

OPERATING SYSTEMS



INPUT, OUTPUT AND ERROR

REDIRECTION IN UNIX/LINUX

- Linux redirection features can be used to detach the default files from stdin, stdout, and stderr and attach other files with them for a single execution of a command.
- The act of detaching default files from stdin, stdout, and stderr and attaching other files with them is known as input, output, and error redirection.



UNIX/LINUX FIFOs

- A named pipe (also called a named FIFO, or just FIFO) is a pipe whose access point is a file kept on the file system.
- By opening this file for reading, a process gets access to the FIFO for reading.
- By opening the file for writing, the process gets access to the FIFO for writing. By default, a FIFO is opened for blocking I/O.
- This means that a process reading from a FIFO blocks until another process writes some data in the FIFO. The same goes the other way around.



UNIX/LINUX FIFOs

- Unnamed pipes can only be used between processes that have an ancestral relationship.
- And they are temporary; they need to be created every time and are destroyed when the corresponding processes exit.
- Named pipes (FIFOs) overcome both of these limitations.



UNIX/LINUX FIFOs

Named pipes are created via:

- `mknod()` system call—(designed to create special device files)
- or `mkfifo()` C library call—(invokes `mknod` system call)
- or by the `mkfifo` command



UNIX/LINUX FIFOs

- Unlike a pipe, a FIFO must be opened before using it for communication.
- A write to a FIFO that no process has opened for reading results in a SIGPIPE signal.
- When the last process to write to a FIFO closes it, an EOF is sent to the reader.
- Multiple processes can write to a FIFO are atomic writes to prevent interleaving of multiple writes.



UNIX/LINUX FIFOs

Two common uses of FIFOs are:

- In client-server applications, FIFOs are used to pass data between a server process and client processes
- Used by shell commands to pass data from one shell pipeline to another, without creating temporary files



UNIX/LINUX FIFOs

- Ordinary pipes exist only while the processes are communicating with one another.
- On both UNIX and Windows systems, once the processes have finished communicating and have terminated, the ordinary pipe ceases to exist.
- Named pipes provide a much more powerful communication tool.
- Communication can be bidirectional, and no parent-child relationship is required.
- Once a named pipe is established, several processes can use it for communication.
- Although FIFOs allow bidirectional communication, only half-duplex transmission is permitted.
- If data must travel in both directions, two FIFOs are typically used. Additionally, the communicating processes must reside on the same machine.



UNIX/LINUX FIFOs FAILS

- File with the given name already exist.
- Pathname too long.
- A component in the pathname not searchable, non-existent or non-directory.
- Destination directory is read-only.
- Not enough memory space.
- Signal caught during mknod.



UNIX/LINUX FIFOs

```
int mknod(const char *pathname, mode_t mode, dev_t dev);
```

```
int mkfifo(const char *pathname, mode_t mode);
```

