A file descriptor is an unsigned integer used by a process to identify an open file. The open() system call returns a pointer to the appropriate entry in the per process open file table, and all file operations are performed by this pointer. The value returned by the open() call is called the file descriptor, which indexes into a system wide table of files opened by all processes, called the file table, which, in turn, is maintained by the kernel. This table also records the mode in which the file has been opened: reading, writing, appending and other modes.

READING A FILE USING TWO PROCESSES ( open() and fork()) :

A process P1 opens a file in read mode, and then forks to create a child process P2.

The following observations are made:

* The value of the file descriptors for both P1 and P2 are same, indicating that the child process inherits the file descriptor from the parent. The file descriptors for both processes being same, they point to the same location in the file table.
* The file is usually read first by the parent process (P1), then the child process (P2).
* The child process starts reading the file from the position where the parent process has stopped.
* Even if the parent process closes the file, the child process reads the file as before, i.e. it starts reading from the position where P1 stopped.
* If a wait() command is given to P1, the child process may start reading the file first.

However, if two open() system calls are made for the parent and child processes, the two file descriptors create two different mappings in the file table. As a result, both P1 and P2 start reading from the beginning of the file, but the data gets printed haphazardly.

WRITING A FILE USING TWO PROCESSES:

Here, P1 opens a file in write mode, and then forks to create a child process P2.

Observations:

* The parent process writes in the file before the child process.
* The strings written by the parent and child processes are appended, and not overwritten.
* If the text file is already created, the open() system call sets the file descriptor to the beginning of the file, and the write mode results in the file getting overwritten.
* If the write() call is made outside an if-else statement, the same string gets printed twice, once by the parent process, and once by the child.