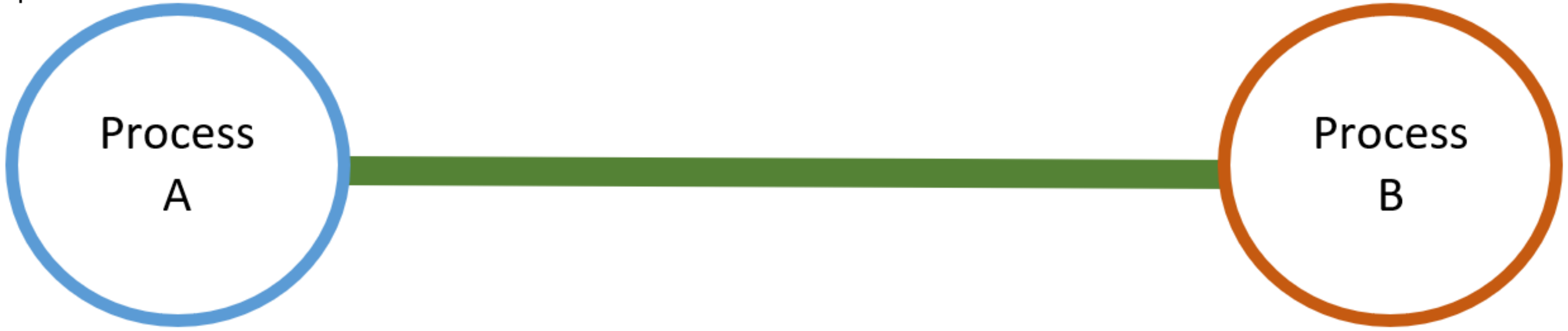
Network Programming

IPC with Pipes

Note: **This class lecture will be recorded!**
If you do not consent to this recording, please do not ask questions via your video, audio or public chat; send your question to the instructor using the private chat.

Lisa Frye, Instructor
frye@Kutztown.edu
Kutztown University

Pipes



Pipes

- Interprocess Communication

  - Characteristics
    - Half-duplex (data flows in one direction)
    - Common ancestor

  - Types
    - Unnamed
    - Named

Pipes in the Shell

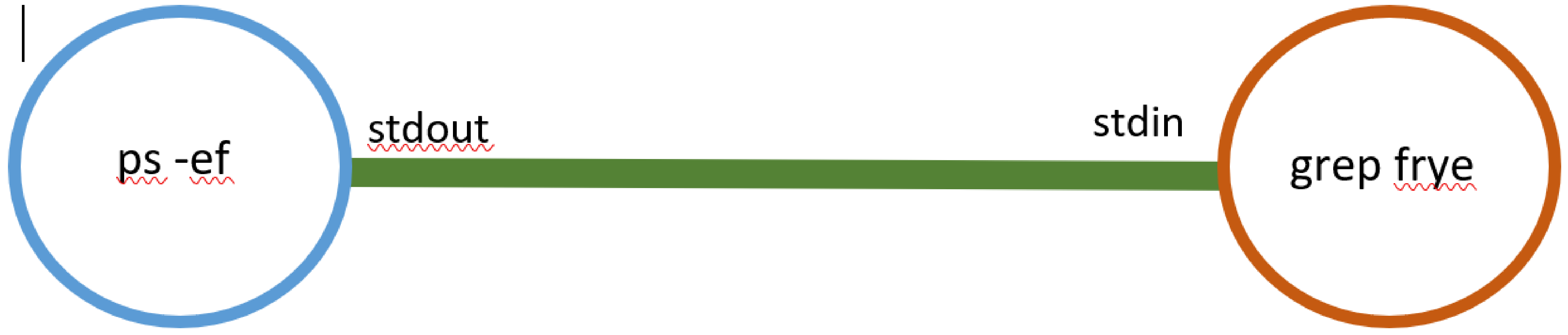
- ps -ef | grep frye

- Processes
  - Parent – fork
  - Child – exec

- Pipe commands
  - Need a pipe
  - Need a process (fork) for each command
  - Redirect standard out for first command to write end of pipe
  - Redirect standard in for second command to read end of pipe

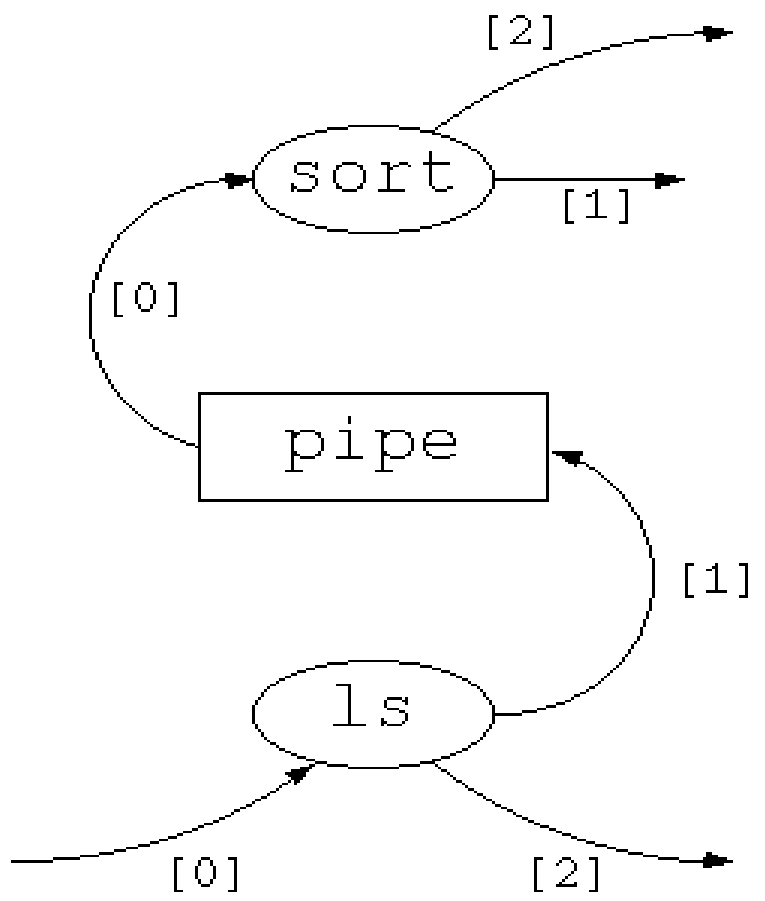
Shell Example Flow

- ps -ef | grep frye



- Draw a flowchart for this, including the system calls

Pipe Shell Example



sort

file descriptor table

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

ls

file descriptor table

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

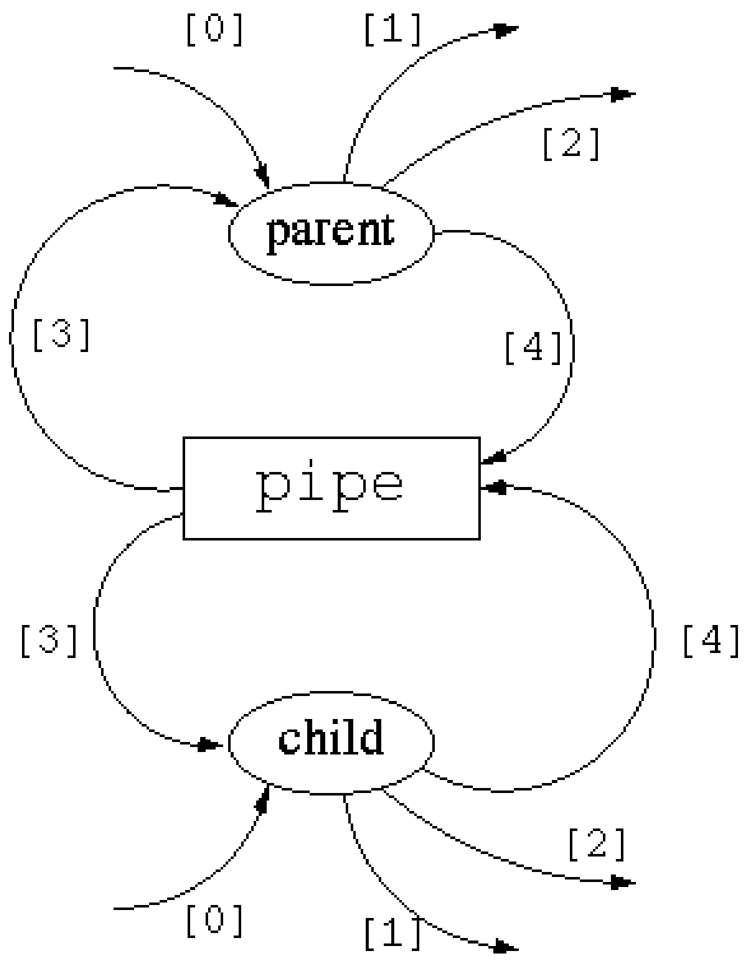
Pipe Creation

- pipe()
  - Two file descriptors
    - Read
    - Write
- File descriptors after fork()

- pipes/pipeEx.c

Pipe Example

- dup2() function call
- pipes/simpleredirect.c



parent

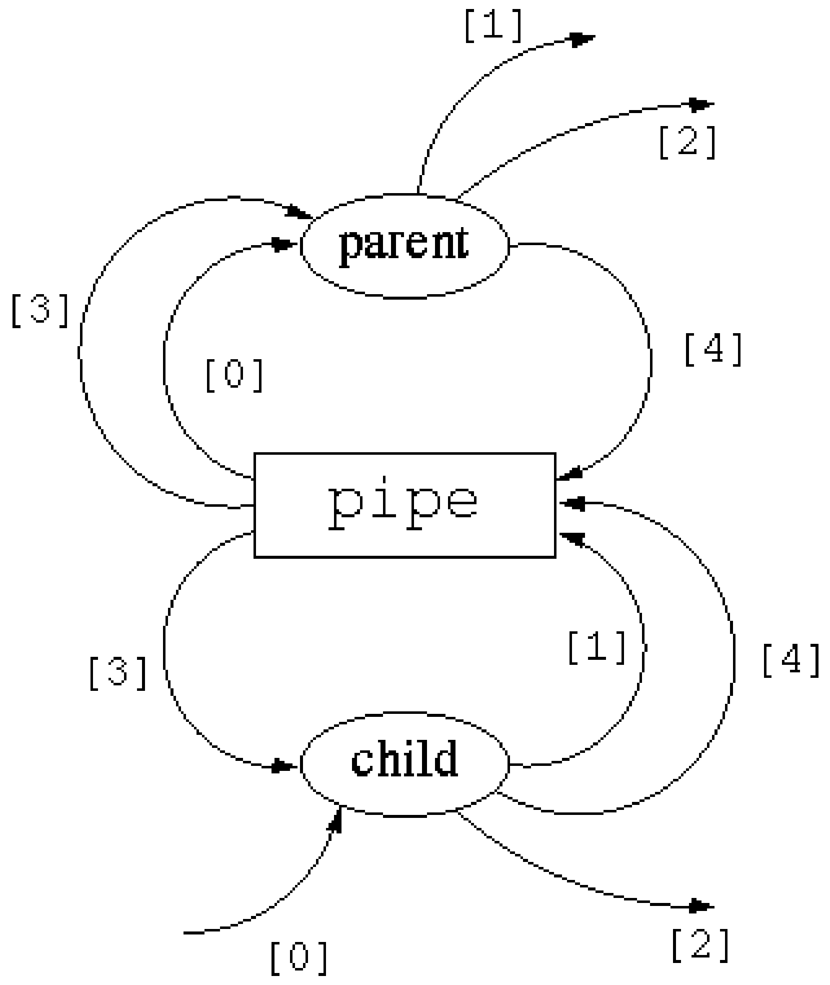
file descriptor table

[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*



parent

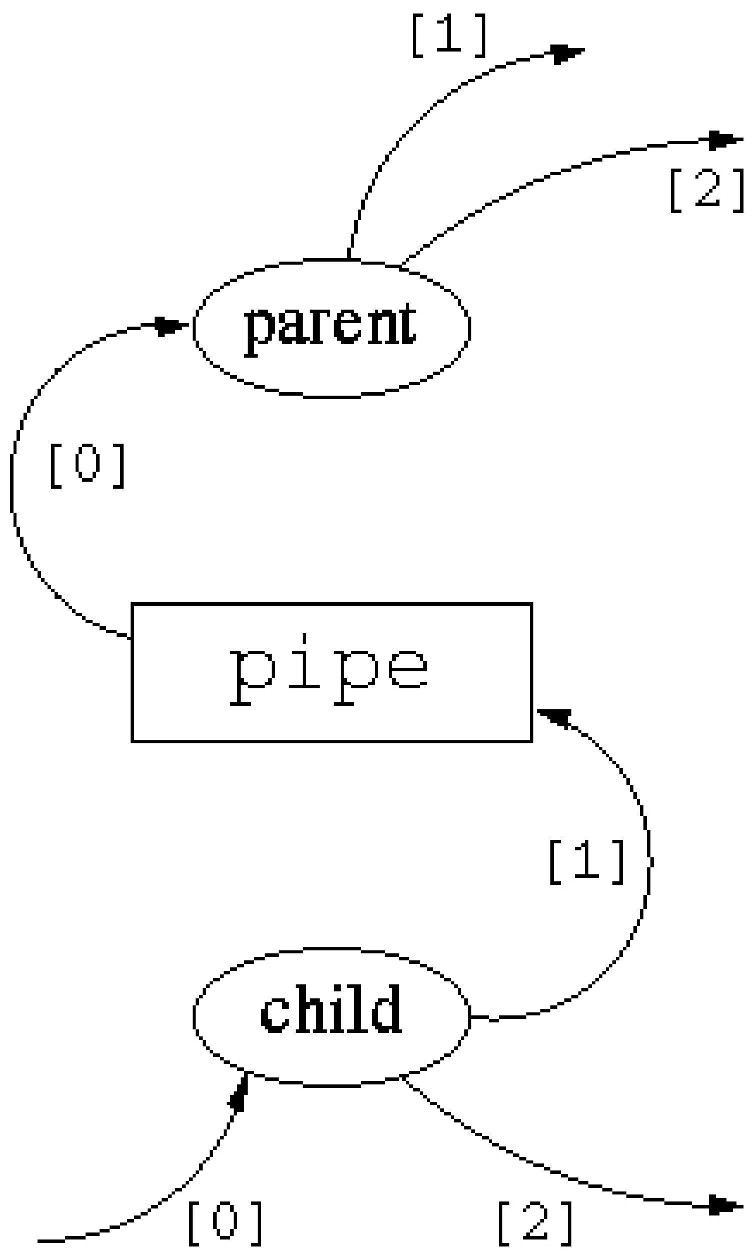
file descriptor table

[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*



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*

Dr

Pipe Usage

- read
- write

- Protocol for reading and writing

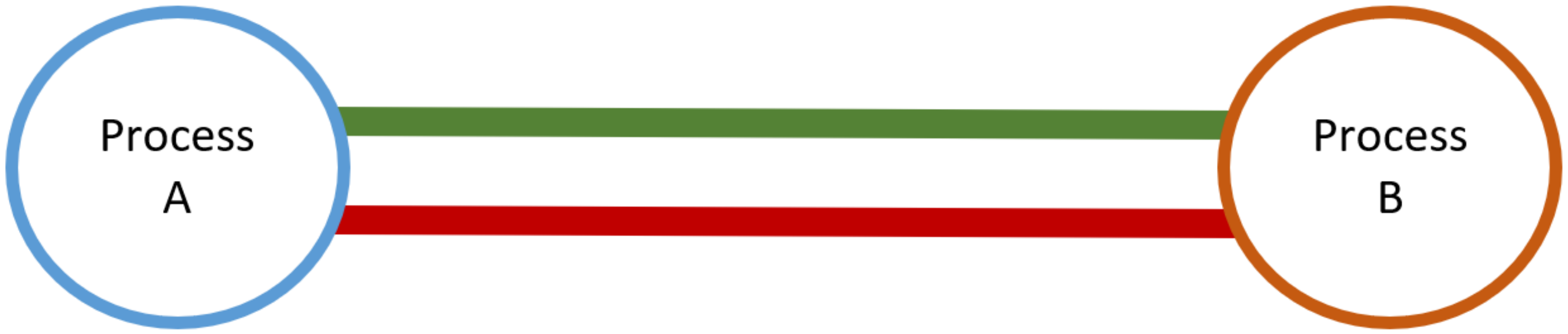
- close()

Reading and Writing

- Finite size
- Read
  - Blocks on empty pipe
  - Otherwise, returns immediately
  - Returns 0 on EOF
- Write
  - Blocks on full pipe
  - Fails if read end not open (SIGPIPE)

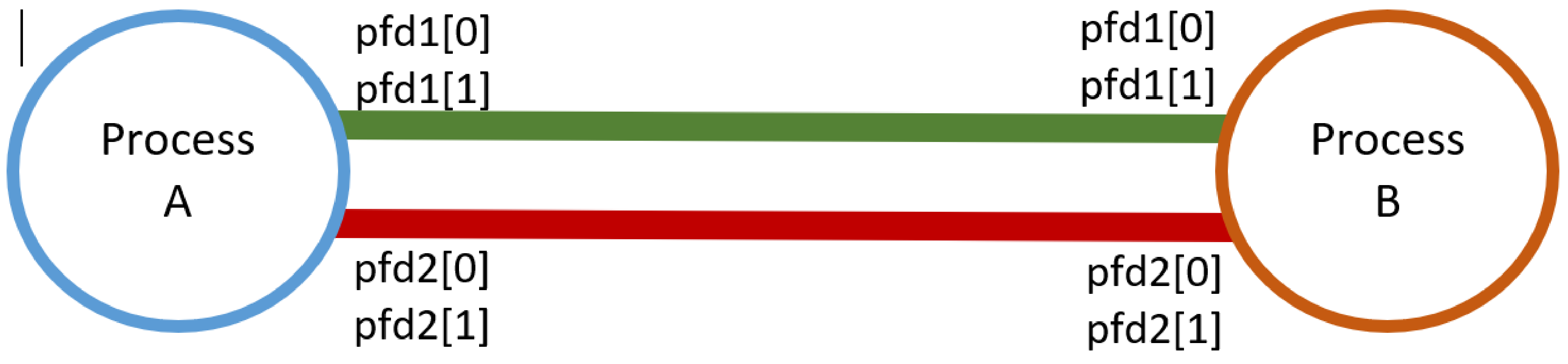
Pipe Synchronization

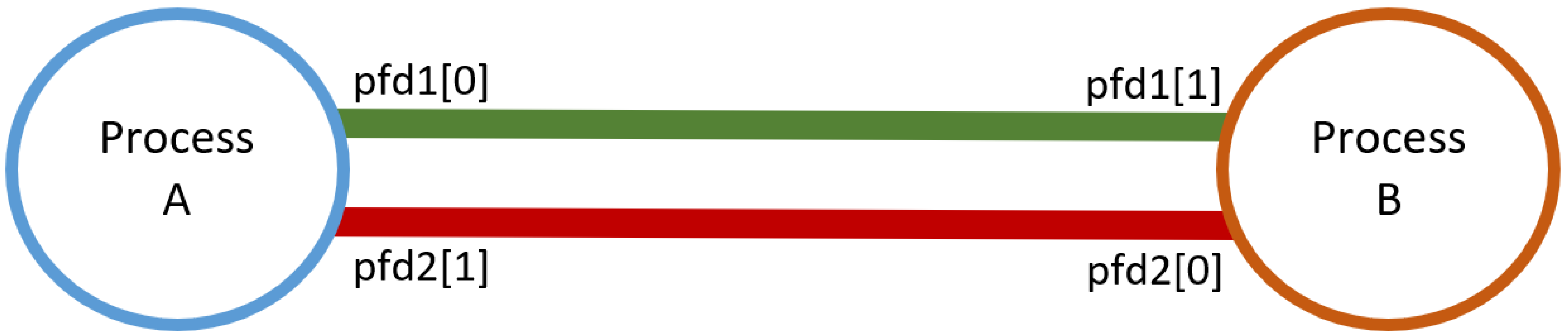
- What must be done if a pipe is used for two-way communication?
- Need two pipes



- Create a barrier or synchronization point

Bi-Directional Communication


Process
A
pfd1[0]
pfd1[1]
pfd2[0]
pfd2[1]
pfd1[0]
pfd1[1]
Process
B
pfd2[0]
pfd2[1]


Process
A
pfd1[0]
pfd2[1]
pfd1[1]
Process
B
pfd2[0]