Today

- Interprocess Communication Mechanisms

Review

- What guarantees do we have about the order of parent and children processes' execution?
- How do we make a thread stop until its child process finishes?

Interprocess Communication

- Independent process cannot affect or be affected by the execution of another process
- Cooperating process can affect or be affected by the execution of another process
- Advantages of process cooperation
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

Producer-Consumer Problem

- Paradigm for cooperating processes
- ***producer*** process produces information that is consumed by a ***consumer*** process

Interprocess Communication Mechanisms

- Cooperating processes need interprocess communication (IPC)

What are ways to allow the processes to communicate?

Interprocess Communication Mechanisms

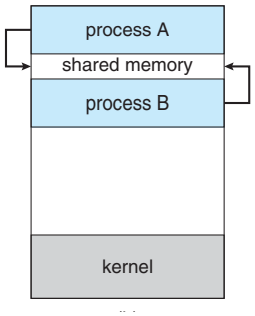
- Interprocess communication (IPC)
  - Shared Memory IPC
  - Message Passing IPC
  - Pipe IPC
  - Sockets

Interprocess Communication: Shared Memory

- An area of memory shared among the processes that wish to communicate
- The communication is under the control of the users processes not the operating system.
- Major issues is to provide mechanism that will allow the user processes to ***synchronize*** their actions when they access shared memory.

IPC via Shared Memory

- Process B uses system calls to create and attach to a shared memory segment.
- Process A uses a system call to map (attach) B's shared memory segment to its own address space.
- Shared memory is then accessed like any other portion of the process' address space.

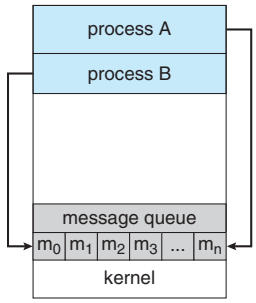


Interprocess Communication – Message Passing

- Mechanism for processes to communicate and to synchronize their actions
- Message system – processes communicate with each other without resorting to shared variables
- IPC facility provides two operations:
  - send(message)
  - receive(message)
- The message size is either fixed or variable

IPC via Message Passing

1. Process A creates a message M.
2. Process A uses a *send* system call to send the message to process B.
3. The message is copied into memory in the kernel's address space.
4. Process B uses a *receive* system call to retrieve the message from A.
5. The message is copied into B's address space.



IPC via Pipes

- Pipes can be used to connect the output of one program to the input of another program:
- Example:
  - ls | grep "Fork"
  - ls | grep "[^*]Demo"
  - ls | grep "[^*]Demo" | sort -r

Sockets

- Connect to a machine and specific port
  - Common ports?
- Use some *protocol* for communication

Sockets

- Connect to a machine and specific port
  - Common ports?
    - 22 : ssh
    - 80: http
- Use some *protocol* for communication
  - http
  - soap

Tradeoffs

- What are some of the relative advantages and disadvantages of these approaches to IPC?

Looking ahead

- Midterm: Wednesday
  - Prep document posted
- Project 2
  - System Calls
    - Interrupts
  - Due in 2 Wednesdays
  - Start on it to help you understand interrupts and system calls
- Monday Office Hours: 3:30-5 p.m.