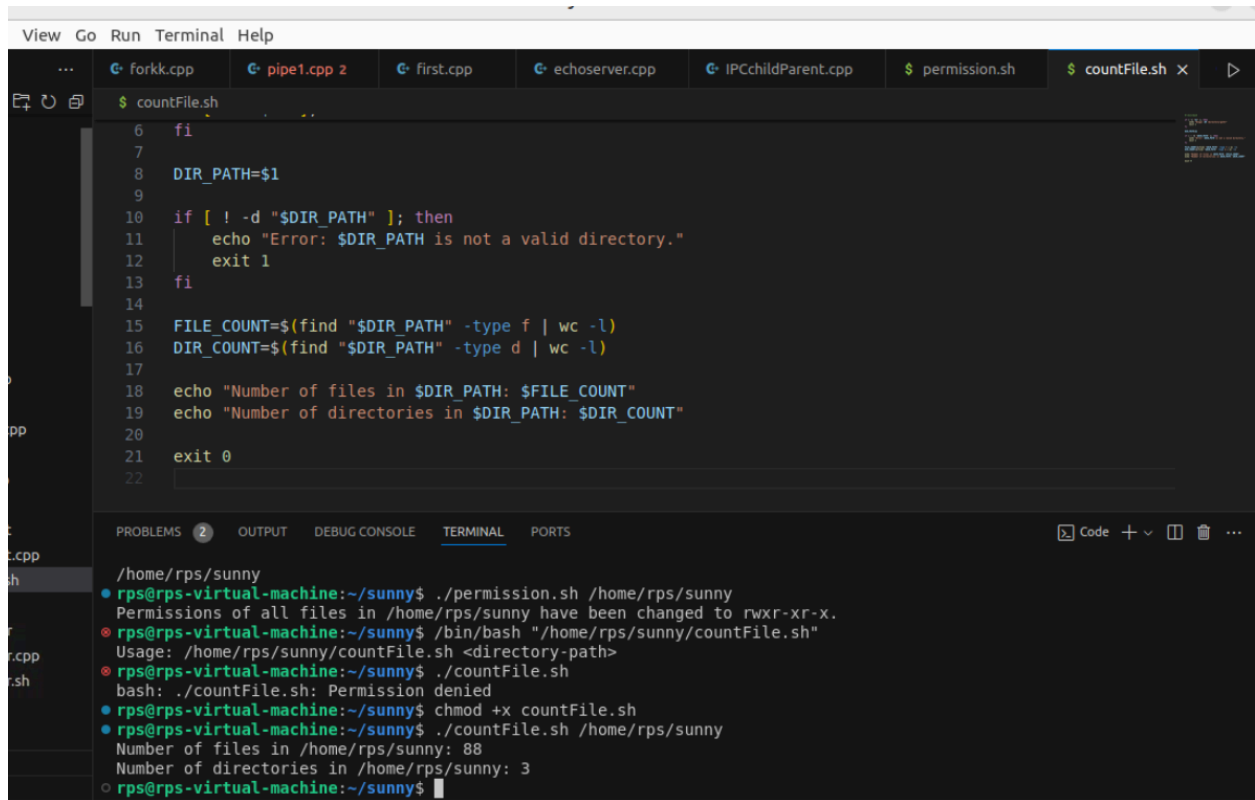
The script should accept one argument, the directory path.

Count the number of files and directories separately.

Print the counts with appropriate labels.

Sample Input:

`./count_files_dirs.sh /path/to/directory`



```
View Go Run Terminal Help
countFile.sh
6  fi
7
8  DIR_PATH=$1
9
10 if [ ! -d "$DIR_PATH" ]; then
11     echo "Error: $DIR_PATH is not a valid directory."
12     exit 1
13 fi
14
15 FILE_COUNT=$(find "$DIR_PATH" -type f | wc -l)
16 DIR_COUNT=$(find "$DIR_PATH" -type d | wc -l)
17
18 echo "Number of files in $DIR_PATH: $FILE_COUNT"
19 echo "Number of directories in $DIR_PATH: $DIR_COUNT"
20
21 exit 0
22

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
/home/rps/sunny
rps@rps-virtual-machine:~/sunny$ ./permission.sh /home/rps/sunny
Permissions of all files in /home/rps/sunny have been changed to rwxr-xr-x.
rps@rps-virtual-machine:~/sunny$ /bin/bash "/home/rps/sunny/countFile.sh"
Usage: /home/rps/sunny/countFile.sh <directory-path>
rps@rps-virtual-machine:~/sunny$ ./countFile.sh
bash: ./countFile.sh: Permission denied
rps@rps-virtual-machine:~/sunny$ chmod +x countFile.sh
rps@rps-virtual-machine:~/sunny$ ./countFile.sh /home/rps/sunny
Number of files in /home/rps/sunny: 88
Number of directories in /home/rps/sunny: 3
rps@rps-virtual-machine:~/sunny$
```

### 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"
```

The screenshot shows the Visual Studio Code interface with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'SUNNY' with a directory '26th\_july'. The file 'replaceText.sh' is open in the editor. The script contains the following code:

```
12
13
14 if [ ! -d "$DIR_PATH" ]; then
15     echo "Error: $DIR_PATH is not a valid directory."
16     exit 1
17 fi
18
19
20 find "$DIR_PATH" -type f | while read -r FILE; do
21     sed -i "s/$SEARCH_STRING/$REPLACEMENT_STRING/g" "$FILE"
22 done
23
24
25 echo "Find and replace operation completed in $DIR_PATH."
26
27 exit 0
28
```

The terminal shows the execution of the script. The user runs `/bin/bash "/home/rps/sunny/replaceText.sh"` with usage information. Then, they run `chmod +x replaceText.sh`, `cd 26th_july/`, `pwd` (showing `/home/rps/sunny/26th_july`), `cd ..`, and finally `./replaceText.sh /home/rps/sunny/26th_july can what`. The script outputs `Find and replace operation completed in /home/rps/sunny/26th_july.`

#### Q. Problem 4: Disk Usage Report

Description: Write a shell script that generates a report of disk usage for a specified directory.

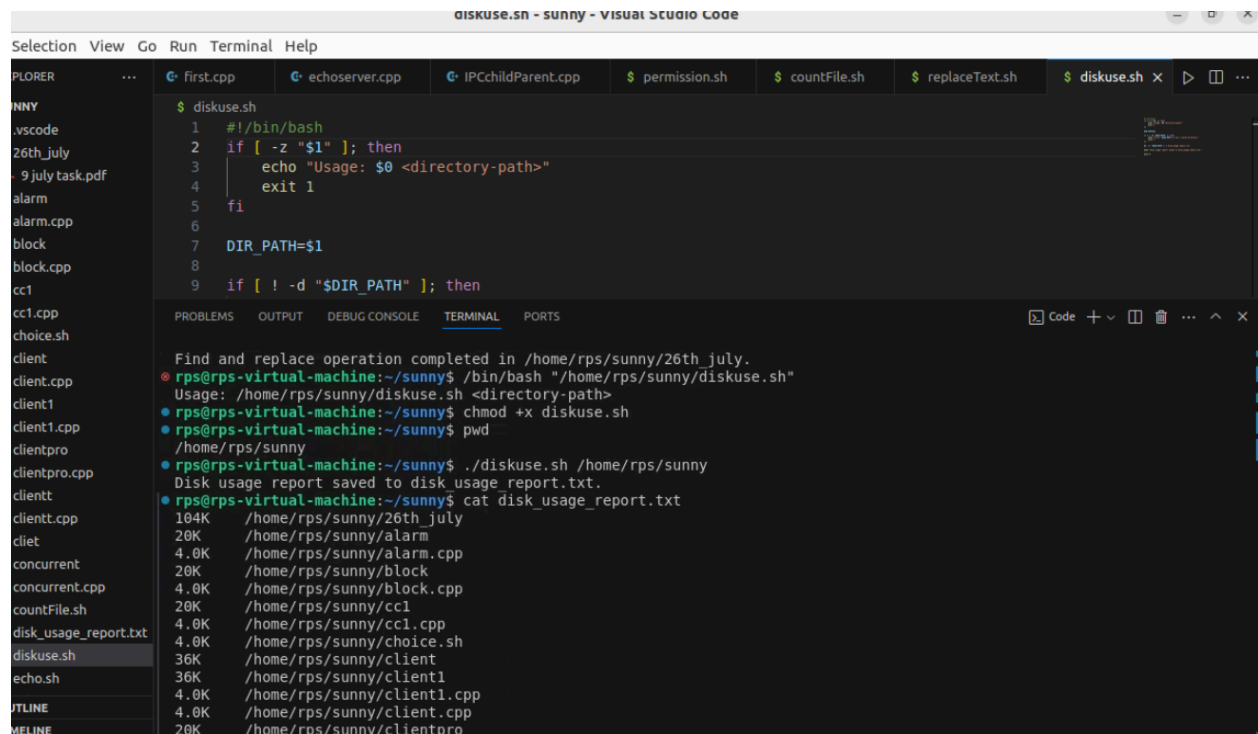
Instructions:

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.



The screenshot shows the Visual Studio Code interface with the file explorer on the left listing various files and folders. The main editor displays the content of `diskuse.sh`, a shell script that takes a directory path as an argument and prints the disk usage for that path. The terminal at the bottom shows the execution of the script with the following commands and output:

```
$ diskuse.sh
1  #!/bin/bash
2  if [ -z "$1" ]; then
3      echo "Usage: $0 <directory-path>"
4      exit 1
5  fi
6
7  DIR_PATH=$1
8
9  if [ ! -d "$DIR_PATH" ]; then
```

Find and replace operation completed in /home/rps/sunny/26th july.  
rps@rps-virtual-machine:~/sunny\$ /bin/bash "/home/rps/sunny/diskuse.sh"  
Usage: /home/rps/sunny/diskuse.sh <directory-path>  
rps@rps-virtual-machine:~/sunny\$ chmod +x diskuse.sh  
rps@rps-virtual-machine:~/sunny\$ pwd  
/home/rps/sunny  
rps@rps-virtual-machine:~/sunny\$ ./diskuse.sh /home/rps/sunny  
Disk usage report saved to disk usage report.txt.  
rps@rps-virtual-machine:~/sunny\$ cat disk\_usage\_report.txt

File	Size
/home/rps/sunny/26th_july	104K
/home/rps/sunny/alarm	20K
/home/rps/sunny/alarm.cpp	4.0K
/home/rps/sunny/block	20K
/home/rps/sunny/block.cpp	4.0K
/home/rps/sunny/cc1	20K
/home/rps/sunny/cc1.cpp	4.0K
/home/rps/sunny/choice.sh	4.0K
/home/rps/sunny/client	36K
/home/rps/sunny/client1	36K
/home/rps/sunny/client1.cpp	4.0K
/home/rps/sunny/client.cpp	4.0K
/home/rps/sunny/clientpro	20K

## 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.

The screenshot shows the Visual Studio Code editor with the filemanagesc.sh script open. The script contains functions for file management and a main loop that takes user input. The terminal output shows the script being executed and the usage instructions.

```

filemanagesc.sh - sunny - Visual Studio Code
Selection View Go Run Terminal Help
XPLORER  ... choserver.cpp  IPCchildParent.cpp  permission.sh  countFile.sh  replaceText.sh  diskuse.sh  filemanagesc.sh
UNNNY
$ filemanagesc.sh
54 delete_file() {
55     local file="$1"
56     if [ ! -f "$file" ]; then
57         echo "Error: $file does not exist."
58     else
59         rm "$file"
60         echo "Deleted $file."
61     fi
62 }
63
64 operation=$1
65 shift
66
67 case "$operation" in
68     list)
69         list_files
70     ;;
71     count)
72         count_files
73     ;;
74     copy)
75         copy_files
76     ;;
77     move)
78         move_files
79     ;;
80     delete)
81         delete_file
82     ;;
83     *)
84         echo "Invalid operation"
85     ;;
86 esac
87
88 while true; do
89     read -p "Enter operation (list, count, copy, move, delete): " operation
90     shift
91     case "$operation" in
92         list) list_files ;;
93         count) count_files ;;
94         copy) copy_files ;;
95         move) move_files ;;
96         delete) delete_file ;;
97         *) echo "Invalid operation" ;;
98     esac
99     read -p "Do you want to continue (y/n): " i
100     if [ $i != "y" ]; then
101         exit 0
102     fi
103 done

```

```

rps@rps-virtual-machine:~/sunny$ /bin/bash "/home/rps/sunny/filemanagesc.sh"
Usage: /home/rps/sunny/filemanagesc.sh <directory-path> <operation> [additional-arguments]
Operations:
list          - List all files in the directory
count         - Display the total number of files
copy <source> <destination> - Copy a specified file to a new location
move <source> <destination> - Move a specified file to a new location
delete <file>  - Delete a specified file
rps@rps-virtual-machine:~/sunny$ chmod +x filemanagesc.sh
rps@rps-virtual-machine:~/sunny$

```

## 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.

The image shows a Visual Studio Code window titled "fileOperSYScall.cpp - sunny - Visual Studio Code". The Explorer sidebar on the left lists files in a project named "SUNNY", including 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, and ignore.

The main editor displays the code for fileOperSYScall.cpp, which includes a main function with error handling for file closing and deletion, and a successful completion message.

```
7 int main() {
54     if (close(fd) == -1) {
55         std::cerr << "Error: Unable to close file " << filename << ": " << strerror(errno) << std::endl;
56         return EXIT_FAILURE;
57     }
58
59     if (unlink(filename) == -1) {
60         std::cerr << "Error: Unable to delete file " << filename << ": " << strerror(errno) << std::endl;
61         return EXIT_FAILURE;
62     }
63
64     std::cout << "File operations completed successfully." << std::endl;
65     return EXIT_SUCCESS;
66 }
67
68
```

The bottom panel shows the TERMINAL output, indicating the successful compilation and execution of the program.

```
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.