

EXPERIMENT: -1 (INTRO TO LINUX CMDs)

Objective: To study important shell commands to perform various operations. Students will also do a comparative study of Linux and Windows commands.

Lab Exercise Solution(s):

Q1. Explore the file system of a Windows system and a Linux system, and write prime differences.

Ans1.

- **Kernel and Source Code:**

Kernel: Linux uses the Linux kernel, which is open-source and can be customized by users and developers. This means that anyone can view, modify, and distribute the source code, leading to a wide variety of Linux distributions (distros) with different features and purposes.

Windows: Windows uses a closed-source kernel developed by Microsoft. The source code is not available to the public, and users can't modify or redistribute it. Windows is distributed in various editions (e.g., Home, Pro, Enterprise) with different features and pricing.

- **Software and Compatibility:**

Linux: Linux has a vast repository of free and open-source software available through package managers like APT, YUM, or Snap. While many popular software titles are available for Linux, some proprietary software and games may not have official Linux support. However, tools like Wine and Proton (for gaming) enable some Windows software to run on Linux.

Windows: Windows offers a wide range of commercial software, including a robust ecosystem for productivity, gaming, and professional applications. Most software is designed specifically for Windows, making it the go-to choice for many businesses and gamers. However, it may not support Linux software without additional tools or virtualization.

- **User Interface and Configuration:**

Linux: Linux provides various desktop environments (e.g., GNOME, KDE, XFCE) that offer different user interfaces and customization options. Users have more control over system configurations and can tailor their desktop experience to suit their preferences. The command-line interface (CLI) is essential for system administration and offers extensive flexibility.

Windows: Windows has a consistent graphical user interface (GUI) across versions, making it relatively user-friendly and familiar to most people. While Windows allows some customization, it may not offer the same level of flexibility and configurability as Linux, especially in terms of system-level changes.

Q2. Create a file in your Linux system, in your current user's home directory, named as file1.txt. Write your name and Registration number in file1.txt using cat command. Now rename the file using mv command, the new name must be "yourRegistrationNo.txt".

Ans2. **Solution with steps: -**

1. Create a file: - **cat > file1.txt**
2. Write your Registration Number then press **Ctrl + D** to save and exit.
3. To rename the file: - **mv file1.txt yourRegistrationNo.txt**

Screenshots: -

1. `:~$ cat > file1.txt`

2.

```
abhay@abhay-virtual-machine:~$ cat > file1.txt
12201683
#press ctrl + D after writing your registration number
abhay@abhay-virtual-machine:~$
```

3. `:~$ mv file1.txt yourRegistrationNo.txt`

Q3. Create a copy of the file you have created with your registration number. Now delete the original file.

Ans3.

Solution: -

- To create a copy of the file: - **cp** yourRegistrationNo.txt **copiedname.txt**
- To delete the original file: - **rm** yourRegistrationNo.txt

Q.4 Create a copy of the file you have created with your registration number. Now delete the original file

Ans4.

Solution: -

- To create a directory with your name: - **mkdir** Name1
- To move all the files into the directory: - **mv** yourRegistrationNo.txt **Abhay**

Q.5 Create multiple directories using a single command. (Directories name can your friends' name.)

Ans5.

Solution: -

- To create multiple directories: - **mkdir** Name1 Name2 Name3

EXPERIMENT: -2 (SHELL PROGRAMMING)

Objective: To learn the basics of shell programming to use variables, accept input from a user and perform tests and make decisions.

Lab Exercise Solution(s):

Q.1 Write a shell script to produce a multiplication table.

Ans1.

Solution: -

Step1: - touch multiplication_table.sh

Step 2: - nano multiplication_table.sh

Step 3: -

```
#!/bin/bash
```

```
echo "Enter the number for which you want to generate the multiplication table: "  
read number
```

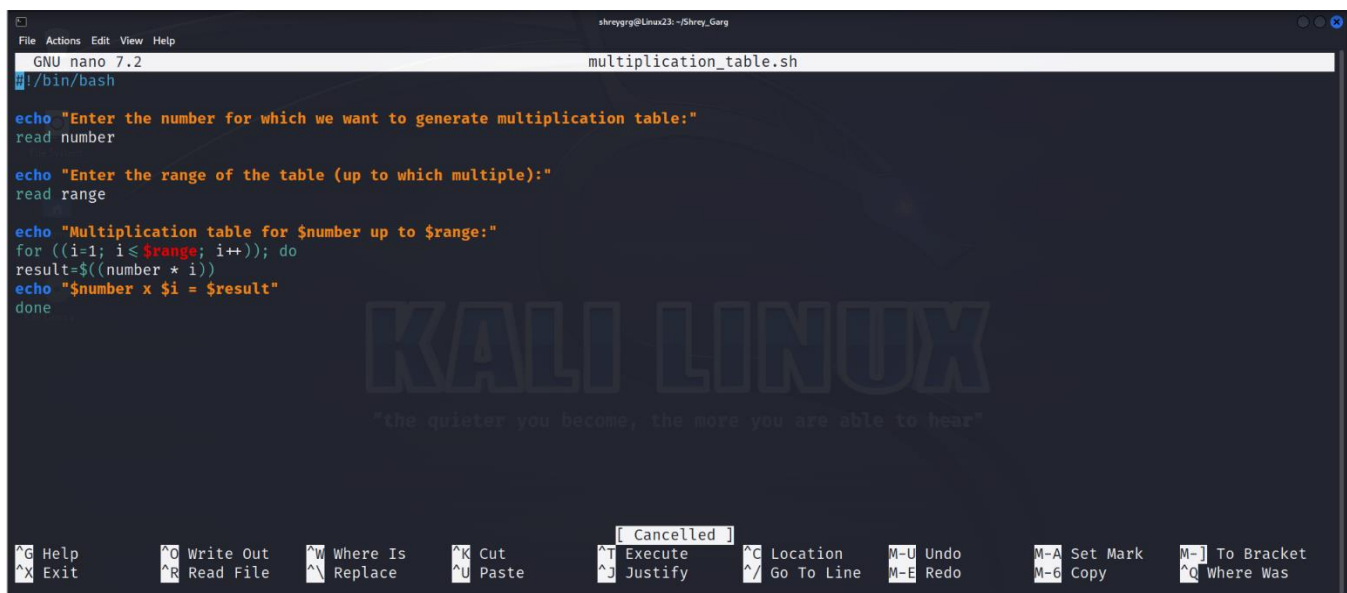
```
echo "Enter the range of the table (up to which you want multiple):"
read range
```

```
echo "Multiplication table for $number up to $range:"
for (( i=1; i<=$range; i++ )); do
    result=$((number * i))
    echo "$number x $i = $result"
done
```

Step 4: - `chmod +x multiplication_table.sh`

Step 5: - `./multiplication_table.sh`

nano multiplication_table.sh



```
GNU nano 7.2 multiplication_table.sh
#!/bin/bash

echo "Enter the number for which we want to generate multiplication table:"
read number

echo "Enter the range of the table (up to which multiple):"
read range

echo "Multiplication table for $number up to $range:"
for ((i=1; i<=$range; i++)); do
    result=$((number * i))
    echo "$number x $i = $result"
done
```

KALI LINUX
"the quieter you become, the more you are able to hear"

[Cancelled]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location	M-U Undo	M-A Set Mark	M-] To Bracket
^X Exit	^R Read File	^_\ Replace	^U Paste	^J Justify	^/_ Go To Line	M-E Redo	M-G Copy	^Q Where Was

Output of the Program

```
(shreygrg@Linux23)-[~/Shrey_Garg]
$ chmod +x multiplication_table.sh

(shreygrg@Linux23)-[~/Shrey_Garg]
$ ./multiplication_table.sh
Enter the number for which we want to generate multiplication table:
6
Enter the range of the table (up to which multiple):
10
Multiplication table for 6 up to 10:
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

Q.2 Write a shell script program to implement a small calculator.

Ans2.

Solution: -

Step 1: - touch calculator.sh

Step 2: - nano calculator.sh

Step 3: -

```
#!/bin/bash
```

```
echo "Select operation:"
```

```
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read choice
```

```
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
```

```
case $choice in
  1)
    result=$((num1 + num2))
    operator="+"
    ;;
  2)
    result=$((num1 - num2))
    operator="-"
    ;;
  3)
    result=$((num1 * num2))
    operator="*"
    ;;
  4)
    if [ $num2 -eq 0 ]; then
      echo "Error: Division by zero"
      exit 1
    fi
    result=$(echo "scale=2; $num1 / $num2" | bc)
    operator="/"
    ;;
  *)
    echo "Invalid choice"
    exit 1
    ;;
esac
echo "$num1 $operator $num2 = $result"
```

Step 4: - `chmod +x calculator.sh`

Step 5: - `./calculator.sh`

nano calculator.sh

```
GNU nano 7.2
#!/bin/bash

echo "Select operation:"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read choice

echo "Enter first number:"
read num1
echo "Enter second number:"
read num2

case $choice in
    1) result=$((num1 + num2))
       operator="+"
       ;;
    2) result=$((num1 - num2))
       operator="-"
       ;;
    3) result=$((num1 * num2))
       operator="*"
       ;;
    4) if [ $num2 -eq 0 ]; then
        echo "Error: Division by zero"
        exit 1
      fi
      result=$(echo "scale=2; $num1 / $num2" | bc)
      operator="/"
      ;;
    *) echo "Invalid choice"
       exit 1
       ;;
esac

echo "$num1 $operator $num2 = $result"
```


Output of the Program

```
(shreygrg@Linux23)-[~/Shrey_Garg]
$ chmod +x calculate.sh

(shreygrg@Linux23)-[~/Shrey_Garg]
$ ./calculate.sh
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
3
Enter first number:
5
Enter second number:
6
5 * 6 = 30
```

Q 3. Write a shell script to display prime numbers up to the given limit.

Ans3. Solution: -

Step 1: - touch prime_number.sh

Step 2: - nano prime_number.sh

Step 3: -

#!/bin/bash

echo "Enter the limit:"

read limit

echo "Prime numbers up to \$limit:"

for ((num = 2; num <= limit; num++)); do

```
is_prime=true
```

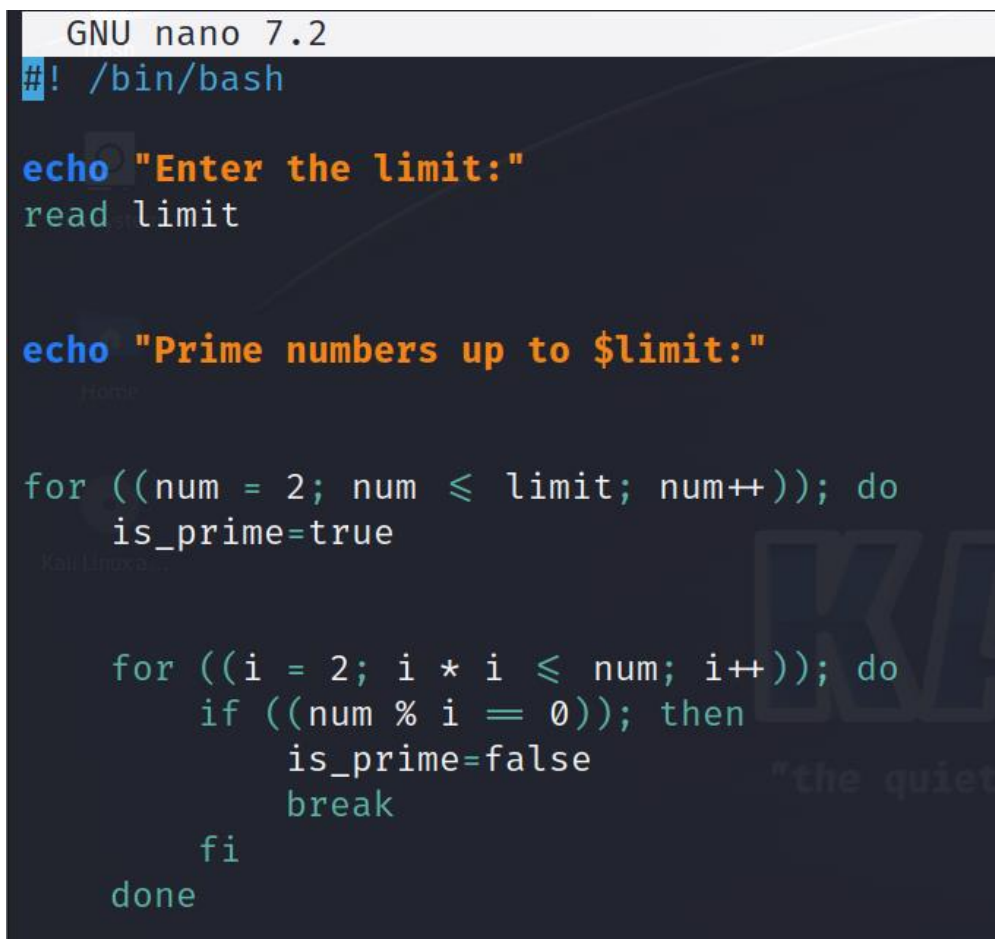
```
for ((i = 2; i * i <= num; i++)); do  
    if ((num % i == 0)); then  
        is_prime=false  
        break  
    fi  
done
```

```
if $is_prime; then  
    echo $num  
fi  
done
```

Step 4: - `chmod +x prime_number.sh`

Step 5: - `./prime_number.sh`

nano prime_number.sh



```
GNU nano 7.2  
#!/bin/bash  
  
echo "Enter the limit:"  
read limit  
  
echo "Prime numbers up to $limit:"  
  
for ((num = 2; num <= limit; num++)); do  
    is_prime=true  
  
    for ((i = 2; i * i <= num; i++)); do  
        if ((num % i == 0)); then  
            is_prime=false  
            break  
        fi  
    done
```

Output of the Program

```
(shreygrg@Linux23)-[~/Shrey_Garg]
$ chmod +x prime_number.sh

(shreygrg@Linux23)-[~/Shrey_Garg]
$ ./prime_number.sh
Enter the limit:
8
Prime numbers up to 8:
2
3
5
7
```

EXPERIMENT: - 3 (File Manipulation and System Calls)

Objective: To learn the working of system calls and the working behind some most used commands in Linux Shell.

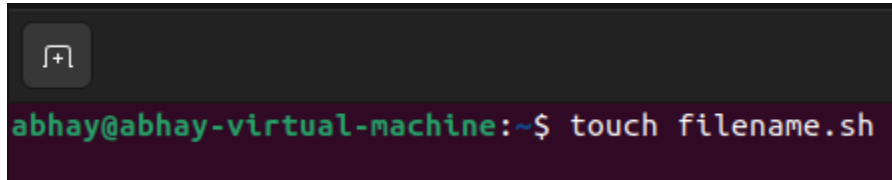
Lab Exercise Solution(s):

Q1. Create a program using system calls to copy either the first half or the second half of a file into a new file.

Ans1.

Step-Wise Solution~

Step-1: Create a file with .sh extension.



```
abhay@abhay-virtual-machine:~$ touch filename.sh
```

Step-2: Open any editor(preferably nano) and edit the file. Press Enter after typing the following command.

```
abhay@abhay-virtual-machine:~$ nano filename.sh
```

Step-3: You'll arrive at the workspace where you're supposed to write your code. Start by typing

#!/bin/bash

Step-4: Type the following solution ~

```
#!/bin/bash

if [ $# -ne 3 ]; then
    echo "Usage: $0 input_file output_file
[first|second]"
    exit 1
fi
```

```
input_file="$1"
output_file="$2"
half_to_copy="$3"

if [ ! -f "$input_file" ]; then
    echo "Error: Input file '$input_file' does
not exist."
    exit 1
fi

input_size=$(wc -c < "$input_file")
half_size=$((input_size / 2))

if [ "$half_to_copy" == "first" ]; then
    dd if="$input_file" of="$output_file" bs=1
count="$half_size" 2>/dev/null
    echo "Copied the first half of '$input_file'
to '$output_file'."
elif [ "$half_to_copy" == "second" ]; then
    dd if="$input_file" of="$output_file" bs=1
skip="$half_size" 2>/dev/null
    echo "Copied the second half of '$input_file'
to '$output_file'."
else
    echo "Error: Invalid option. Use 'first' or
'second'."
```

```
exit 1  
fi
```

Step-5: After writing the code, **Press CTRL + O** and **ENTER** to save the file. Then **CTRL + X** to exit the editor.

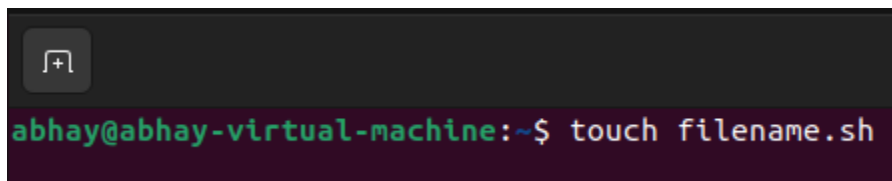
Step-6: In order to run the file, you have to provide the execution permission to the file. To do that, type **chmod +x filename.sh**

Step-7: After providing the permission, you're good to execute the file. Type **./filename.sh** to execute the file.

Q2. Develop a program using system calls to read input from the console until the user inputs '\$', and then save the input into a file.

Ans2. **Step-Wise Solution ~**

Step-1: Create a file with .sh extension.

A terminal window with a dark background. In the top-left corner, there is a small icon of a terminal window with a plus sign. The terminal text shows the user 'abhay' on a machine named 'abhay-virtual-machine' at the home directory, typing the command 'touch filename.sh'.

```
abhay@abhay-virtual-machine:~$ touch filename.sh
```

Step-2: Open any editor(preferably nano) and edit the file.

Press Enter after typing the following command.

```
abhay@abhay-virtual-machine:~$ nano filename.sh
```

Step-3: You'll arrive at the workspace where you're supposed to write your code. Start by typing

#!/bin/bash

Step-4: Type the following solution ~

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Usage: $0 output_file"
    exit 1
fi

output_file="$1"

# Prompt the user and read input until '$' is entered
echo "Enter text (type '$' to save and exit):"
while true; do
    read input
    if [ "$input" = "$" ]; then
        break
    else
        echo "$input" >> "$output_file"
    fi
done
```

```
echo "Input saved to '$output_file'."
```

Step-5: After writing the code, **Press CTRL + O** and **ENTER** to save the file. Then **CTRL + X** to exit the editor.

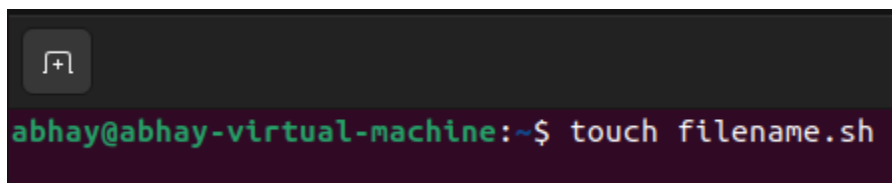
Step-6: In order to run the file, you have to provide the execution permission to the file. To do that, type **chmod +x filename.sh**

Step-7: After providing the permission, you're good to execute the file. Type **./filename.sh** to execute the file.

Q3. Design a program using system calls to read the contents of a file without using a char array and display the contents directly on the console. Please refrain from using built-in functions like `sizeof()` and `strlen()`.

Ans3. **Step-Wise Solution ~**

Step-1: Create a file with .sh extension.



```
abhay@abhay-virtual-machine:~$ touch filename.sh
```


Step-2: Open any editor(preferably nano) and edit the file. Press Enter after typing the following command.

```
abhay@abhay-virtual-machine:~$ nano filename.sh
```

Step-3: You'll arrive at the workspace where you're supposed to write your code. Start by typing

#!/bin/bash

Step-4: Type the following solution ~

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    const char *filename = argv[1];
    int fd = open(filename, O_RDONLY);

    if (fd == -1) {
        perror("open");
        return 1;
    }
}
```

```
char buffer;
ssize_t bytesRead;

while ((bytesRead = read(fd, &buffer, 1)) > 0) {
    if (write(STDOUT_FILENO, &buffer, 1) == -1) {
        perror("write");
        return 1;
    }
}

if (bytesRead == -1) {
    perror("read");
    return 1;
}

close(fd);

return 0;
}
```

Step-5: After writing the code, **Press CTRL + O** and **ENTER** to save the file. Then **CTRL + X** to exit the editor.

Step-6: In order to run the file, you have to provide the execution permission to the file. To do that, type **chmod +x filename.sh**

Step-7: After providing the permission, you're good to execute the file. Type **./filename.sh** to execute the file.

EXPERIMENT: - 4 (Directory Manipulation)

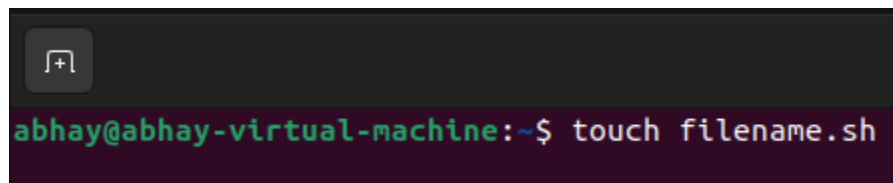
Objective: To learn the working of directory system calls and the working behind some most used commands in Linux.

Solutions:

Q1. Create a program using directory system calls to make a directory on the desktop, then create a file inside that directory and finally list the contents of the directory.

Ans1. Step-Wise Solution ~

Step-1: Create a file with .sh extension.

A terminal window with a dark background. The prompt is 'abhay@abhay-virtual-machine:~\$'. The command 'touch filename.sh' is entered and executed, with the output 'touch filename.sh' displayed in green text.

```
abhay@abhay-virtual-machine:~$ touch filename.sh
```

Step-2: Open any editor(preferably nano) and edit the file. Press Enter after typing the following command.

A terminal window with a dark background. The prompt is 'abhay@abhay-virtual-machine:~\$'. The command 'nano filename.sh' is entered and executed, with the output 'nano filename.sh' displayed in green text.

```
abhay@abhay-virtual-machine:~$ nano filename.sh
```

Step-3: You'll arrive at the workspace where you're supposed to write your code. Start by typing

#!/bin/bash

Step-4: Type the following solution ~

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <dirent.h>
#include <string.h>

int main() {
    const char *desktop_path =
"/home/yourusername/Desktop"; // Replace 'yourusername'
with your actual username
    const char *directory_name = "my_directory";
    const char *file_name = "my_file.txt";

    // Create the directory on the desktop
    char directory_path[256];
    snprintf(directory_path, sizeof(directory_path),
"%s/%s", desktop_path, directory_name);

    if (mkdir(directory_path, 0777) == -1) {
        perror("Error creating directory");
        return 1;
    }
}
```

```
// Create a file inside the directory
char file_path[256];
snprintf(file_path, sizeof(file_path), "%s/%s",
directory_path, file_name);

int fd = open(file_path, O_CREAT | O_WRONLY, 0644);
if (fd == -1) {
    perror("Error creating file");
    return 1;
}

const char *file_content = "This is a sample file.\n";
write(fd, file_content, strlen(file_content));
close(fd);

// List the contents of the directory
DIR *dir = opendir(directory_path);
if (dir == NULL) {
    perror("Error opening directory");
    return 1;
}

printf("Contents of '%s':\n", directory_name);

struct dirent *entry;
while ((entry = readdir(dir)) != NULL) {
    if (strcmp(entry->d_name, ".") != 0 &&
strcmp(entry->d_name, "..") != 0) {
        printf("%s\n", entry->d_name);
    }
}
```

```
    closedir(dir);  
  
    return 0;  
}
```

Step-5: After writing the code, **Press CTRL + O** and **ENTER** to save the file. Then **CTRL + X** to exit the editor.


Step-6: In order to run the file, you have to provide the execution permission to the file. To do that, type **chmod +x filename.sh**

Step-7: After providing the permission, you're good to execute the file. Type **./filename.sh** to execute the file.

Q2. Develop a program using directory and file manipulation system calls to copy the contents of one directory into a newly created directory.

Ans2. **Step-Wise Solution ~**

Step-1: Create a file with .sh extension.



```
abhay@abhay-virtual-machine:~$ touch filename.sh
```

Step-2: Open any editor(preferably nano) and edit the file. Press Enter after typing the following command.

```
abhay@abhay-virtual-machine:~$ nano filename.sh
```

Step-3: You'll arrive at the workspace where you're supposed to write your code. Start by typing

#!/bin/bash

Step-4: Type the following solution ~

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>

void copyDirectory(const char *src, const char *dest) {
    char cmd[256];
    snprintf(cmd, sizeof(cmd), "cp -r %s %s", src, dest);
    system(cmd);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
```

```
        fprintf(stderr, "Usage: %s source_directory  
destination_directory\n", argv[0]);  
        return 1;  
    }  
  
    const char *srcDir = argv[1];  
    const char *destDir = argv[2];  
  
    copyDirectory(srcDir, destDir);  
  
    printf("Contents of '%s' copied to '%s'.\n", srcDir,  
destDir);  
  
    return 0;  
}
```

Step-5: After writing the code, **Press CTRL + O** and **ENTER** to save the file. Then **CTRL + X** to exit the editor.

Step-6: In order to run the file, you have to provide the execution permission to the file. To do that, type **chmod +x filename.sh**

Step-7: After providing the permission, you're good to execute the file. Type **./filename.sh** to execute the file.

Submitted to: -

Mr. Ashish Kumar Singh

Section: K22QY

Submitted By: -

**Shrey Garg
(12218692)**

**Abhay P Aneesh
(12201683)**