

Reg No : 21BCB0107
Name : Shivam Dave
Slot : B2

Exercise 1 – Operating System Basics

Question – I

Demonstrate various Linux commands

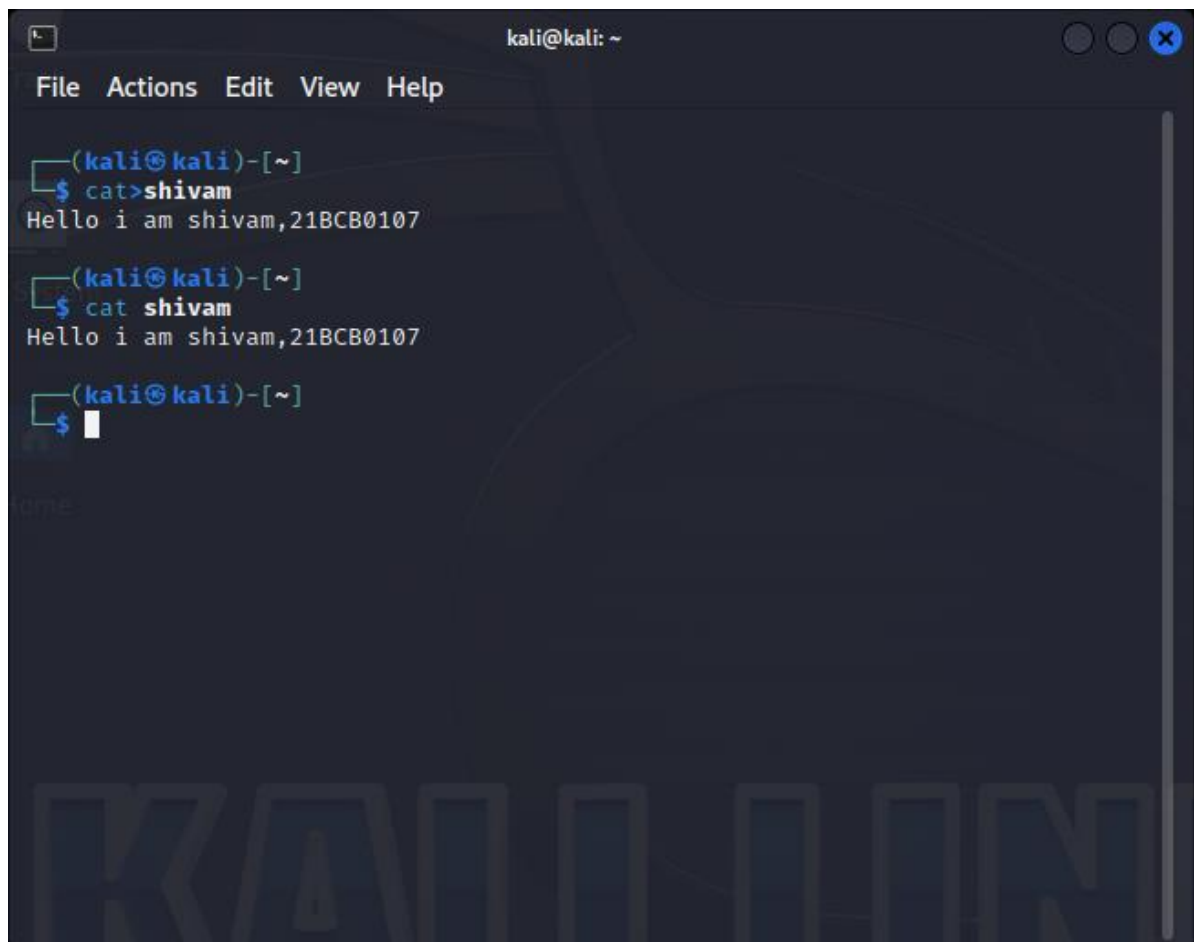
1. Create a file:

```
cat>shivam
```

```
Hello I am shivam,21BCB0107
```

```
cat shivam
```

Output:



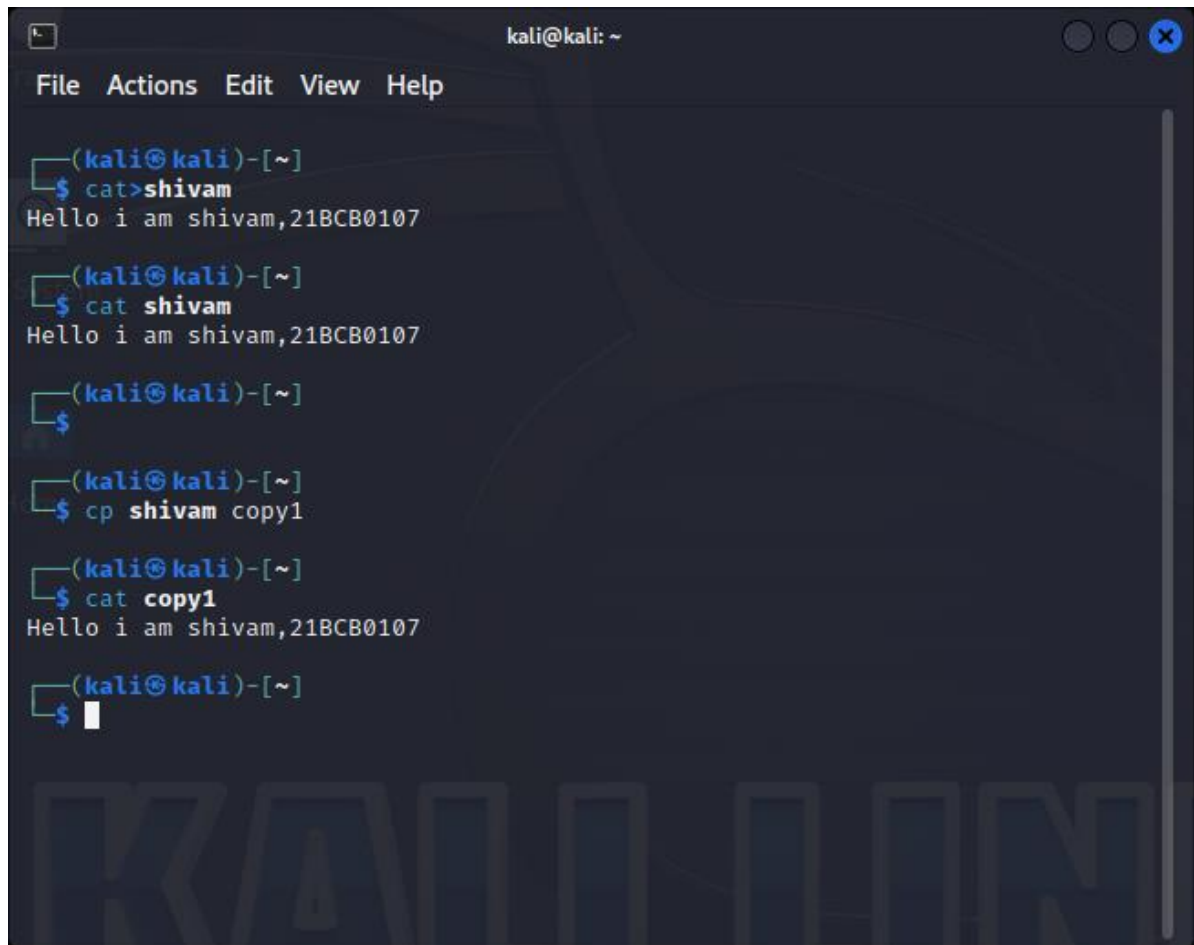
```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ cat>shivam  
Hello i am shivam,21BCB0107  
(kali@kali)-[~]  
$ cat shivam  
Hello i am shivam,21BCB0107  
(kali@kali)-[~]  
$
```

2. Copying file:

```
cat shivam copy1
```

```
cat copy1
```

Output:

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the following commands and output:

```
(kali@kali)-[~]  
$ cat>shivam  
Hello i am shivam,21BCB0107  
  
(kali@kali)-[~]  
$ cat shivam  
Hello i am shivam,21BCB0107  
  
(kali@kali)-[~]  
$  
  
(kali@kali)-[~]  
$ cp shivam copy1  
  
(kali@kali)-[~]  
$ cat copy1  
Hello i am shivam,21BCB0107  
  
(kali@kali)-[~]  
$
```

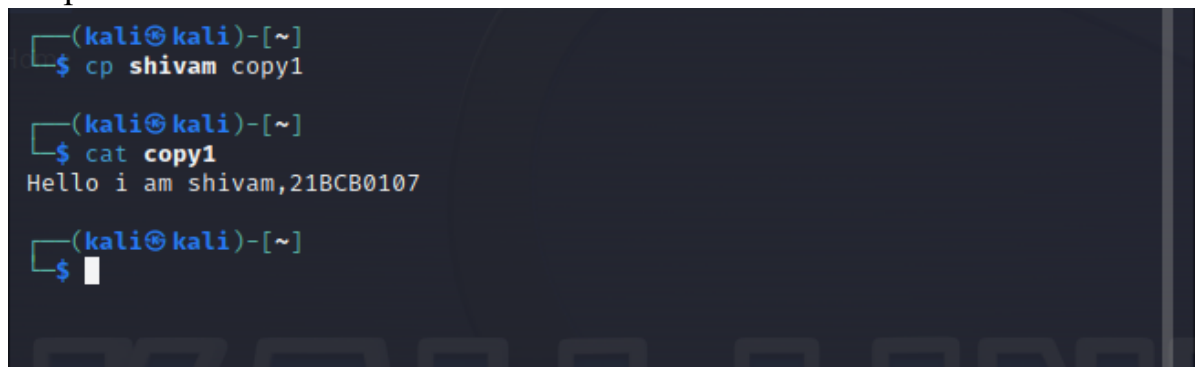
3. Renaming file:

Code:

cp shivam copy1

cat copy1

Output:

A terminal window showing the following commands and output:

```
(kali@kali)-[~]  
$ cp shivam copy1  
  
(kali@kali)-[~]  
$ cat copy1  
Hello i am shivam,21BCB0107  
  
(kali@kali)-[~]  
$
```

4. Removing file:

Code:

rm cp1

cat cp1

Output:

```
(kali㉿kali)-[~]  
$ rm cp1  
  
(kali㉿kali)-[~]  
$ cat cp1  
cat: cp1: No such file or directory  
  
(kali㉿kali)-[~]  
$
```

5. Creating a directory:

Code:

Mkdir dir1

Output:

```
(kali㉿kali)-[~]  
$ mkdir dir1  
  
(kali㉿kali)-[~]  
$
```

6. Moving and copying files into one directory

Code:

Mv shivam dir1

Output:

```
(kali㉿kali)-[~]  
$ mv shivam dir1  
  
(kali㉿kali)-[~]  
$
```

7. Copying a directory

Code:

cp -r dir1 dir1copy

Output:

```
(kali㉿kali)-[~]  
$ cp -r dir1 dir1copy  
  
(kali㉿kali)-[~]  
$
```

8. Renaming a directory

Code:

Mv dir1 d1

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ cat shivam  
cat: shivam: No such file or directory  
(kali@kali)-[~]  
$ cat>shivam  
Hello I am shivam,21BCB0107  
(kali@kali)-[~]  
$ cat shivam  
Hello I am shivam,21BCB0107  
(kali@kali)-[~]  
$ mv shivam dir1  
(kali@kali)-[~]  
$ cp -r d1 d1copy  
cp: cannot stat 'd1': No such file or directory  
(kali@kali)-[~]  
$ cp -r dir1 dir1copy  
(kali@kali)-[~]  
$ mv dir1 d1  
(kali@kali)-[~]  
$
```

9. Removing a directory

Code:

Rmdir d1copy

Rmdir -r d1copy

Output:

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ mv dir1 d1  
  
(kali@kali)-[~]  
$ rmdir d1copy  
rmdir: failed to remove 'd1copy': No such file or directory  
  
(kali@kali)-[~]  
$ rm -d1 d1copy  
rm: invalid option -- '1'  
Try 'rm --help' for more information.  
  
(kali@kali)-[~]  
$ cp -r d1 d1copy  
  
(kali@kali)-[~]  
$ rmdir d1copy  
rmdir: failed to remove 'd1copy': Directory not empty  
  
(kali@kali)-[~]  
$ rmdir -r d1copy  
rmdir: invalid option -- 'r'  
Try 'rmdir --help' for more information.  
  
(kali@kali)-[~]  
$
```

10. Listing a directory's contents

Code:

d1

ls

Output:

```
kali@kali: ~/d1
File Actions Edit View Help

(kali@kali)-[~]
$ rm -d1 d1copy
rm: invalid option -- '1'
Try 'rm --help' for more information.

(kali@kali)-[~]
$ cp -r d1 d1copy

(kali@kali)-[~]
$ rmdir d1copy
rmdir: failed to remove 'd1copy': Directory not empty

(kali@kali)-[~]
$ rmdir -r d1copy
rmdir: invalid option -- 'r'
Try 'rmdir --help' for more information.

(kali@kali)-[~]
$ d1

(kali@kali)-[~/d1]
$ ls
shivam

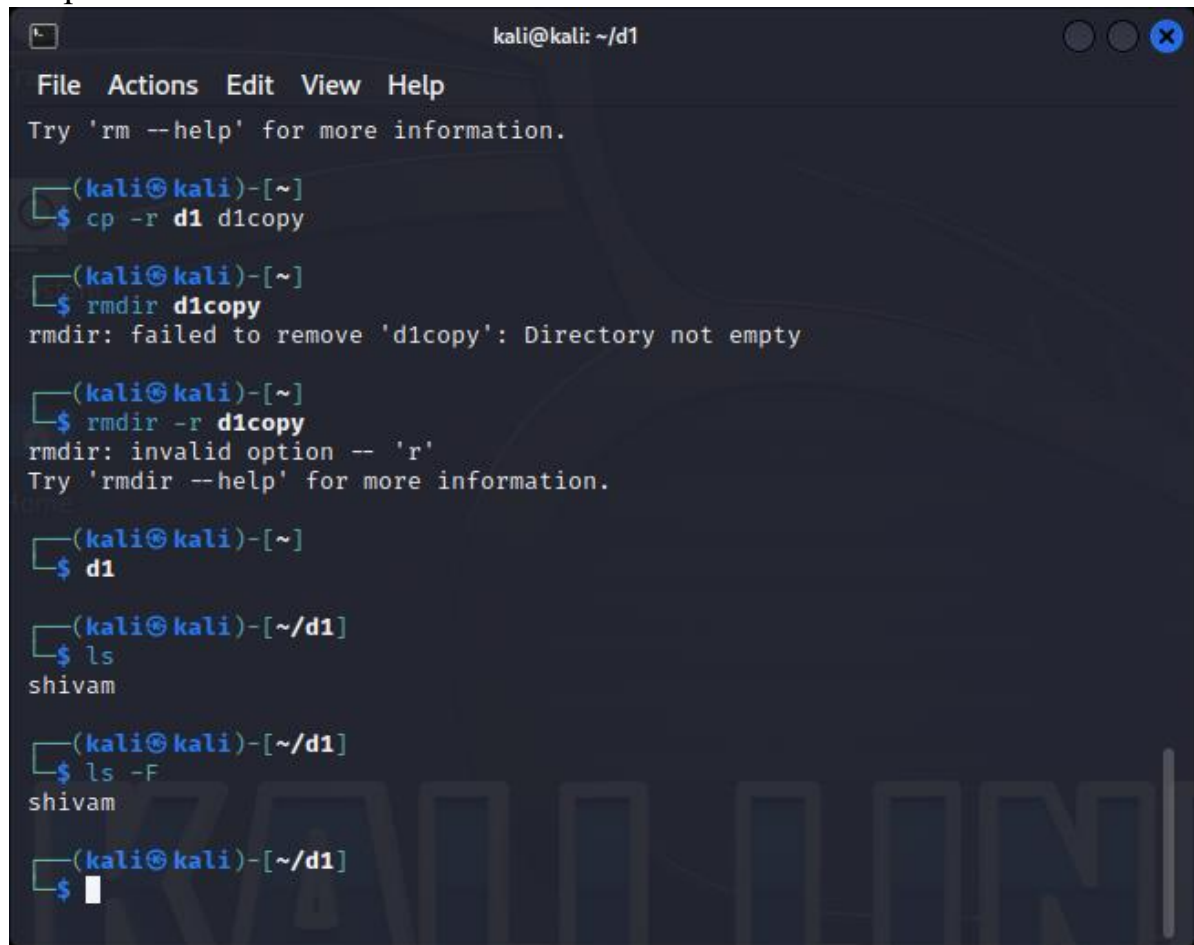
(kali@kali)-[~/d1]
$
```

11) Using options on the ls command

Code:

Ls -F

Output:

A terminal window titled 'kali@kali: ~/d1' with a menu bar (File, Actions, Edit, View, Help). The terminal shows a series of commands and their outputs. First, 'cp -r d1 d1copy' is executed. Then, 'rmdir d1copy' fails with the message 'rmdir: failed to remove 'd1copy': Directory not empty'. Next, 'rmdir -r d1copy' fails with 'rmdir: invalid option -- 'r'' and a suggestion to try 'rmdir --help'. Then, 'd1' is entered, changing the directory to ~/d1. 'ls' is run, showing 'shivam'. Finally, 'ls -F' is run, also showing 'shivam'. The prompt is currently '\$' in the ~/d1 directory.

```
kali@kali: ~/d1
File Actions Edit View Help
Try 'rm --help' for more information.

(kali@kali)-[~]
$ cp -r d1 d1copy

(kali@kali)-[~]
$ rmdir d1copy
rmdir: failed to remove 'd1copy': Directory not empty

(kali@kali)-[~]
$ rmdir -r d1copy
rmdir: invalid option -- 'r'
Try 'rmdir --help' for more information.

(kali@kali)-[~]
$ d1

(kali@kali)-[~/d1]
$ ls
shivam

(kali@kali)-[~/d1]
$ ls -F
shivam

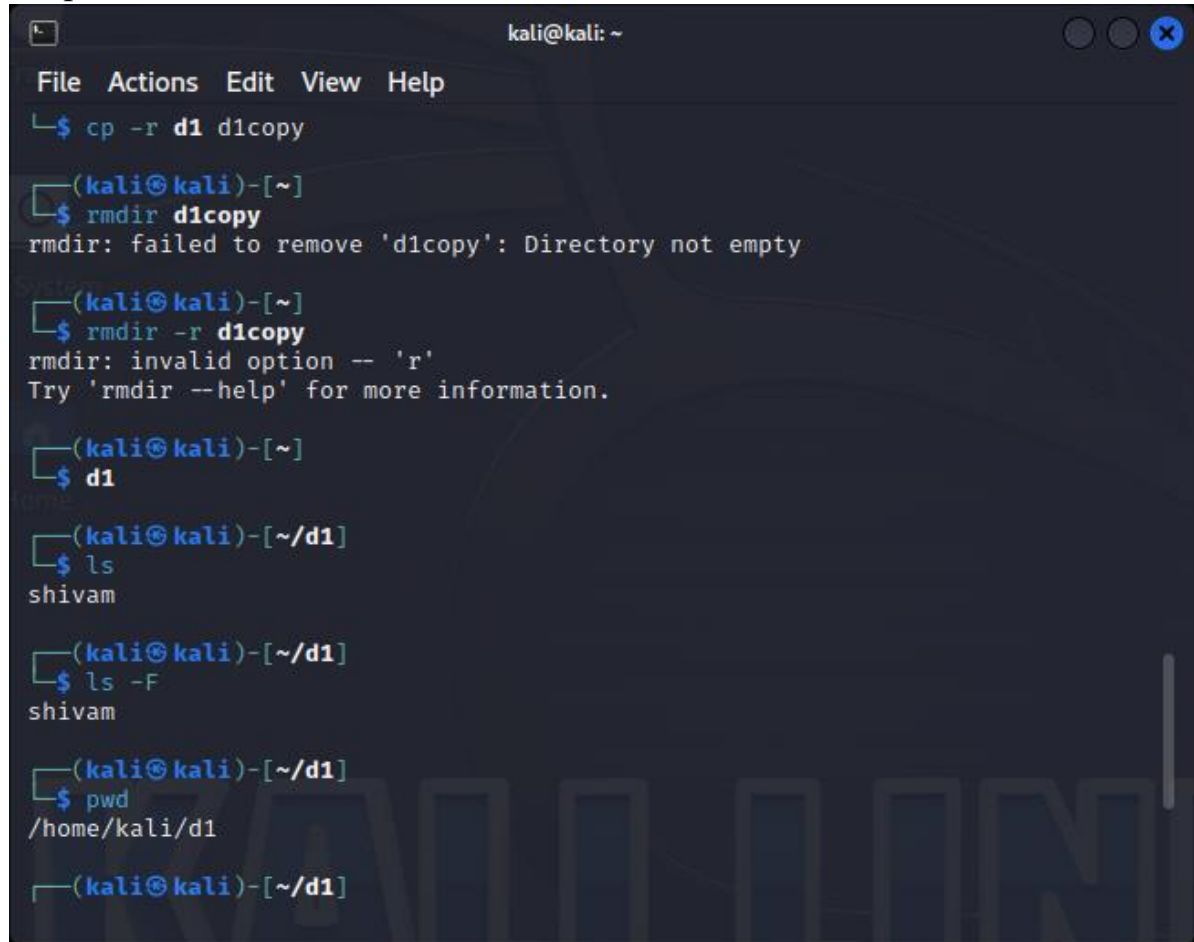
(kali@kali)-[~/d1]
$
```

12) Determining which directory you are currently in.

Code:

pwd

Output:

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The terminal shows a sequence of commands and their outputs. First, 'cp -r d1 d1copy' is executed. Then, 'rmdir d1copy' fails with the message 'rmdir: failed to remove 'd1copy': Directory not empty'. Next, 'rmdir -r d1copy' fails with 'rmdir: invalid option -- 'r'. Try 'rmdir --help' for more information.'. Then, 'd1' is entered. The prompt changes to '~/.d1', where 'ls' shows 'shivam'. Then 'ls -F' also shows 'shivam'. Finally, 'pwd' shows '/home/kali/d1'.

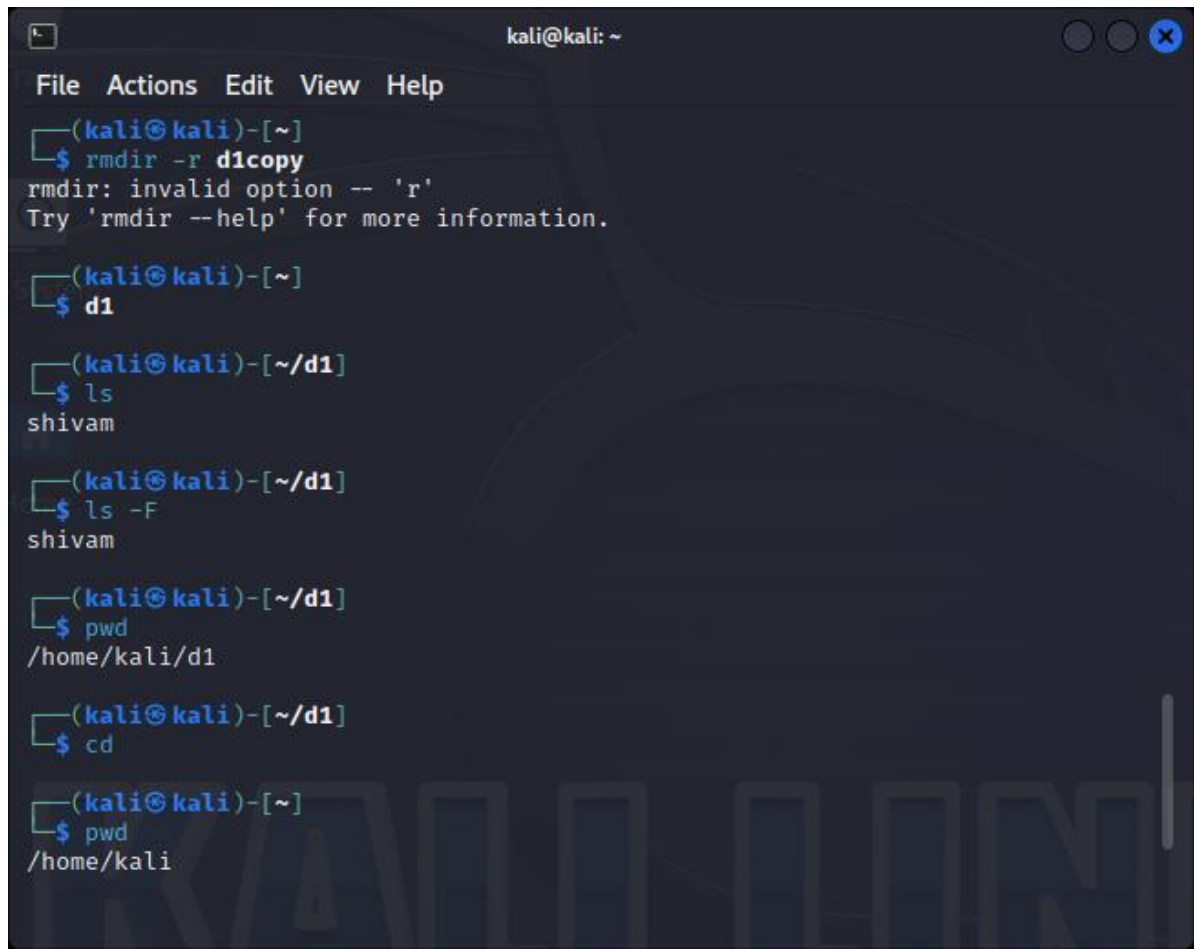
```
kali@kali: ~  
File Actions Edit View Help  
└─$ cp -r d1 d1copy  
  
(kali@kali)-[~]  
└─$ rmdir d1copy  
rmdir: failed to remove 'd1copy': Directory not empty  
  
(kali@kali)-[~]  
└─$ rmdir -r d1copy  
rmdir: invalid option -- 'r'  
Try 'rmdir --help' for more information.  
  
(kali@kali)-[~]  
└─$ d1  
  
(kali@kali)-[~/d1]  
└─$ ls  
shivam  
  
(kali@kali)-[~/d1]  
└─$ ls -F  
shivam  
  
(kali@kali)-[~/d1]  
└─$ pwd  
/home/kali/d1  
  
(kali@kali)-[~/d1]
```

13) Moving from one directory to another

Code:

cd
pwd

Output:

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The terminal shows a sequence of commands and their outputs: 1. Command: `rmkdir -r d1copy`; Output: `rmkdir: invalid option -- 'r'`, `Try 'rmkdir --help' for more information.` 2. Command: `d1` 3. Prompt: `(kali@kali)-[~/d1]` 4. Command: `ls`; Output: `shivam` 5. Command: `ls -F`; Output: `shivam` 6. Prompt: `(kali@kali)-[~/d1]` 7. Command: `pwd`; Output: `/home/kali/d1` 8. Prompt: `(kali@kali)-[~/d1]` 9. Command: `cd` 10. Prompt: `(kali@kali)-[~]` 11. Command: `pwd`; Output: `/home/kali`

14)Directory abbreviations for faster navigation.

Code:

`cd`

`cd d1`

`cd ~`

`pwd`

Output:

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~/d1]  
$ pwd  
/home/kali/d1  
  
(kali@kali)-[~/d1]  
$ cd  
  
(kali@kali)-[~]  
$ pwd  
/home/kali  
  
(kali@kali)-[~]  
$ cd  
  
(kali@kali)-[~]  
$ cd d1  
  
(kali@kali)-[~/d1]  
$ cd ~  
  
(kali@kali)-[~]  
$ pwd  
/home/kali  
  
(kali@kali)-[~]  
$
```

15) Naming and deleting a variable

Code: read Name

“SHIVAM DAVE”

echo “Hello, \$Name”

unset Name

echo \$Name

Output:

```
kali@kali: ~  
File Actions Edit View Help  
system  
(kali@kali)-[~]  
$ read Name  
SHIVAM DAVE  
  
(kali@kali)-[~]  
$ echo "Hello, $Name"  
Hello, SHIVAM DAVE  
  
(kali@kali)-[~]  
$ unset Name  
  
(kali@kali)-[~]  
$ echo "$Name"  
  
KALI LINUX  
"the quieter you become, the more you are a"
```

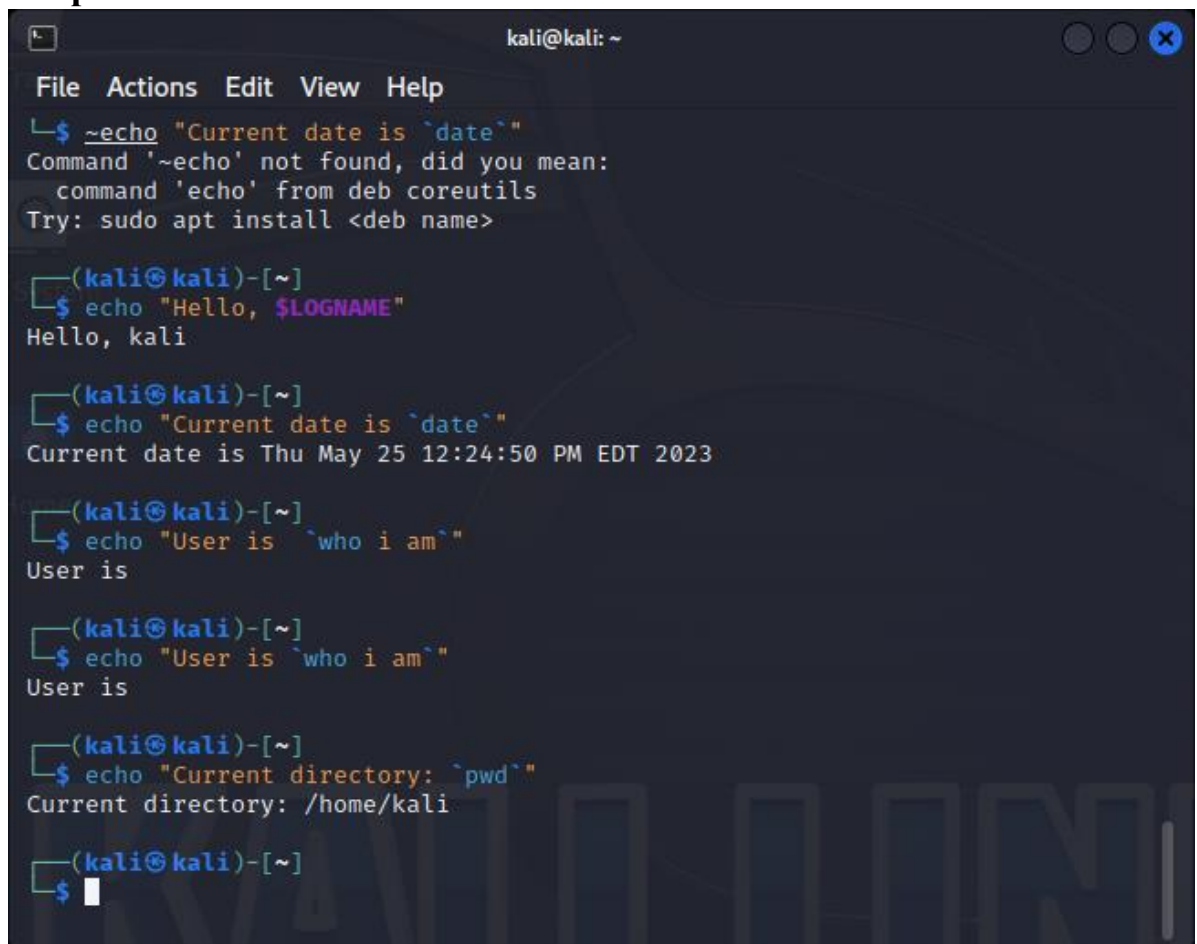
QUESTION– II – Shell Scripts

1. Write Script to see current date, time, username, and current directory

Source code:

```
echo "Hello, $LOGNAME"  
echo "Current date is `date`"  
echo "User is `who i am`"  
echo "Current directory `pwd`"
```

Output:



The screenshot shows a terminal window titled 'kali@kali: ~'. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal content shows the following sequence of commands and outputs:

```
kali@kali: ~  
File Actions Edit View Help  
$ ~echo "Current date is `date`"  
Command '~echo' not found, did you mean:  
  command 'echo' from deb coreutils  
Try: sudo apt install <deb name>  
  
(kali@kali)-[~]  
$ echo "Hello, $LOGNAME"  
Hello, kali  
  
(kali@kali)-[~]  
$ echo "Current date is `date`"  
Current date is Thu May 25 12:24:50 PM EDT 2023  
  
(kali@kali)-[~]  
$ echo "User is `who i am`"  
User is  
  
(kali@kali)-[~]  
$ echo "User is `who i am`"  
User is  
  
(kali@kali)-[~]  
$ echo "Current directory: `pwd`"  
Current directory: /home/kali  
  
(kali@kali)-[~]  
$
```

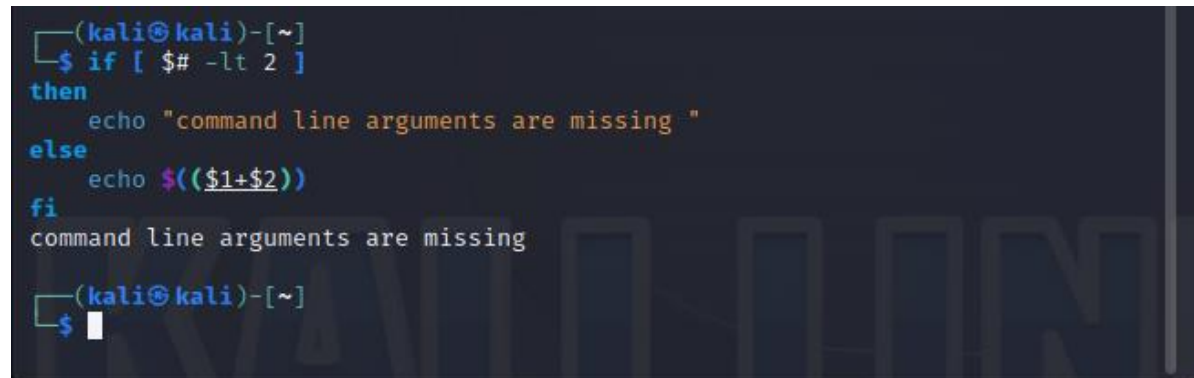
2. How to write shell script that will add two numbers, which are supplied as command line argument, and if this two numbers are not given show error and its usage.

Source Code:

```
if [ $# -lt 2 ]  
then
```

```
    echo "command line arguments are missing "
else
    echo $((($1+$2))
fi
```

Output:



```
(kali@kali)-[~]
$ if [ $# -lt 2 ]
then
    echo "command line arguments are missing "
else
    echo $((($1+$2))
fi
command line arguments are missing
(kali@kali)-[~]
$
```

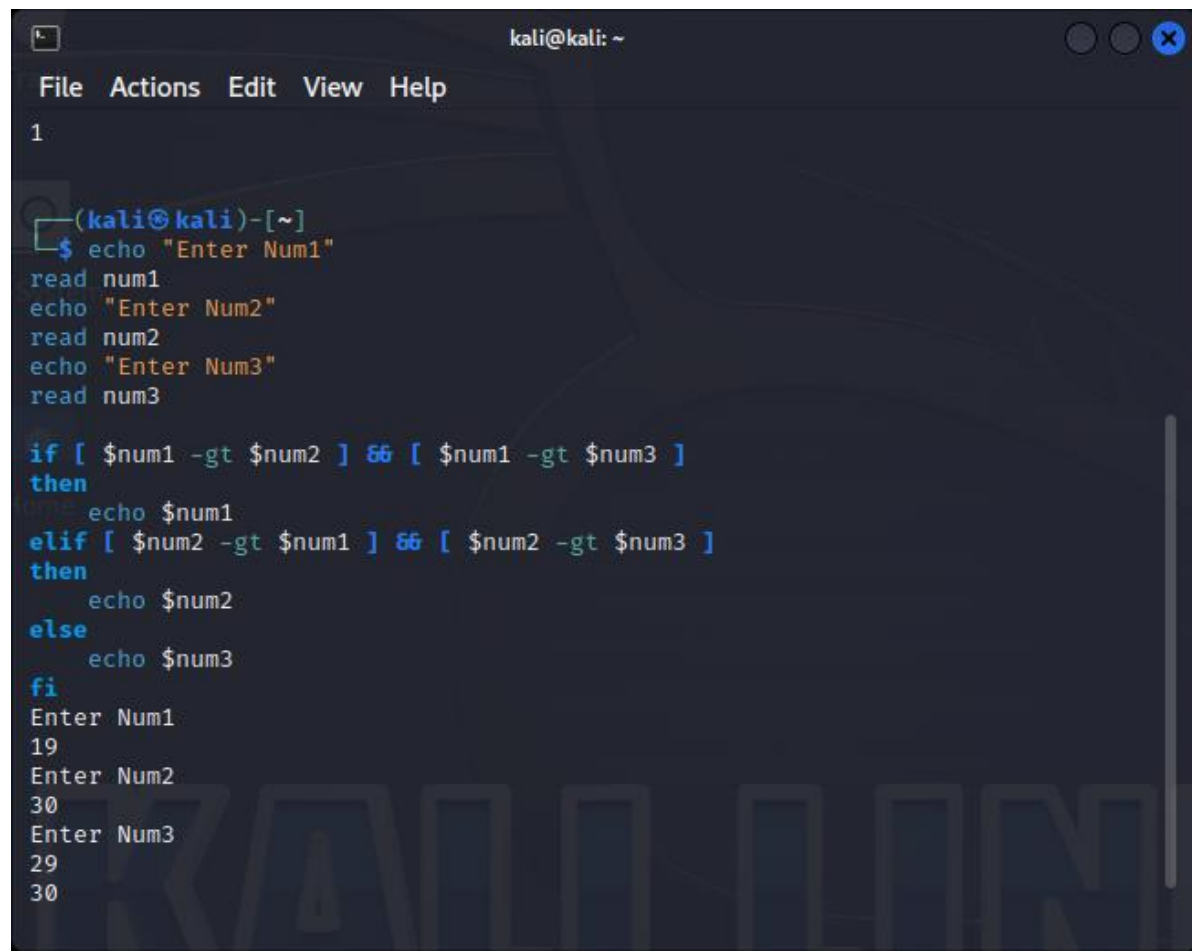
3. **Write Script to find out biggest number from given three nos.**
Numbers are supplied as command line argument. Print error if sufficient arguments are not supplied.

Source Code:

```
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3
```

```
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo $num2
else
    echo $num3
fi
```

Output:



```
kali@kali: ~  
File Actions Edit View Help  
1  
  
(kali@kali)-[~]  
$ echo "Enter Num1"  
read num1  
echo "Enter Num2"  
read num2  
echo "Enter Num3"  
read num3  
  
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]  
then  
    echo $num1  
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]  
then  
    echo $num2  
else  
    echo $num3  
fi  
Enter Num1  
19  
Enter Num2  
30  
Enter Num3  
29  
30
```

4. Write script to print the following numbers as 5,4,3,2,1 using while loop.

Source Code:

```
i=5  
while test $i != 0  
do  
    echo "$i  
"  
    i=`expr $i - 1`  
done
```

Output:

A terminal window titled 'kali@kali: ~' with a menu bar (File, Actions, Edit, View, Help). The script being executed is a while loop that starts with 'i=5' and 'while test \$i != 0'. Inside the loop, it echoes the value of 'i' and then decrements it by 1 using 'i=`expr \$i - 1``'. The output shows the numbers 5, 4, 3, 2, and 1 in sequence. The prompt returns to '(kali@kali)-[~]' after the loop completes.

```
(kali@kali)-[~]  
$ i=5  
while test $i != 0  
do  
    echo "$i"  
    i=`expr $i - 1`  
done  
5  
4  
3  
2  
1  
  
(kali@kali)-[~]  
$
```

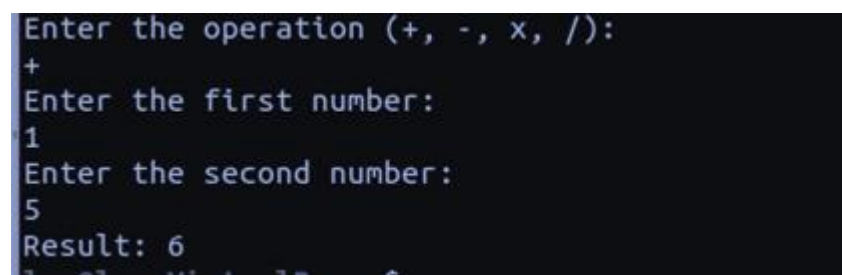
5. Write Script, using case statement to perform basic math operation as follows: + addition, - subtraction, x multiplication, / division

Source Code:

```
# Perform the math operation based on the operator  
case $operator in  
    "+")  
        result=$((number1 + number2))  
        echo "Result: $result"  
        ;;  
    "-")  
        result=$((number1 - number2))  
        echo "Result: $result"  
        ;;  
    "x")  
        result=$((number1 * number2))  
        echo "Result: $result"  
        ;;  
    "/")  
        if [ $number2 -eq 0 ]; then  
            echo "Error: Division by zero is not allowed."  
        fi  
        ;;  
    *)  
        echo "Invalid operator"  
        ;;  
esac
```

```
        else
            result=$((number1 / number2))
            echo "Result: $result"
        fi
    ;;
*)
    echo "Error: Invalid operator."
    ;;
Esac
```

OUTPUT:



```
Enter the operation (+, -, x, /):
+
Enter the first number:
1
Enter the second number:
5
Result: 6
```


Reg No : 21BCB0107
Name : Shivam Dave
Slot : B2

QUESTION 3 – System Calls

1. Write Programs using the following system calls of LINUX operating system:

a. fork, exec, getpid, exit, wait, close, opendir, readdir,

Source Code:

Using ‘fork’ and ‘exec’:

Code:

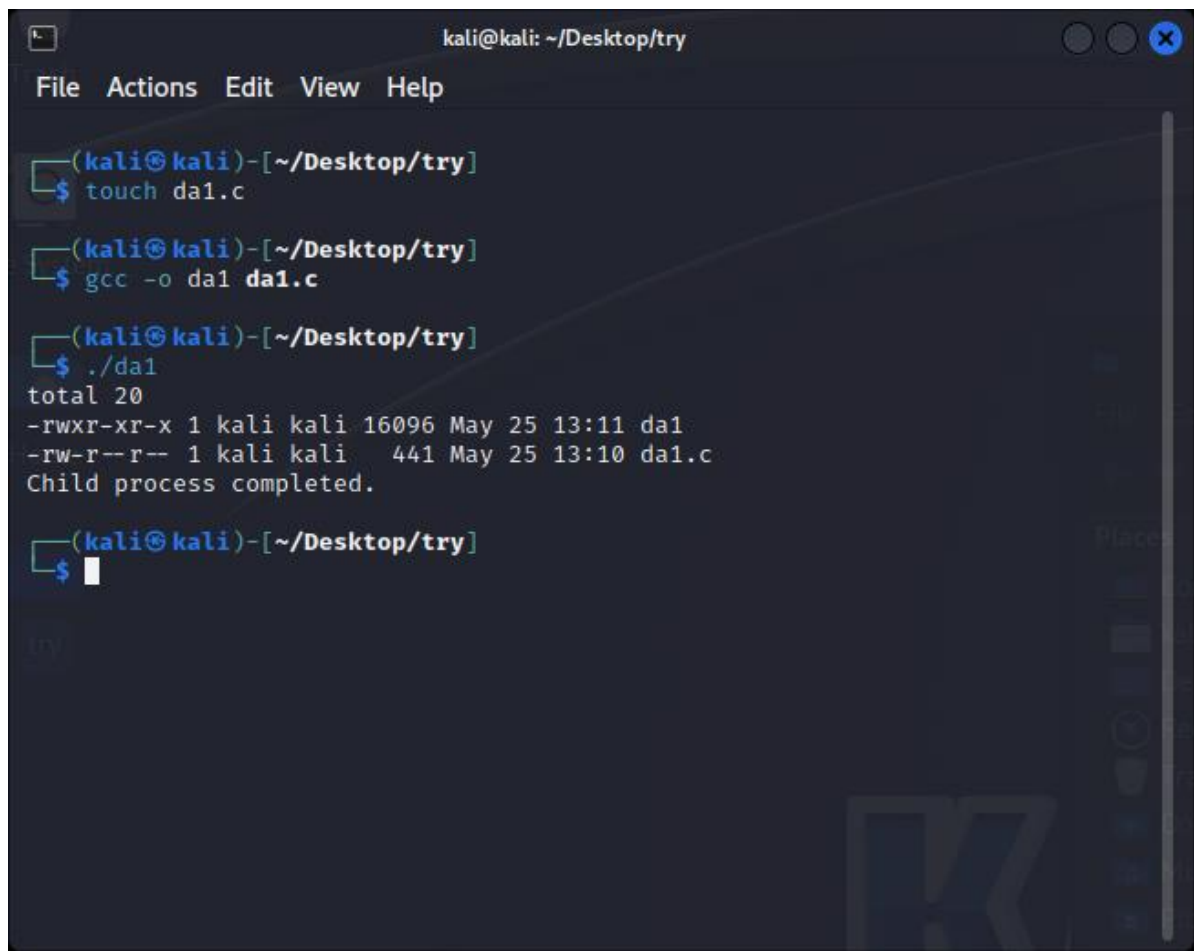
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // Child process
        execl("/bin/ls", "ls", "-l", NULL);
    } else if (pid > 0) {
        // Parent process
        wait(NULL);
        printf("Child process completed.\n");
    } else {
        // Fork failed
        printf("Fork failed.\n");
        return 1;
    }

    return 0;
}
```

Output:

A terminal window titled 'kali@kali: ~/Desktop/try' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the following commands and output:

```
(kali@kali)-[~/Desktop/try]
$ touch da1.c

(kali@kali)-[~/Desktop/try]
$ gcc -o da1 da1.c

(kali@kali)-[~/Desktop/try]
$ ./da1
total 20
-rwxr-xr-x 1 kali kali 16096 May 25 13:11 da1
-rw-r--r-- 1 kali kali 441 May 25 13:10 da1.c
Child process completed.

(kali@kali)-[~/Desktop/try]
$
```

Using getpid and exit:

Code:

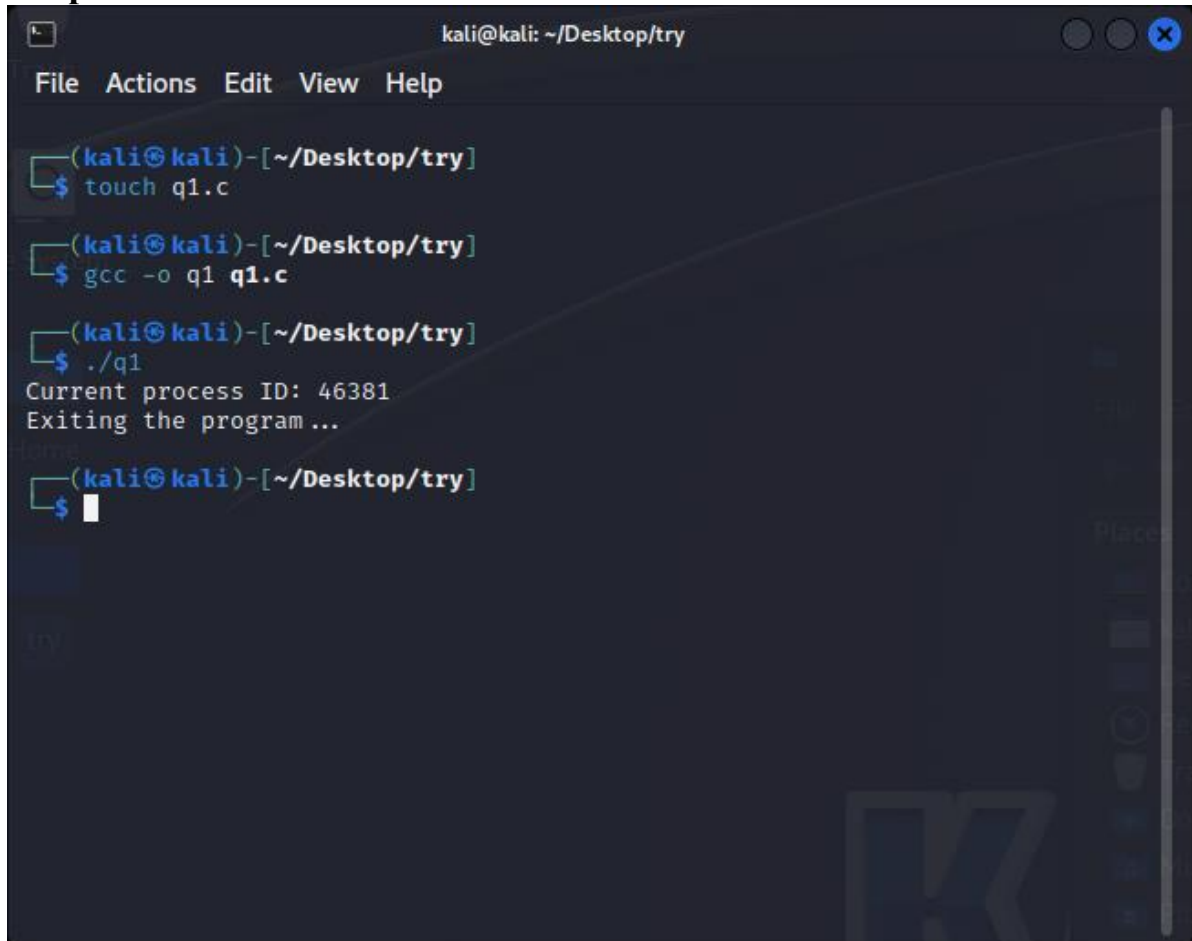
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
```

```
int main() {
    pid_t pid = getpid();

    printf("Current process ID: %d\n", pid);
    printf("Exiting the program...\n");

    exit(0);
}
```

Output:

A terminal window titled 'kali@kali: ~/Desktop/try' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the following commands and output:

```
(kali@kali)-[~/Desktop/try]
$ touch q1.c
(kali@kali)-[~/Desktop/try]
$ gcc -o q1 q1.c
(kali@kali)-[~/Desktop/try]
$ ./q1
Current process ID: 46381
Exiting the program...
(kali@kali)-[~/Desktop/try]
$
```

Using 'wait'

Source Code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
```

```
int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // Child process
        printf("Child process executing...\n");
        sleep(2);
        printf("Child process completed.\n");
    } else if (pid > 0) {
        // Parent process
        printf("Parent process waiting for child to complete...\n");
        wait(NULL);
    }
}
```

```

    printf("Parent process resumed.\n");
} else {
    // Fork failed
    printf("Fork failed.\n");
    return 1;
}

return 0;
}

```

Output:



```

(kali@kali)-[~/Desktop/try]
$ touch q12.c

(kali@kali)-[~/Desktop/try]
$ gcc -o q12 q12.c

(kali@kali)-[~/Desktop/try]
$ ./q12
Parent process waiting for child to complete...
Child process executing...
Child process completed.
Parent process resumed.

(kali@kali)-[~/Desktop/try]
$

```

Using 'close'

Source Code:

```

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

int main() {
    int fd = open("file.txt", O_RDONLY);

    if (fd == -1) {
        printf("Failed to open the file.\n");
        return 1;
    }

    printf("File opened successfully.\n");

    if (close(fd) == -1) {
        printf("Failed to close the file.\n");
    }
}

```

```

        return 1;
    }

    printf("File closed successfully.\n");

    return 0;
}

```

Output:

```

kali@kali: ~/Desktop/try
File Actions Edit View Help
Parent process waiting for child to complete...
Child process executing...
Child process completed.
Parent process resumed.

(kali@kali)-[~/Desktop/try]
$ touch q13.c

(kali@kali)-[~/Desktop/try]
$ gcc -o q13 q13.c

(kali@kali)-[~/Desktop/try]
$ ./q13
Failed to open the file.

(kali@kali)-[~/Desktop/try]
$ touch q13.c

(kali@kali)-[~/Desktop/try]
$ gcc -o q13 q13.c

(kali@kali)-[~/Desktop/try]
$ ./q13
Failed to open the file.

(kali@kali)-[~/Desktop/try]
$

```

Using 'opendir' and 'readdir':

Source Code:

```

#include <stdio.h>
#include <dirent.h>

int main() {
    DIR *dir = opendir(".");

    if (dir == NULL) {
        printf("Failed to open directory.\n");
        return 1;
    }
}

```

```

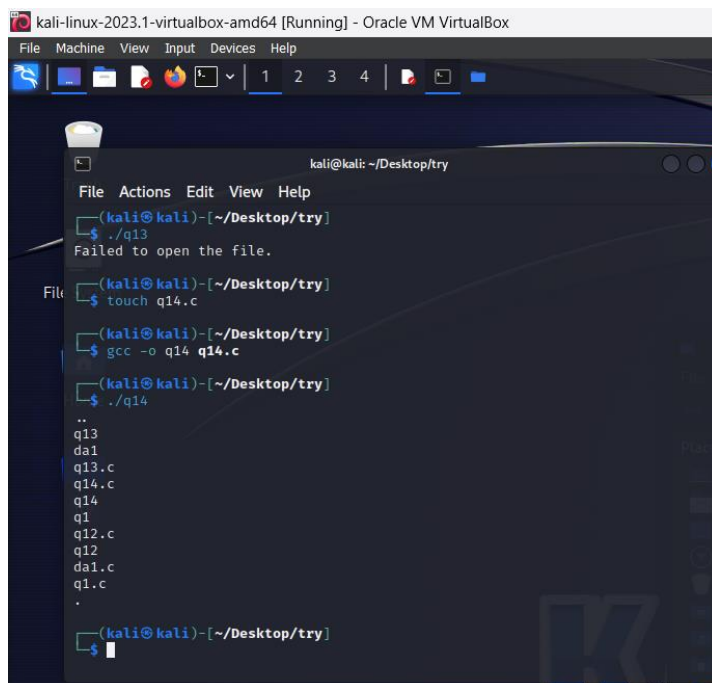
}

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

closedir(dir);

return 0;
}
Output:

```



```

kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4

kali@kali: ~/Desktop/try
File Actions Edit View Help
(kali@kali)-[~/Desktop/try]
$ ./q13
Failed to open the file.
(kali@kali)-[~/Desktop/try]
$ touch q14.c
(kali@kali)-[~/Desktop/try]
$ gcc -o q14 q14.c
(kali@kali)-[~/Desktop/try]
$ ./q14
..
q13
da1
q13.c
q14.c
q14
q1
q12.c
q12
da1.c
q1.c
.
(kali@kali)-[~/Desktop/try]
$

```

2. Write Programs using I/O system calls of LINUX operating system: open, read, write etc.

Source Code:

```

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

#define BUF_SIZE 4096

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

```

```

    return 1;
}

int fd = open(argv[1], O_RDONLY);
if (fd == -1) {
    perror("Failed to open file");
    return 1;
}

char buffer[BUF_SIZE];
ssize_t bytes_read;

while ((bytes_read = read(fd, buffer, BUF_SIZE)) > 0) {
    // Process or display the read data
    write(STDOUT_FILENO, buffer, bytes_read);
}

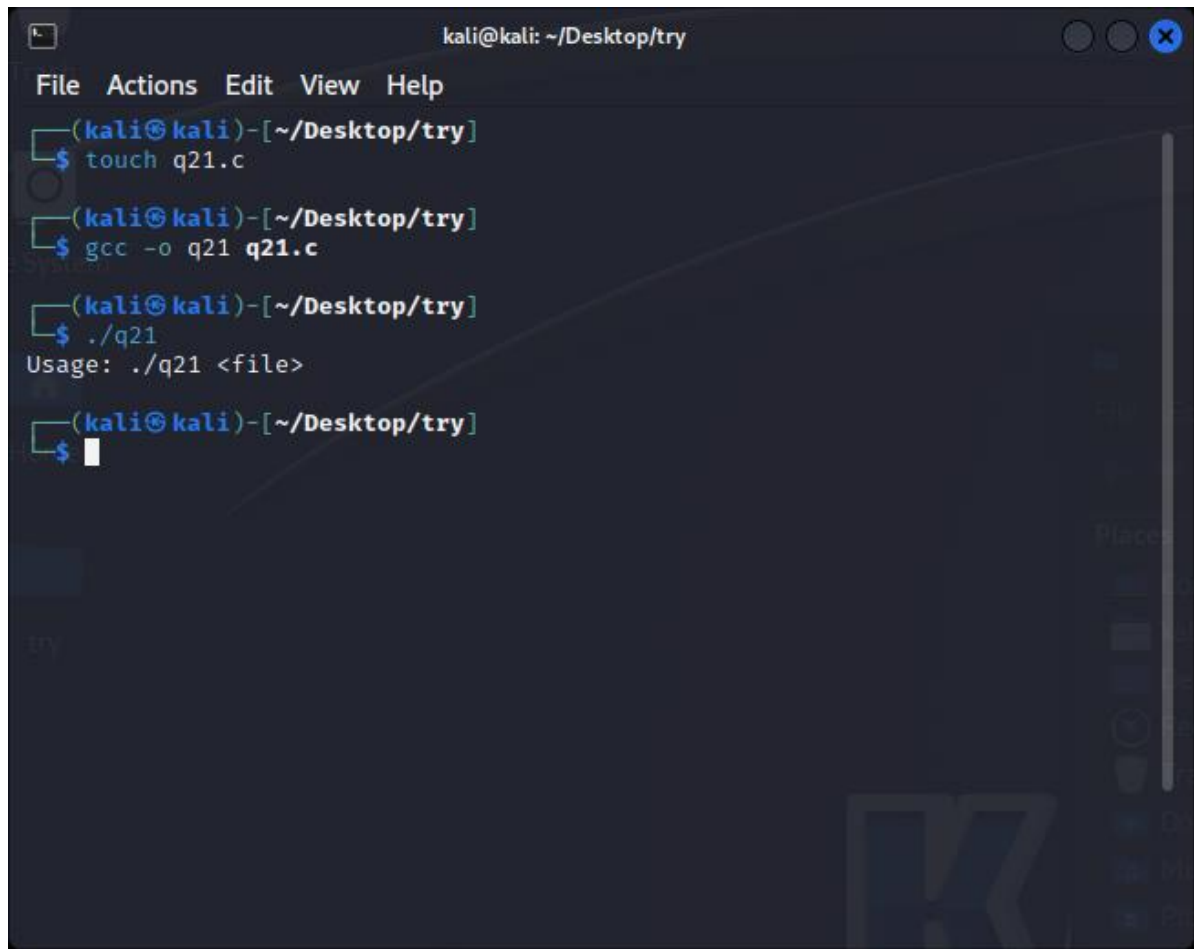
if (bytes_read == -1) {
    perror("Read error");
    return 1;
}

close(fd);

return 0;
}

```

Output:

A terminal window titled 'kali@kali: ~/Desktop/try' with a menu bar (File, Actions, Edit, View, Help). The terminal shows the following commands and output:

```
(kali@kali)-[~/Desktop/try]
$ touch q21.c
(kali@kali)-[~/Desktop/try]
$ gcc -o q21 q21.c
(kali@kali)-[~/Desktop/try]
$ ./q21
Usage: ./q21 <file>
(kali@kali)-[~/Desktop/try]
$
```

b) write()

Source Code:

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

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <file> <text>\n", argv[0]);
        return 1;
    }

    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0666);
    if (fd == -1) {
        perror("Failed to open file");
        return 1;
    }

    const char *text = argv[2];
```



```

ssize_t bytes_written = write(fd, text, strlen(text));

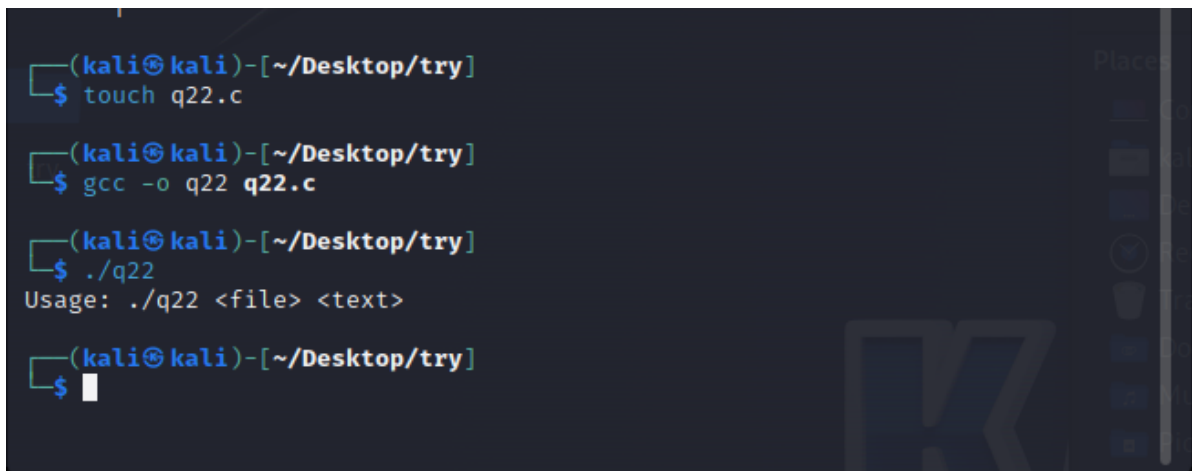
if (bytes_written == -1) {
    perror("Write error");
    return 1;
}

close(fd);

return 0;
}

```

Output:



```

(kali㉿kali)-[~/Desktop/try]
$ touch q22.c

(kali㉿kali)-[~/Desktop/try]
$ gcc -o q22 q22.c

(kali㉿kali)-[~/Desktop/try]
$ ./q22
Usage: ./q22 <file> <text>

(kali㉿kali)-[~/Desktop/try]
$

```

3. Write Programs using C to simulate LINUX commands

a. ls command,

Source code:

```

#include <stdio.h>
#include <dirent.h>

int main() {
    struct dirent *entry;
    DIR *directory = opendir(".");

    if (directory == NULL) {
        printf("Error opening directory.\n");
        return 1;
    }

    while ((entry = readdir(directory)) != NULL) {
        printf("%s\n", entry->d_name);
    }
}

```

```

    closedir(directory);
    return 0;
}

```

Output:

```

kali@kali: ~/Desktop/try
❏ gcc -o q22 q22.c
kali@kali: ~/Desktop/try
❏ ./q22
Usage: ./q22 <file> <text>
kali@kali: ~/Desktop/try
❏ gcc -o q21 q21.c
kali@kali: ~/Desktop/try
❏ gcc -o q22 q21.c
kali@kali: ~/Desktop/try
❏ ./q22
q21
q22.c
q23
q25
q21
q23.c
q24.c
q24
q21
q22.c
q22
q21.c
q22
q23.c
q21.c
q21.c

```

b. grep command

Source Code:

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

#define MAX_LINE_LENGTH 1000

```

```

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: grep <pattern> <filename>\n");
        return 1;
    }
}

```

```

char *pattern = argv[1];
char *filename = argv[2];

```

```

FILE *file = fopen(filename, "r");
if (file == NULL) {
    printf("Error opening file: %s\n", filename);
    return 1;
}

```

```

char line[MAX_LINE_LENGTH];
while (fgets(line, MAX_LINE_LENGTH, file) != NULL) {

```

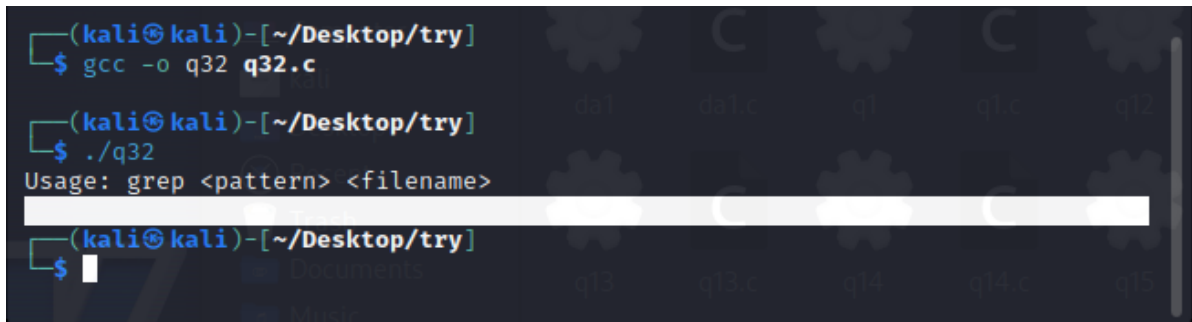
```

        if (strstr(line, pattern) != NULL) {
            printf("%s", line);
        }
    }

    fclose(file);
    return 0;
}

```

Output:



```

(kali@kali)-[~/Desktop/try]
$ gcc -o q32 q32.c

(kali@kali)-[~/Desktop/try]
$ ./q32
Usage: grep <pattern> <filename>

(kali@kali)-[~/Desktop/try]
$

```

4. Create a file with few lines, write a C program to read the file and delete the spaces more than one in the file.

Source Code:

```

#include <stdio.h>
#include <stdlib.h>

void removeExtraSpaces(char* filename) {
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        printf("Error opening the file.\n");
        return;
    }

    char tempFilename[] = "temp.txt";
    FILE* tempFile = fopen(tempFilename, "w");
    if (tempFile == NULL) {
        printf("Error creating temporary file.\n");
        fclose(file);
        return;
    }

    int prevSpace = 0; // flag to track previous space
    int currentChar;
    while ((currentChar = fgetc(file)) != EOF) {

```

```

    if (currentChar == ' ') {
        if (!prevSpace) {
            fputc(currentChar, tempFile); // write the first space
        }
        prevSpace = 1;
    } else {
        fputc(currentChar, tempFile); // write non-space characters
        prevSpace = 0;
    }
}

fclose(file);
fclose(tempFile);

if (remove(filename) != 0) {
    printf("Error deleting the original file.\n");
    return;
}

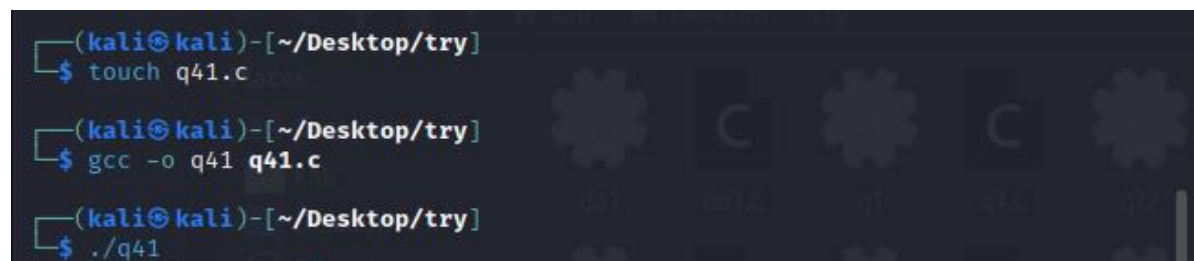
if (rename(tempFilename, filename) != 0) {
    printf("Error renaming the temporary file.\n");
    return;
}

printf("Spaces removed successfully.\n");
}

int main() {
    char filename[] = "example.txt";
    removeExtraSpaces(filename);
    return 0;
}

```

Output:



```

(kali㉿kali)-[~/Desktop/try]
$ touch q41.c

(kali㉿kali)-[~/Desktop/try]
$ gcc -o q41 q41.c

(kali㉿kali)-[~/Desktop/try]
$ ./q41

```

5. Write a program:

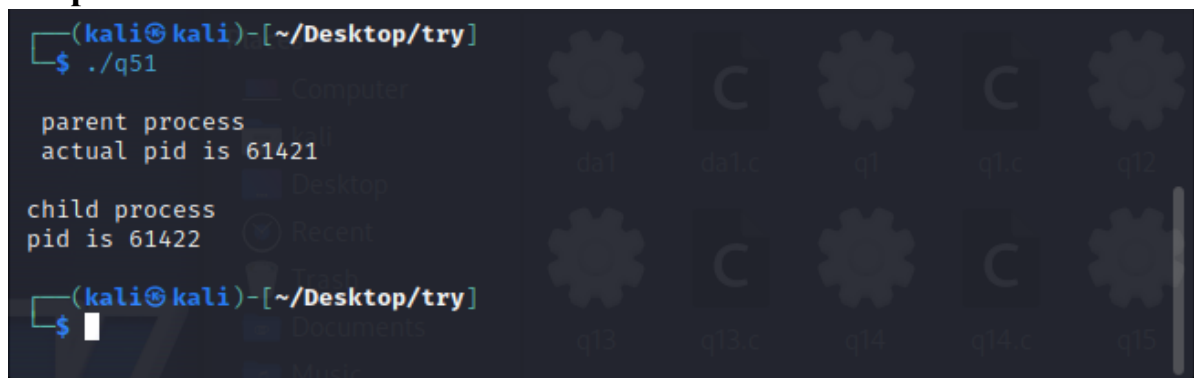
a. To create parent & child process and print their id.

Source Code:

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

int main()
{
    int pid;
    pid=fork();
    if(pid<0)
    {
        printf("\n Error ");
        exit(1);
    }
    else if(pid==0)
    {
        printf("\nchild process ");
        printf("\npid is %d ",getpid());
        exit(0);
    }
    else
    {
        printf("\n parent process ");
        printf("\n actual pid is %d \n ",getpid());
        exit(1);
    }
}
```

Output:



```
(kali㉿kali)-[~/Desktop/try]
$ ./q51

parent process
actual pid is 61421

child process
pid is 61422

(kali㉿kali)-[~/Desktop/try]
$
```

b. To create a zombie process.

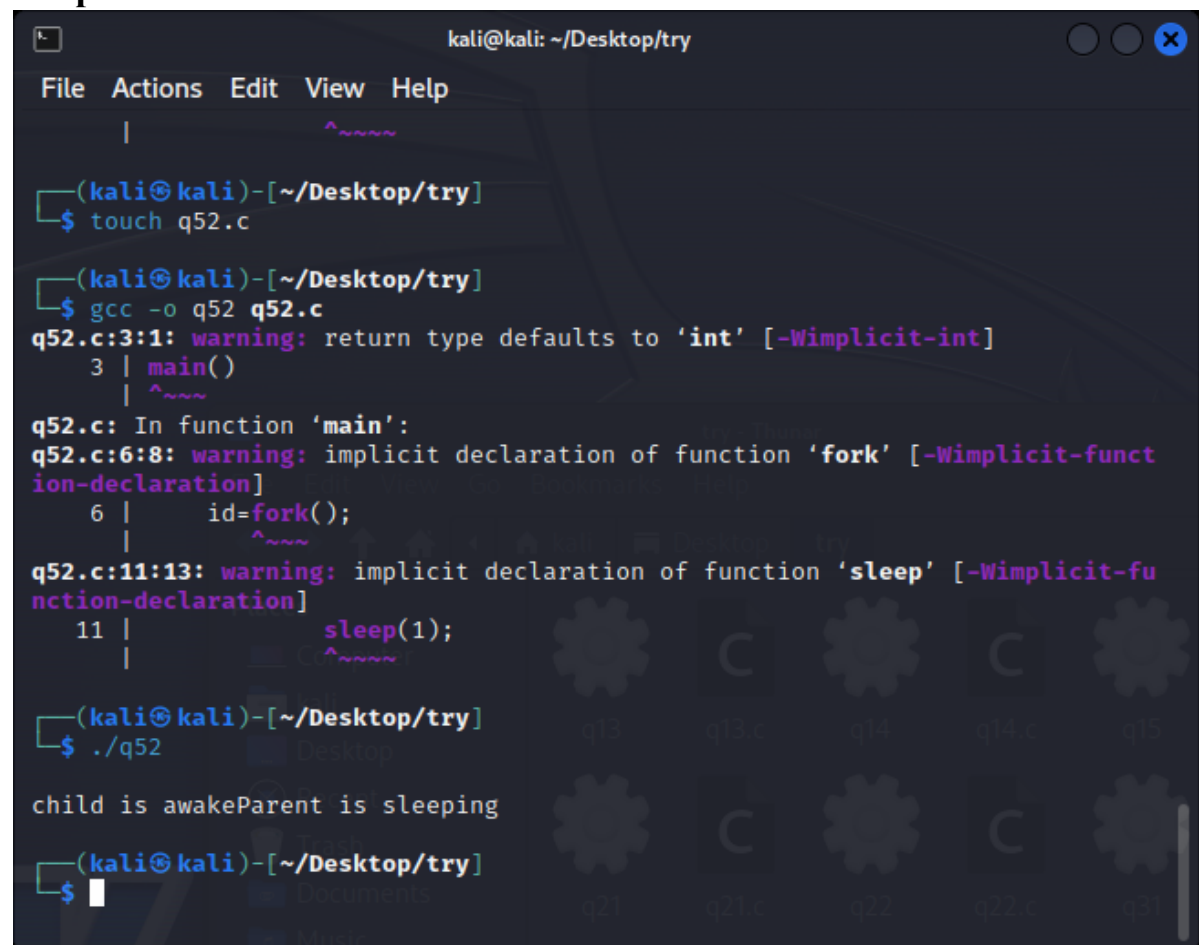
Source Code:

```
#include<stdio.h>
```

```
main()
{
    int id;
    id=fork();

    if(id>0)
    {
        printf("\nParent is sleeping");
        sleep(1);
    }
    if(id==0)
        printf("child is awake");
}
```

Output:



```
kali@kali: ~/Desktop/try
File Actions Edit View Help

(kali@kali)-[~/Desktop/try]
$ touch q52.c

(kali@kali)-[~/Desktop/try]
$ gcc -o q52 q52.c
q52.c:3:1: warning: return type defaults to 'int' [-Wimplicit-int]
  3 | main()
    | ^~~~~
q52.c: In function 'main':
q52.c:6:8: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
  6 |     id=fork();
    |         ^~~~~
q52.c:11:13: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
 11 |         sleep(1);
    |         ^~~~~

(kali@kali)-[~/Desktop/try]
$ ./q52
child is awakeParent is sleeping

(kali@kali)-[~/Desktop/try]
$
```

c. To create orphan process.

Source Code:

```
#include<stdio.h>
#include <sys/types.h>
#include <unistd.h>

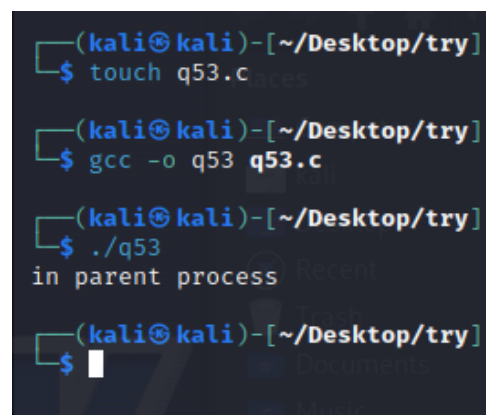
int main()
{
    // Create a child process
    int pid = fork();

    if (pid > 0)
        printf("in parent process");

    // Note that pid is 0 in child process
    // and negative if fork() fails
    else if (pid == 0)
    {
        sleep(30);
        printf("in child process");
    }

    return 0;
}
```

Output:



```
(kali㉿kali)-[~/Desktop/try]
$ touch q53.c

(kali㉿kali)-[~/Desktop/try]
$ gcc -o q53 q53.c

(kali㉿kali)-[~/Desktop/try]
$ ./q53
in parent process

(kali㉿kali)-[~/Desktop/try]
$
```

d. To make the process to sleep for few seconds.

Source Code:

```
#include<stdio.h>
```

```

#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>

int main(void){
    pid_t f;

    f = fork();

    if(f < 0){
        printf("Failed to fork\n");
        _exit(1);
    }

    else if(f == 0){
        int i;

        printf("\nChild: PID is %d\n", getpid());
        for(i = 0; i < 10; i++){
            printf("c ");
            if(i == 5)
                sleep(2);
        }
        printf("\n");

        _exit(0);
    }

    else{
        int j;

        printf("\nParent: PID is %d\n", getpid());
        for(j = 0; j < 10; j++){
            printf("p ");
        }
        printf("\n");
    }

    return 0;
}

```


Output:

```
(kali㉿kali)-[~/Desktop/try]
$ gcc -o q54 q54.c

(kali㉿kali)-[~/Desktop/try]
$ ./q54
Parent: PID is 64334
p p p p p p p p p
Child: PID is 64335
```

e.Implement the program to pass messages using pipes.

Source Code:

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

int main()
{
    int pid;
    pid=fork();
    if(pid<0)
    {
        printf("\n Error ");
        exit(1);
    }
    else if(pid==0)
    {
        printf("\nchild process ");
        printf("\npid is %d ",getpid());
        exit(0);
    }
    else
    {
        printf("\n parent process ");
        printf("\n actual pid is %d \n ",getpid());
        exit(1);
    }
}
```

Output:

```
kali@kali: ~/Desktop/try
File Actions Edit View Help

Parent: PID is 64334
p p p p p p p p p p

Child: PID is 64335

(kali@kali)-[~/Desktop/try]
$ c c c c c c c c c c

(kali@kali)-[~/Desktop/try]
$ touch q55.c

(kali@kali)-[~/Desktop/try]
$ gcc -o q55 q55.c

(kali@kali)-[~/Desktop/try]
$ ./q55

parent process
child process
actual pid is 65125
pid is 65126

(kali@kali)-[~/Desktop/try]
$
```