- In computer programming and operating systems, a file descriptor is a unique integer value that identifies an open file or I/O stream.
- It serves as an abstraction to interact with files, sockets, pipes, and other input/output resources in a uniform way.

When a process opens a file or creates a new one, the operating system assigns a file descriptor to that file, allowing the process to read from or write to the file. The file descriptor acts as a reference or handle to access the file's data and other attributes.

Key points about file descriptors:

1. Integer Representation: File descriptors are represented as non-negative integers. In most systems, 0, 1, and 2 are reserved for standard input (stdin), standard output (stdout), and standard error (stderr), respectively.
2. Managing Open Files: The operating system maintains a table that maps file descriptors to the corresponding open files for each process. This table keeps track of file position, permissions, and other information related to the file.
3. Accessing Files: File descriptors are used in various system calls to interact with files. For example, the open(), read(), write(), close(), and other related functions take a file descriptor as an argument.
4. File Descriptor Duplication: Processes can duplicate file descriptors using the dup() or dup2() system calls. This feature allows a process to have multiple file descriptors pointing to the same open file, enabling concurrent access.
5. Standard I/O Redirection: File descriptors play a vital role in standard I/O redirection. For instance, the output of a program can be redirected to a file using the shell syntax (>) by changing the file descriptor associated with standard output.
6. Closing File Descriptors: It's essential for a process to close file descriptors properly

after using them to free up system resources. Not closing file descriptors can lead to resource leaks.