Interprocess Communication

PIPES

Material from:
https://pdfs.semanticscholar.org/presentation/dff0/af8a4ce3d0c898c08c39c1449373e4b1795e.pdf

Carlos Alberto Rincón Castro | carincon@uh.edu

Pipes

- **Create a pipe: system call pipe()**

```
#include <unistd.h>
int pipe(int fd[2]);
```

- **Create a unidirectional communication buffer with two file descriptors: fd[0] for read and fd[1] for write.**

- **Data write and read on a first-in-first-out base. No external or permanent name, and can only be accessed through two file descriptors.**

- **The pipe can only be used by the process that created it and its descendants (i.e., child & grand-child processes)**

Pipes

Unnamed Pipes

Pipes

close(fd): closes a file descriptor.

dup(newfd) and dup2(newfd, oldfd): Duplicate a file descriptor.

Dup and dup2 system calls create a copy of a given file descriptor. This new descriptor actually does not behave like a copy, but like an alias of the old one.

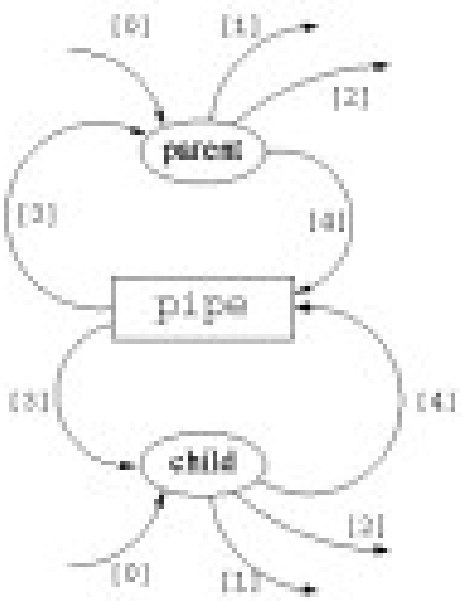
Read

Not necessarily atomic: may read less bytes.
Blocking: if no data, but write file descriptor still opens.
If empty, and all file descriptors for the write end are closed, read sees end-of-file and returns 0.

Write

Atomic for at most PIPE_BUF bytes (512, 4K, or 64K).
Blocking: if buffer is full, and read file descriptors open.
When all file descriptors referring to the read end of a pipe are closed cause a SIGPIPE signal for the calling process.

Pipes



parent

file descriptor table

[0]	*standard input*
[1]	*standard output*
[2]	*standard error*
[3]	pipe *read*
[4]	pipe *write*

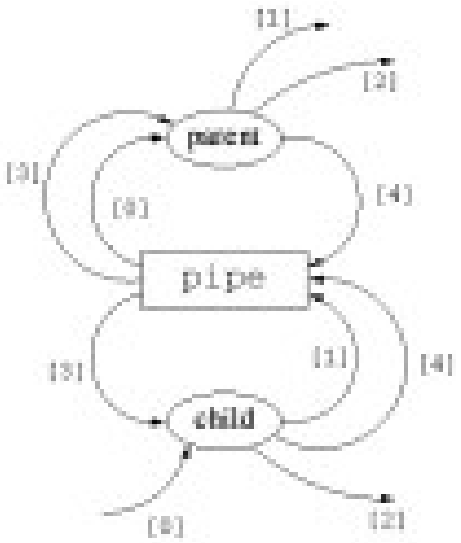
child

file descriptor table

[0]	*standard input*
[1]	*standard output*
[2]	*standard error*
[3]	pipe *read*
[4]	pipe *write*

After calling the pipe function

Pipes



parent

file descriptor table

[0]	pipe *read*
[1]	*standard output*
[2]	*standard error*
[3]	pipe *read*
[4]	pipe *write*

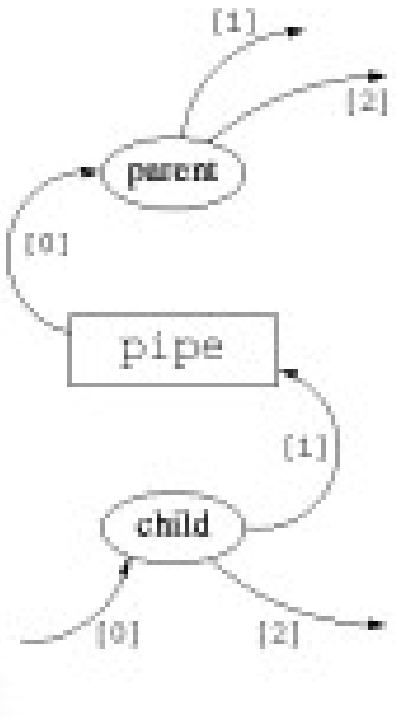
child

file descriptor table

[0]	*standard input*
[1]	pipe *write*
[2]	*standard error*
[3]	pipe *read*
[4]	pipe *write*

After calling the dup2 function

Pipes



parent

file descriptor table

[0]	pipe *read*
[1]	*standard output*
[2]	*standard error*

child

file descriptor table

[0]	*standard input*
[1]	pipe *write*
[2]	*standard error*

After calling the close function

Pipes

Pros:

- Simple
- Flexible
- Efficient communication

Cons:

No way to open an already existing pipe. This makes it impossible for two arbitrary processes to share the same pipe, unless the pipe was created by a common ancestor process.