**Index:**

1. **Program to illustrate File Management System calls.**
2. **Exercise regarding File Management System calls(read contents from file1 and perform square root for the respected integers, write it into file2)**

**Note:** Outputs: Left side

##### **open,read , write,close System Calls**

We all know what is a file , it is nothing but a collection of similar records and here each and every file is associated with its file descriptors(fd) and using that file descriptor we can read or write in to that file and initially to get the fd of a file ,we have to make call to open() by sending path of the file as an argument (either absolute or relative path) and then we can read or write in to that file using the fd of that file.

* File descriptor 0 is for standard input (key board) , if we mention fd in read() function as 0 means to read from key board itself.
* File descriptor 1 is for standard output (monitor/screen) , if we mention fd in write() function as 1 means to write on the screen (monitor) itself.
* File descriptor 2 for standard error

**int open(const char\* path, int oflags,mode\_t mode);**

**purpose**: Either to open already existing file or to create a new file with the specified path as in 1st argument

**Arguments**:

1st one either we have to specify absolute path or relative path

 2nd one is for setting permissions on the file we are creating. Generally we wont bother about this one and we will include this argument while we creating the file but not while opening.

3rd one we have to specify the mode of opening of file

**O\_RDONLY =>**TO OPEN FILE IN READ MODE

**O\_WRONLY =>**TO OPEN FILE IN WRITE MODE

**O\_RDWR =>** TO OPEN IN BOTH READ AND WRITE MODE

**O\_CREAT =>**TO CREATE FILE NEWLY

**O\_APPEND =>**TO OPEN FILE IN APPEND

we can use the above flags in combined form also using | operator just like

O\_RDWR|O\_CREAT

**size\_t read(int fd, void\* buf, size\_t data);**

fd => file descriptor of file from which you want to read

buf => expects the pointer to buffer where it has to read the data from file

data => size of the data

**size\_t write(int fd, void\* buf, size\_t data);**

fd => file descriptor of file in which you want to write

buf=> expects the pointer to buffer which has to be written into file.

data=>size of the data

##### **Experiment # 1(a): Use Of Creat And Open System Calls**

##### **AIM:** To write a program in C to Illustrate the use of create and open system calls using command line arguments.

**PROGRAM :**

#include<stdio.h>

#include<fcntl.h>

#include<stdlib.h>

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

{

int fid1,fid2,i;

char buff[500];

if(argc!=3)

{

printf("error");

exit(1);

}

fid1=open(argv[1],O\_RDONLY);

i=read(fid1,buff,500);

creat(argv[2],S\_IRWXU);

fid2=open(argv[2],O\_RDWR);

write(fid2,buff,i);

close(fid1);

close(fid2);

}

**Result:**

**Experiment # 1(b): Illustrate the use of creat and open system calls**

##### **AIM:**To write a program in C to Illustrate the use of open and create system calls without using command line arguments

**PROGRAM:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

main()

{

int fid1,fid2,i;

char buff[500];

fid1=open("test",O\_RDONLY);

i=read(fid1,buff,100);

creat("sample",S\_IRWXU);

fid2=open("sample",O\_RDWR);

write(fid2,buff,i);

close(fid1);

close(fid2);

}

Result:

**Viva Questions**

**1.What is about open() system call**

open system call is used to open a file, except that instead of returning a file pointer, it returns a file descriptor, which is just an int. open returns -1 if any error occurs.

**2. Syntax of creat() system call**

**int creat(const char \*pathname,int accessp)**

**3.What is file descriptor**

**file descriptor** is an abstract indicator for accessing a file. The term is generally used in [POSIX](http://en.wikipedia.org/wiki/POSIX)[operating systems](http://en.wikipedia.org/wiki/Operating_system). A file descriptor is an [index](http://en.wikipedia.org/wiki/Index_(information_technology)) for an entry in a [kernel](http://en.wikipedia.org/wiki/Kernel_(computer_science))-resident data structure containing the details of all open files.