Pipes and FIFOs

- Pipes and FIFOs are a much more general purpose Inter-Process Communications form than are signals.
- FIFOs are also called *named pipes*.
- Both are *byte-streams*. There are no message boundaries between data written to a pipe or FIFO.

Pipes

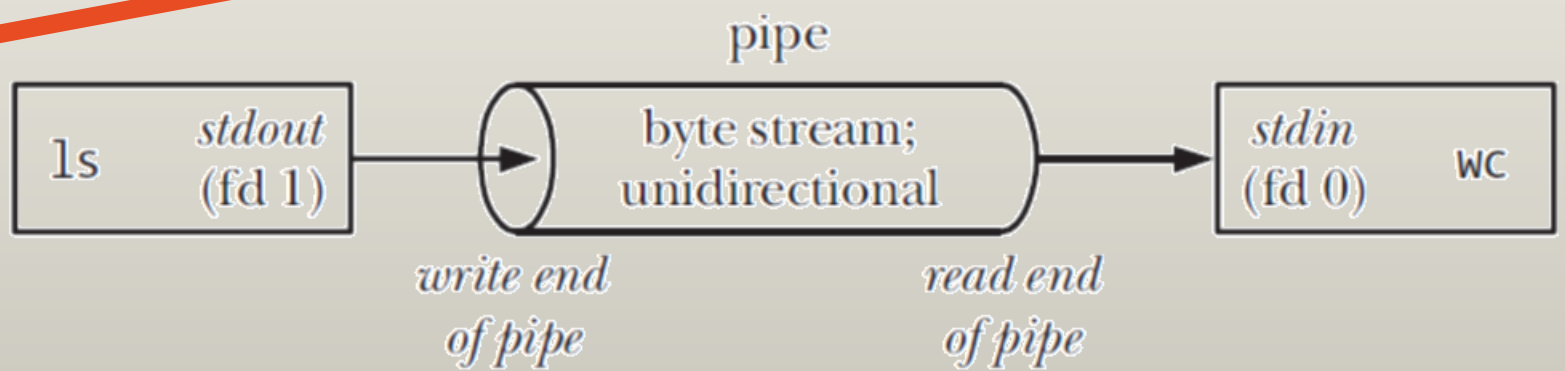
- Pipes are used between **related** processes. Sometimes called anonymous pipes.
- Processes using pipes to communicate will have a common ancestor process.
- Pipes are created by a parent process then child processes are created (using *fork*()) to make use of the pipes.

Pipes

You've used pipes on the command line between processes.

```
ls | wc -l
```

pipe



Pipes

- Pipes are **kernel** structures. They are buffers kept in the kernel.
- Pipes are sequentially read/writen only. You cannot seek (*lseek()*) forward or backward on a pipe.
- You can have multiple readers/writers, but it is difficult.
- Pipes are **unidirectional**. There is a read end on the pipe and a write end on a pipe.
- Pipes are anonymous (there is no entry in the file system for a pipe).
- Pipes have a limited capacity.

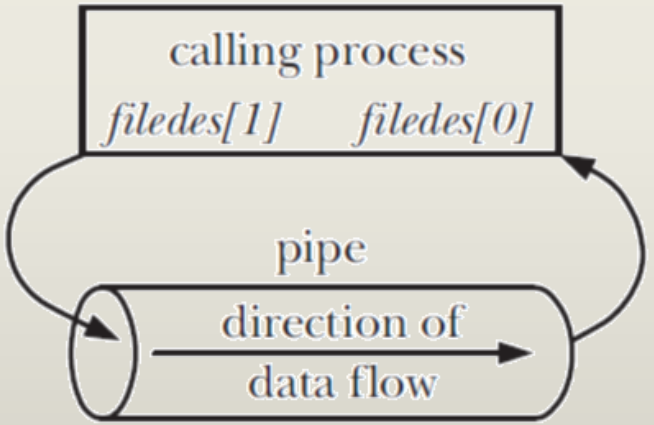
Creating Pipes

- Created using *pipe()*:

```
int filedes[2];
pipe(filedes);

...

write(filedes[1], buf, count);
read(filedes[0], buf, count);
```



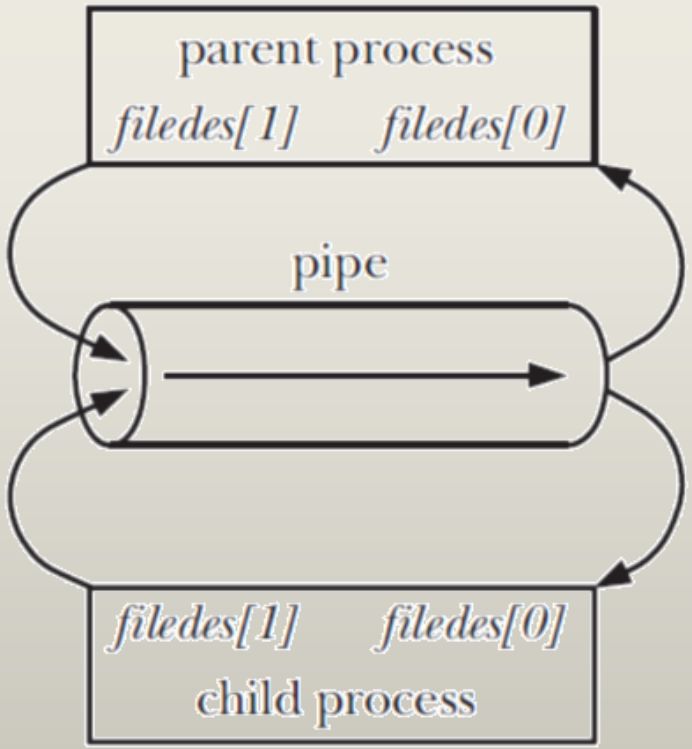
Sharing a Pipe

```
int filedes[2];

pipe(filedes);

child_pid = fork();
```

fork() duplicates parent's file descriptors

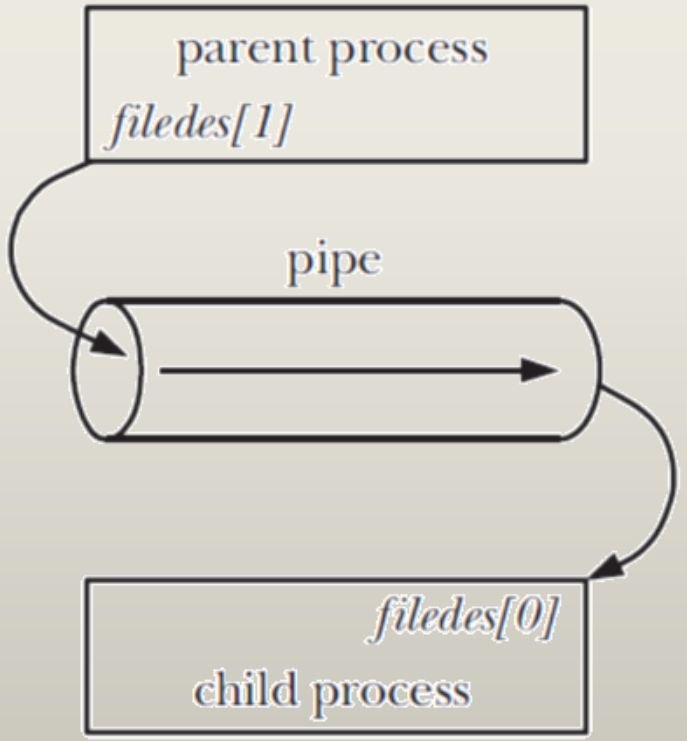


Sharing a Pipe

```
int filedes[2];

pipe(filedes);

child_pid = fork();
if (child_pid == 0) {
    close(filedes[1]);
    /* Child now reads */
} else {
    close(filedes[0]);
    /* Parent now writes */
}
```



Close Unused Sides of the Pipe

- Parent the child process **must** close the unused file descriptors.
  - This is necessary of the correct use of pipes.
- *close()* write end
  - *read()* returns 0 (EOF)
- *close()* read end
  - *write()* fails with EPIPE error and SIGPIPE signal

I/O on Pipes

- read() **blocks** if pipe is empty
- write() **blocks** if pipe is full
- Writes <= `PIPE_BUF` guaranteed to be atomic
  - Multiple writers > `PIPE_BUF` may be interleaved
  - POSIX: `PIPE_BUF` at least 512 Bytes
  - Linux: PIPE_BUF is 4096 Bytes
- Can use dup2() to connect filters via a pipe.
- You don't have to open pipes, only close unused side.

FIFOs
aka *Named Pipes*

- Anonymous pipes can **only** be used between related processes.
- A FIFO is a pipe with a **name** in the file system.
- Creation:
  - mkfifo(path, permissions);
- Any process can open and use a FIFO (based on permissions).
- I/O is the same as for pipes.
- Unlike pipes, you **do** have to open FIFOs before use.
- FIFOs are also kernel data structures.

Opening a FIFO

- open(path, O_RDONLY);
  - Open read end of a FIFO
- open(path, O_WRONLY);
  - Open the write side of a FIFO
- Calls to open() will **block** until the other end of the FIFO is opened.
  - Opens are synchronized.
  - You can also
  - open(path, O_RDONLY | O_NONBLOCK);