

Operating Systems Concepts

Implementing IPC



CS 4375, Fall 2025

Instructor: MD Armanuzzaman (*Arman*)

marmanuzzaman@utep.edu

September 15, 2025

Summery

- Threads
 - Concurrent programming
 - Why threads?
 - Threads vs Processes
 - Thread pools
 - Threading implementation & multithreading models
 - Threading issues
 - Semantics of `fork()` and `exec()`
 - Thread cancellation
 - Signal handling

Agenda

- More xv6 system calls
- Inter process communication
 - Pipes
- In class activity
 - Implement IPC through pipes

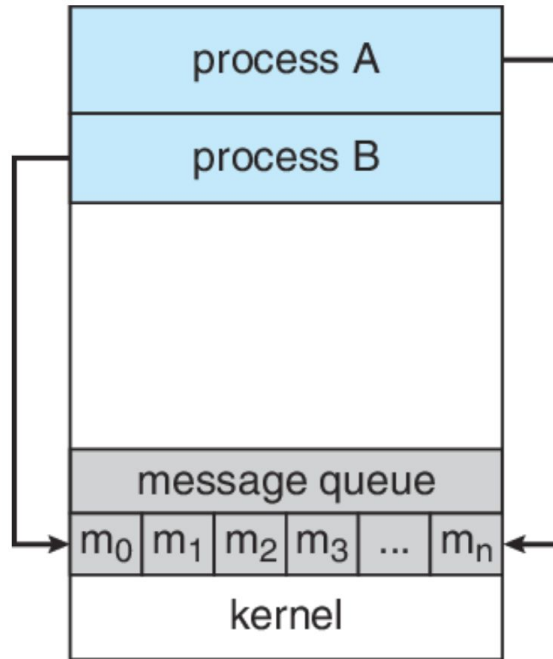
File descriptor

- Unique identifier for a file or I/O resource
- “Small integer representing a kernel-managed object that a process may read from or write to”
- Usage instances
 - Opening a file/directory
 - Opening a device
 - Creating a **pipe**
 - Duplicating a existing *fd*

File descriptor (cont'd)

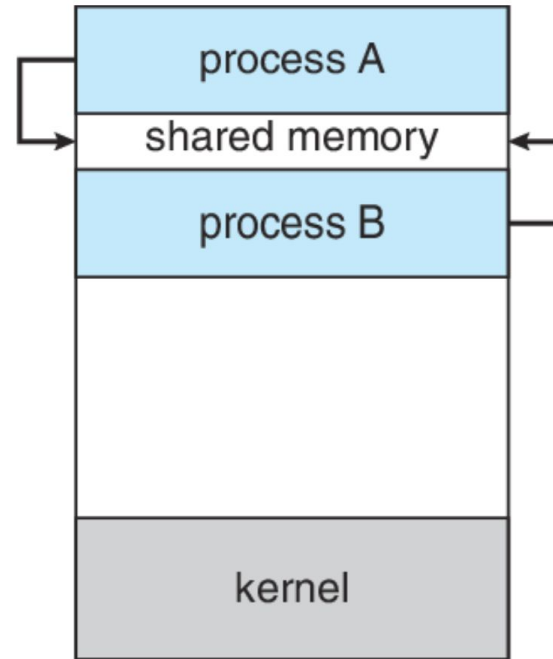
- Standard input ($fd := 0$)
 - Process reads
- Standard output ($fd := 1$)
 - Process writes
- Standard error ($fd := 2$)
 - Process writes error messages

Inter process communication



(a)

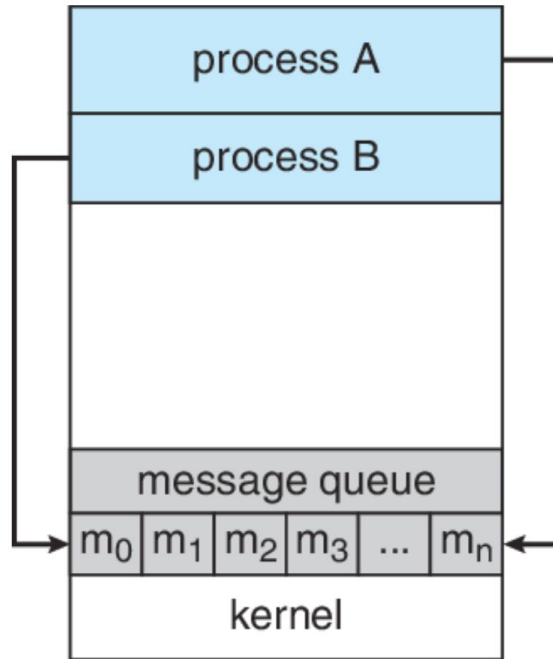
Message Passing



(b)

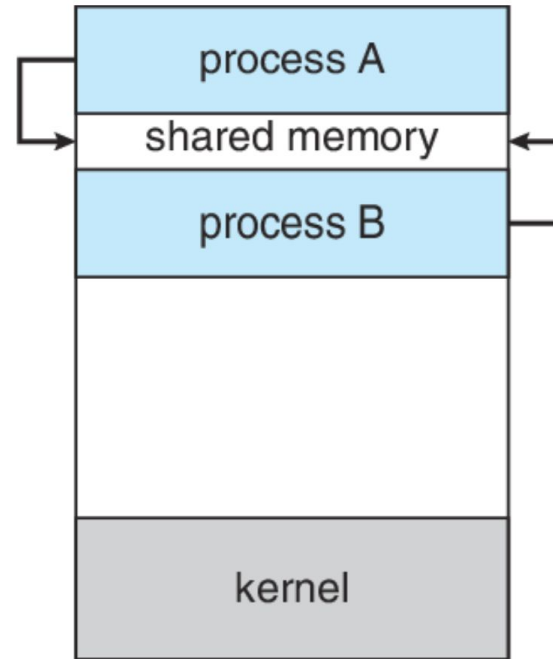
Shared Memory

Inter process communication



(a)

Message Passing



(b)

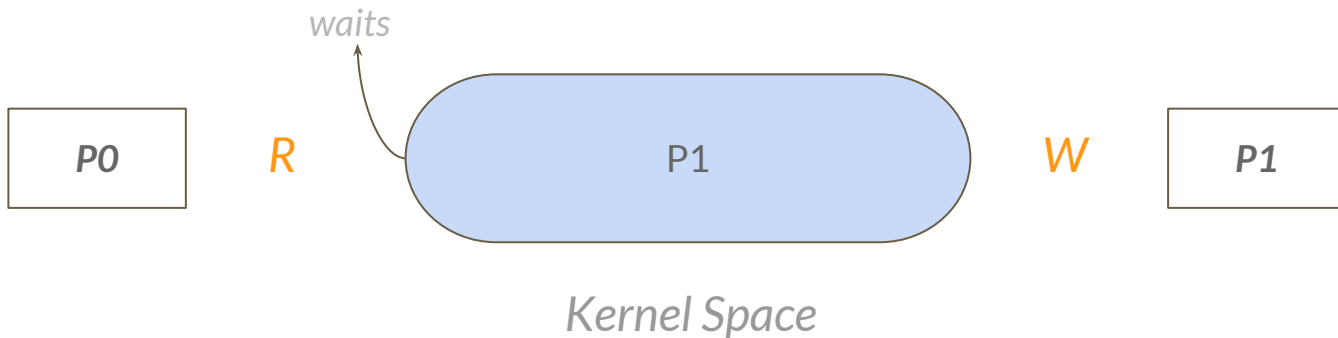
Shared Memory

Pipes

- A pair of small *kernel* buffers for processes
 - As file descriptor
 - Read & Write
- Provide a message passing IPC for two processes

Pipes

- A pair of small *kernel* buffers for processes
 - As file descriptor
 - Read & Write
- Provide a message passing IPC for two processes



In class activity

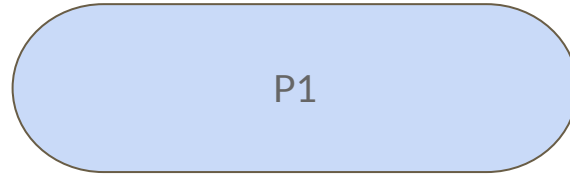
- Write a program that uses UNIX system calls to “*ping-pong*” a byte between two processes over a pair of pipes, *one for each direction*.
- *Optional/take home:* Measure the program’s performance, in exchanges per second.

In class activity

Parent

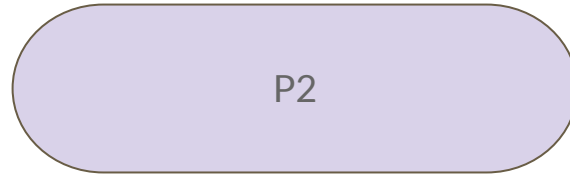
Child

R



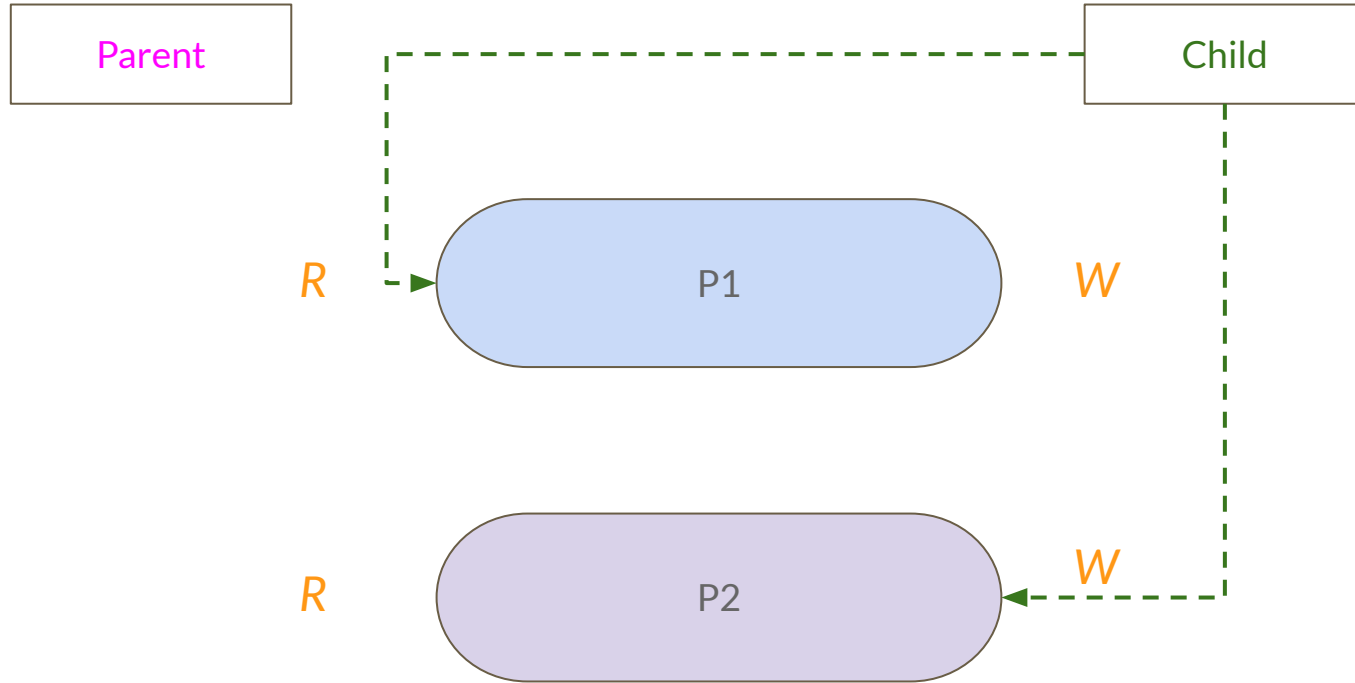
W

R



W

In class activity



In class activity

