FIFO file:

It is a special pipe device file which provides a temporary buffer for two or more processes to communicate by writing data to and reading data from the buffer. The size of the buffer is fixed to PIPE\_BUF. Data in the buffer is accessed in a first-in-first-out manner. The buffer is allocated when the first process opens the FIFO file for read or write. The buffer is discarded when all processes close their references (stream pointers) to the FIFO file. Data stored in a FIFO buffer is temporary.

The prototype of mkfifo is:

**int mkfifo(const char \*path\_name, mode\_t mode);**

The first argument pathname is the pathname(filename) of a FIFO file to be created. The second argument mode specifies the access permission for user, group and others and as well as the S\_IFIFO flag to indicate that it is a FIFO file. On success it returns 0 and on failure it returns –1.

**open:**

This is used to establish a connection between a process and a file i.e. it is used to open an existing file for data transfer function or else it may be also be used to create a new file. The returned value of the open system call is the file descriptor (row number of the file table), which contains the inode information.

The prototype of open function is,

**int open(const char \*pathname, int accessmode, mode\_t permission);**

♣ If successful, open returns a nonnegative integer representing the open file descriptor.

♣ If unsuccessful, open returns –1.

♣ The first argument is the name of the file to be created or opened. This may be an absolute pathname or relative pathname.

♣ If the given pathname is symbolic link, the open function will resolve the symbolic link reference to a non symbolic link file to which it refers.

The second argument is access modes, which is an integer value that specifies how actually the file should be accessed by the calling process.

♣ Generally the access modes are specified in . Various access modes are:

O\_RDONLY - open for reading file only

O\_WRONLY - open for writing file only

O\_RDWR - opens for reading and writing file.

**size\_t read(int fdesc, void \*buf, size\_t nbyte);**

♣ If successful, read returns the number of bytes actually read.

♣ If unsuccessful, read returns –1.

♣ The first argument is an integer, fdesc that refers to an opened file.

♣ The second argument, buf is the address of a buffer holding any data read.

♣ The third argument specifies how many bytes of data are to be read from the file.

**ssize\_t write(int fdesc, const void \*buf, size\_t size);**

♣ If successful, write returns the number of bytes actually written.

♣ If unsuccessful, write returns –1.

♣ The first argument, fdesc is an integer that refers to an opened file.

♣ The second argument, buf is the address of a buffer that contains data to be written.

♣ The third argument, size specifies how many bytes of data are in the buf argument.

♣ The return value is usually equal to the number of bytes of data successfully written to a file. (size value)