1). To create 'n' children. When the children will terminate, display total cumulative time children spent in user and kernel mode.

Ans:-

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/time.h>
#include<sys/resource.h>
#include<sys/wait.h>
#include<unistd.h>
int main(int argc , char **argv){
int n = atoi(argv[1]);
int i,status;
pid_t pid;
struct rusage r_usage;
struct timeval user_time,kernel_time;
long total_user_usec=0, total_kernal_usec=0;

for(i<0; i<n; i++){
pid = fork();
if(pid < 0){
perror("fork error");
exit(1);
}
else if(pid==0){
printf("child %d started \n",i+1);
sleep(5);
printf("child %d finished \n",i+1);
```

```
exit(0);
}
}

while((pid = wait(&status))>0){
if(getrusage(RUSAGE_CHILDREN,& r_usage) < 0){
perror("getrusage error");
exit(1);
}
user_time = r_usage.ru_utime;
kernel_time = r_usage.ru_stime;

printf("child %d: user time =%ld microseconds,kernel time = %ld
microseconds.\n",pid,user_time.tv_usec,kernel_time.tv_usec);
total_user_usec += user_time.tv_usec;
total_kernal_usec += kernel_time.tv_usec;
}

printf("Total time spend :%ld \n",total_user_usec);

printf("Total time spend :%ld \n",total_kernal_usec);

return 0;
}
```

Output:-

student@SCMIRT-32: ~/Desktop

File Edit View Search Terminal Help

```
compilation terminated.
student@SCMIRT-32:~/Desktop$ gcc program11.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc program11.c -o Myexe
student@SCMIRT-32:~/Desktop$ Myexe
Myexe: command not found
student@SCMIRT-32:~/Desktop$ Myexe
Myexe: command not found
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ .a/out
bash: .a/out: No such file or directory
student@SCMIRT-32:~/Desktop$ a/out
bash: a/out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ ./a.out
Failed to open file
student@SCMIRT-32:~/Desktop$ gcc program11.c
student@SCMIRT-32:~/Desktop$ ./a.out
Segmentation fault (core dumped)
student@SCMIRT-32:~/Desktop$
```

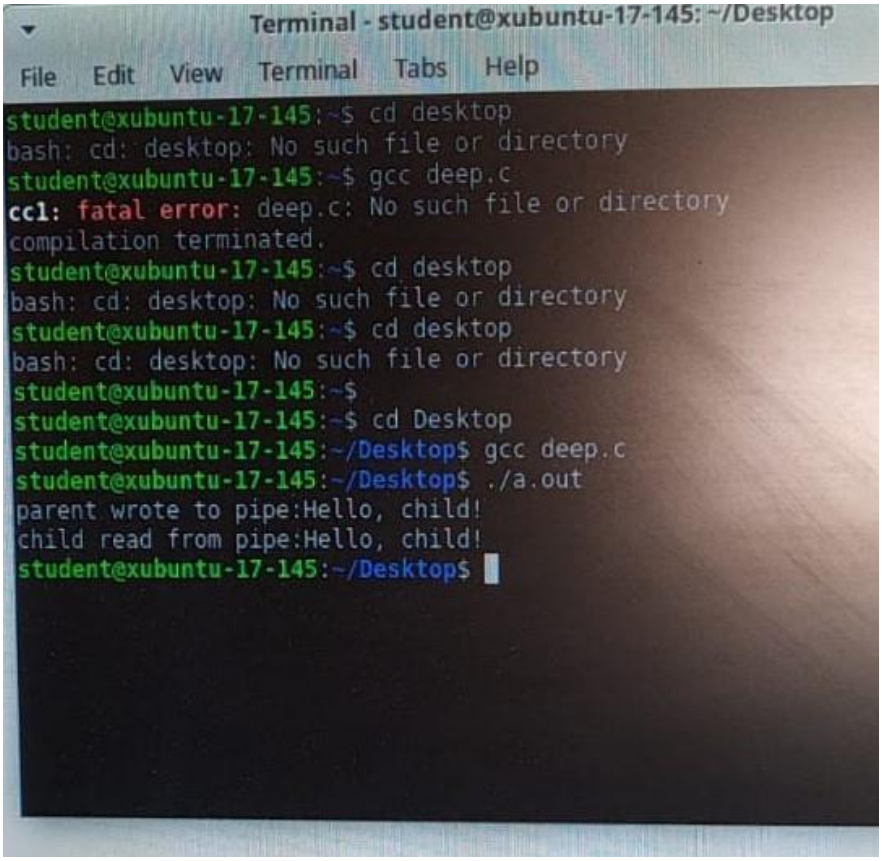
2). To generate parent process to write unnamed pipe and will read from it.

Ans:-

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

#define BUFFER_SIZE 25



int main() {
   int fd[2];
   pid_t pid;
   char write_msg[BUFFER_SIZE]="Hello, child!";
      char read_msg[BUFFER_SIZE];


     if(pipe(fd)<0) {
        perror("pipe error");
         exit(1);
    }

      pid=fork();
       if(pid<0){
         perror("fork error");
           exit(1);

 }else if (pid==0) {
    close(fd[1]);

    if(read(fd[0], read_msg,BUFFER_SIZE)<0){
           perror("read error");
          exit(1);
}
    printf("child read from pipe:%s\n", read_msg);


     close(fd[0]);
      exit(0);
    }else {
      close(fd[0]);

       if(write(fd[1],write_msg, strlen(write_msg)+1)<0){
           perror("write error");
            exit(1);
}
    printf("parent wrote to pipe:%s\n",write_msg);


    close(fd[1]);

     exit(0);
```

```
}
  return 0;
}
```

Output:-

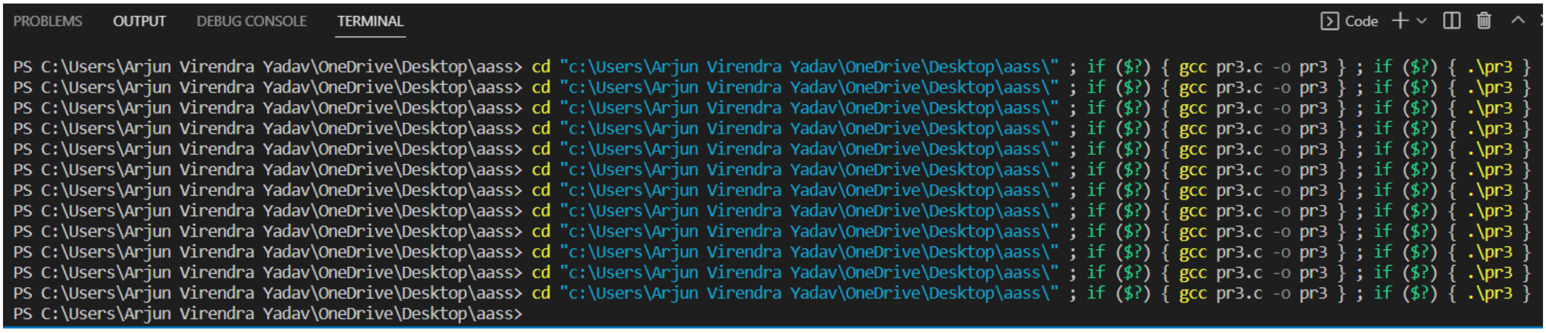


3). To create a file with hole in it.

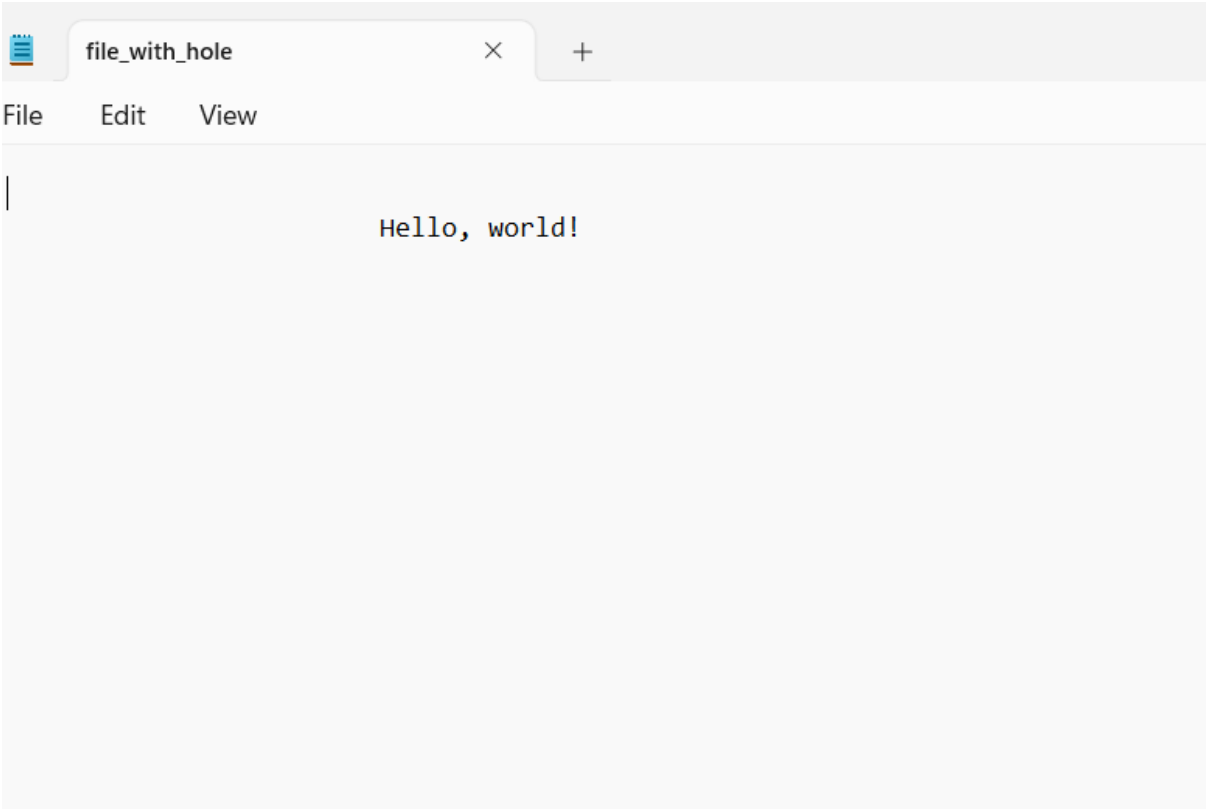
Ans:-

```
    #include <stdio.h>

#include <fcntl.h>
#include <unistd.h>
int main() {
 int fd;
 char data[] = "Hello, world!\n";
 off_t offset = 1024; // move file pointer 1024 bytes from beginning of file
 // open file for writing
 fd = open("file_with_hole", O_WRONLY | O_CREAT, 0666);
 if (fd < 0) {
 perror("open");
 return 1;
 }
 // move file pointer to offset and write data
 if (lseek(fd, offset, SEEK_SET) == -1) {
 perror("lseek");
 return 1;
 }
 if (write(fd, data, sizeof(data)) != sizeof(data)) {
 perror("write");
 return 1;
 }
 // close file
 if (close(fd) < 0) {
 perror("close");
 return 1;
 }
 return 0;
}
```

Output:-



```
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass> cd "c:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass\" ; if ($?) { gcc pr3.c -o pr3 } ; if ($?) { .\pr3 }
PS C:\Users\Arjun Virendra Yadav\OneDrive\Desktop\aass>
```

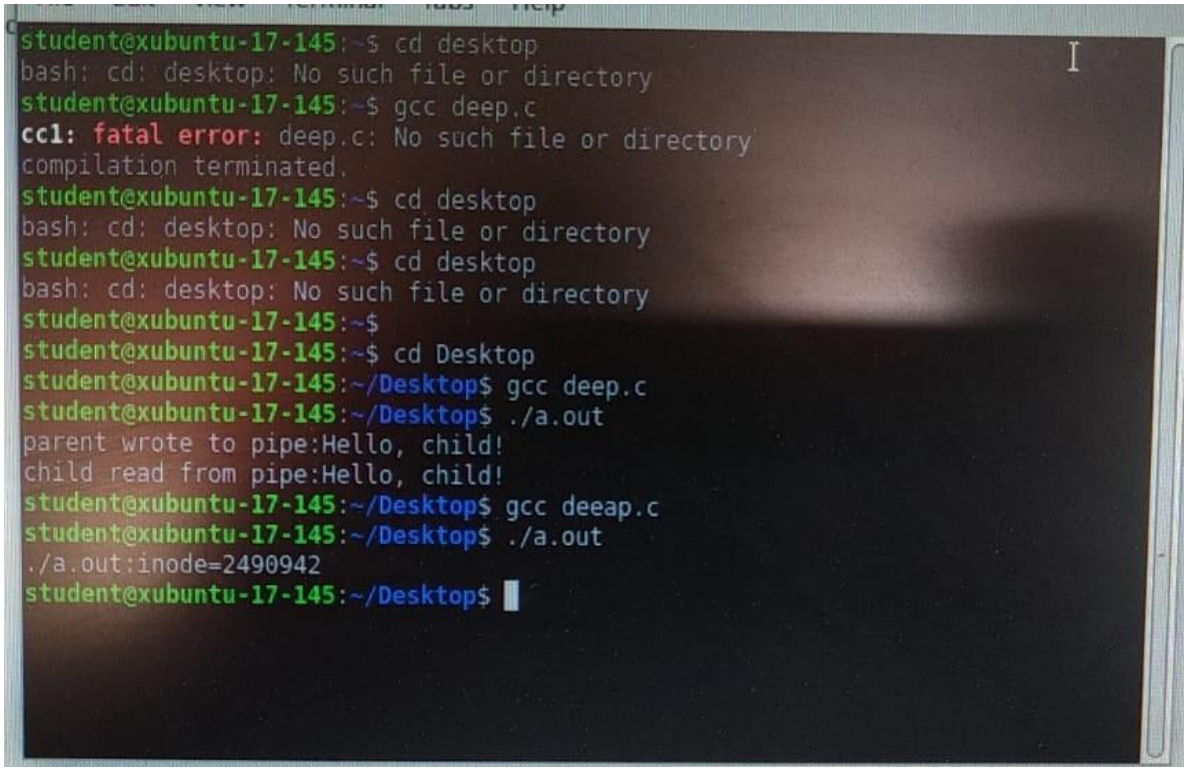

file_with_hole
File
Edit
View
Hello, world!

4). Takes multiple files as Command Line Arguments and print their inode number.

Ans:-

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

 int main(int argc, char*argv[]){
     struct stat buf;
     int i;

  // loop through coom
     for(i=0;i<argc;i++){
      // get file
       if(stat(argv[i],&buf)<0){
          perror("stat error");
            continue;
        }
              //print node
             printf("%s:inode=%ld\n",argv[i],buf.st_ino);}

           return 0;
    }
```

Output:-



5). To handle the two-way communication between parent and child using pipe.

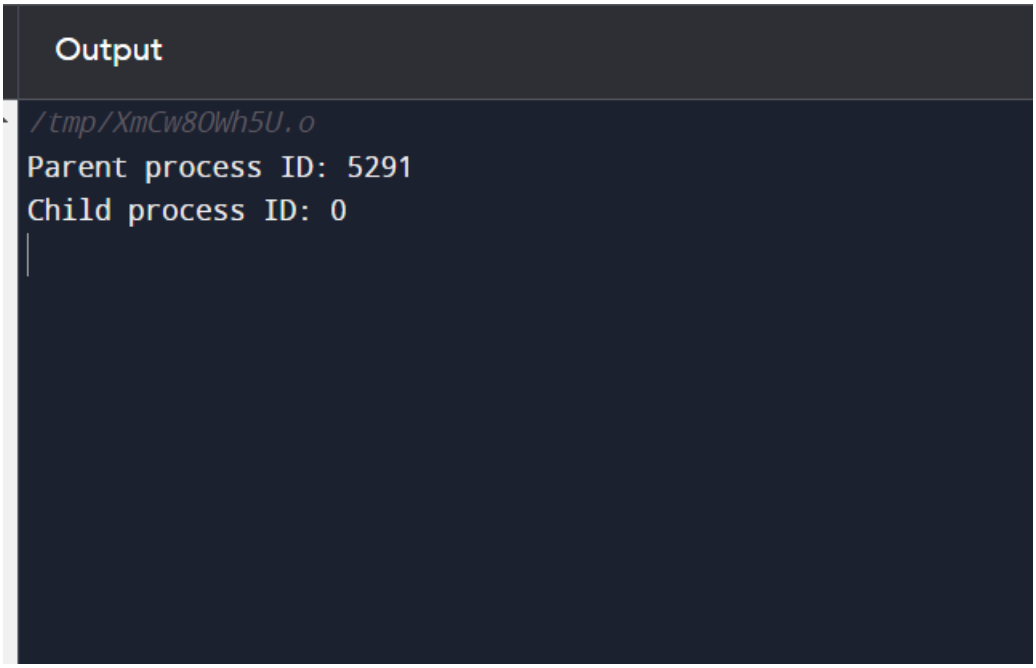
Ans:-

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

int main(int argc, char *argv[]){

    int pid;
    int p[2];   /* pipe "p" */
    int q[2];   /* pipe "q" */
    int a;
    int b;

    /* Create Pipe. Pipe P is used to transfer information
    from the parent process to the child process */
    a = pipe(p);
    if(a == -1)
    {
    fprintf(stderr, "Pipe Failed.\n");
    return EXIT_FAILURE;
    }

    /* Create second pipe. Pipe Q is used to transfer information
    from the child process to the parent process. */
```

```
b = pipe(q);
if(b == -1)
{
 fprintf(stderr, "Pipe Failed.\n");
 return EXIT_FAILURE;
}

/* Create child process */
pid = fork();

switch(pid){

        case -1: /* fork failed */
                 perror("main: fork");
                 exit(1);

                /* Child process will execute a loop, waiting for command
                from the parent process. Child executes the command. Child
                returns a response to the parent */
                case 0: /* Child process */
        printf("Child process ID: %d\n", pid);
        break;
        /* do some things */

        /* Parent process will execute a loop, asking user for a one
                line command. Parent sends command to child for execution.
```

```
            Parent waits for the response of the child. Parent finally
            reports the result (displayed on screen). */
            default: /* Parent process */
        printf("Parent process ID: %d\n", pid);
        break;
        /* do some things */
    }
    getchar();
    return 0;
}
```

Output:-



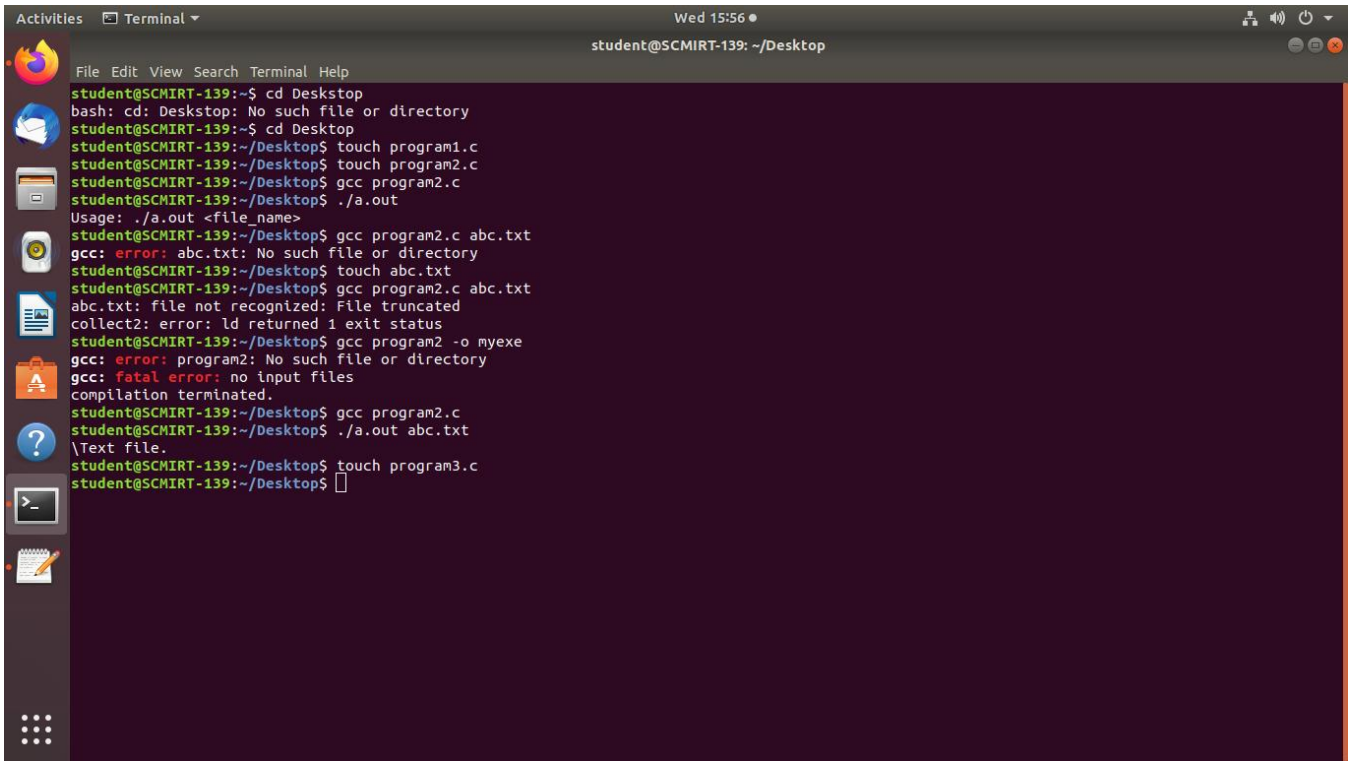
6). Print the type of file where file name accepted through Command Line.

Ans:-

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

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

    char *file_name = argv[1];

    char *extension = strrchr(file_name, '.');
    if (extension == NULL) {
        printf("File type cannot be determined.\n");
        return 1;
    }

    if (strcmp(extension, ".txt") == 0) {
        printf("Text file.\n");
    } else if (strcmp(extension, ".doc") == 0 || strcmp(extension, ".docx") == 0) {
        printf("Microsoft Word document.\n");
    } else if (strcmp(extension, ".pdf") == 0) {
        printf("PDF document.\n");
```

```
    } else {
        printf("File type not recognized.\n");
    }

    return 0;
}
```

Output:-



7). To demonstrate the use of atexit() function.

Ans:-

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

void cleanup1() {
   printf("Cleanup 1\n");
}

void cleanup2() {
   printf("Cleanup 2\n");
}

int main() {
   atexit(cleanup1);
   atexit(cleanup2);

   printf("Hello, world!\n");

   return 0;
}
```

Output:-

```
student@SCMIRT-32: ~/Desktop
File  Edit  View  Search  Terminal  Help
student@SCMIRT-32:~$ cd Desktop
student@SCMIRT-32:~/Desktop$ gcc deepak.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak7.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ ./a.out
File opened successfully
student@SCMIRT-32:~/Desktop$ gcc deepak7.c
student@SCMIRT-32:~/Desktop$ ./a.out
Hello, world!
Cleanup 2
Cleanup 1
student@SCMIRT-32:~/Desktop$
```

8). Open a file goes to sleep for 15 seconds before terminating.

Ans:-

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

int main() {
    FILE *fp;

    fp = fopen("filename.txt", "r");
    if (fp == NULL) {
        printf("Failed to open file\n");
        return 1;
    }

    printf("File opened successfully\n");

    sleep(15);

    fclose(fp);

    return 0;
}
```

Output:-

```
student@SCMIRT-32: ~/Desktop
File  Edit  View  Search  Terminal  Help
student@SCMIRT-32:~$ cd Desktop
student@SCMIRT-32:~/Desktop$ gcc deepak.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak7.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ .a/.out
bash: .a/.out: No such file or directory
student@SCMIRT-32:~/Desktop$ gcc deepak8.c
student@SCMIRT-32:~/Desktop$ ./a.out
File opened successfully
```

9). To print the size of the file

Ans:-

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

int main() {
    FILE *fp;
    long size;

    fp = fopen("filename.txt", "r");
    if (fp == NULL) {
        printf("Failed to open file\n");
        return 1;
    }

    fseek(fp, 0L, SEEK_END);    // Move the file pointer to the end of the file
    size = ftell(fp);           // Get the current position of the file pointer
    fclose(fp);

    printf("Size of file: %ld bytes\n", size);

    return 0;
}
```

Output:-

student@SCMIRT-32: ~/Desktop

File Edit View Search Terminal Help

```
student@SCMIRT-32:~$ cd Desktop
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ ./a.out
Failed to open file
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ ./a.out
Failed to open file
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ ./a.out
Failed to open file
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ ./a.out
Failed to open file
student@SCMIRT-32:~/Desktop$ gcc dipp.c
student@SCMIRT-32:~/Desktop$ ./a.out
Size of file: 8520 bytes
student@SCMIRT-32:~/Desktop$
```