CMPT 300
Operating System I

4.2 –Process 4
Chapter 3

Dr. Hazra Imran

Inter-Process Communication (IPC)

- So far we have seen **independent** processes
  - Each process runs code independently
  - Parents and aware of their children, and children are aware of their parents, but they do not interact
  - Besides the ability to wait for a process termination

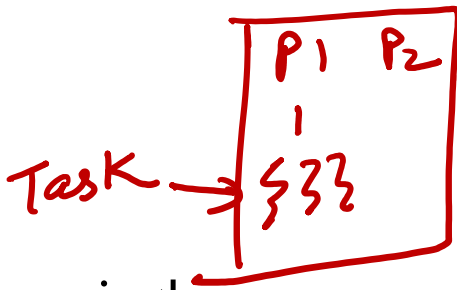
- But often we need processes to **cooperate**
  - To share information (e.g., access to common data) ,To speed up computation (e.g., to use multiple cores)
  - Because it's convenient (e.g., some applications are naturally implemented as sets of interacting processes)

The means of communication between cooperating processes is called **Inter-Process Communication (IPC)**

Inter-Process Communication (IPC)

Advantages of process cooperation

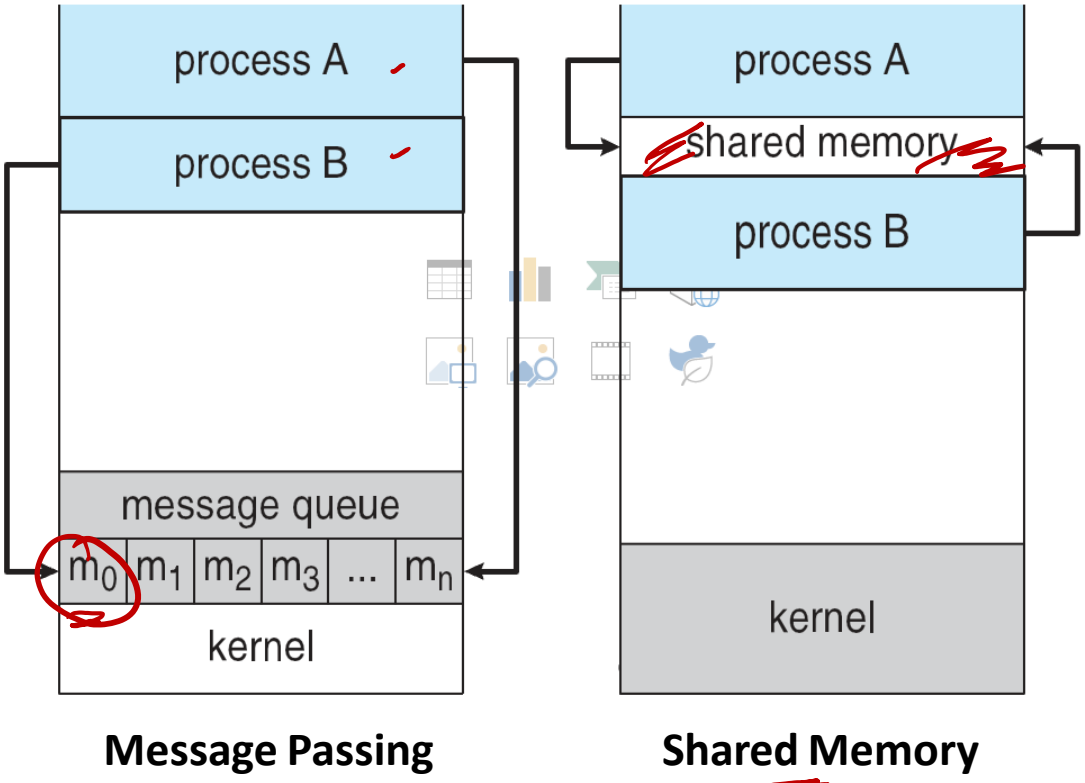
- Information sharing
- Computation speed-up
- Modularity
  - For example, a single server process dedicated to a single client may have multiple threads running--each performing a different task for the client.
- Convenience
  - For example, a network browser is open, while the user has a remote terminal program running (such as telnet), and a word processing program editing data.



Inter-Process Communication (IPC)

Two models of IPC

- **Shared memory**
- **Message passing**



Message Passing **Shared Memory**

Inter-Process Communication (IPC)

Message Passing

- Performed through the kernel memory space
- Simple to implement (pre-defined region in memory)
- One system call per communication operation, i.e., one send, one receive (slower communication)
- Synchronization is easier

Shared Memory

- Performed using available memory
- More difficult to implement: processes need to be aware of the shared memory region's location
- synchronization is very difficult
- fast communication

Inter-Process Communication (IPC)

Other methods used to send data between processes.

- **Shared Memory:** data is sent via block of shared memory visible to both processes
- **Message Passing/Queue:** a queue/data stream provided by the OS to send data between processes
- **File:** data to be shared is written to a file, accessed by both processes
- **Socket:** data is sent via network interface between processes
- **Pipe:** data is sent, unidirectionally, from one process to another via OS-managed data buffer

Reminder

- No class in Friday (June 3)

Next

- Thread (Chapter 4)