Introduction to Operating Systems

1. Introduction
2. Processes and Threads
    - *IPC (Interprocess Communication)*
1. Memory Management
2. File Systems
3. Input / Output
4. Deadlocks

Inter-Process Communication (IPC)

- Why process cooperation (communication)?
  - *Information sharing*
    - *Files, requests, etc.*
  - *Computation speedup*
    - *Multi-processor environments can run different portions of a task to achieve speedup*
  - *Modularity*
    - *Divide the system into several modules and processes/threads for clean implementation purposes*

Inter-Process Communication (IPC)

- Cooperating (communicating) processes can be
  - *Running in the same machine*
    - *This is what we will look at in this course*
  - *Running in different machines*
    - *Must communicate through the network*
    - *Distributed systems – out of the scope of this course*

Inter-Process Communication (IPC)

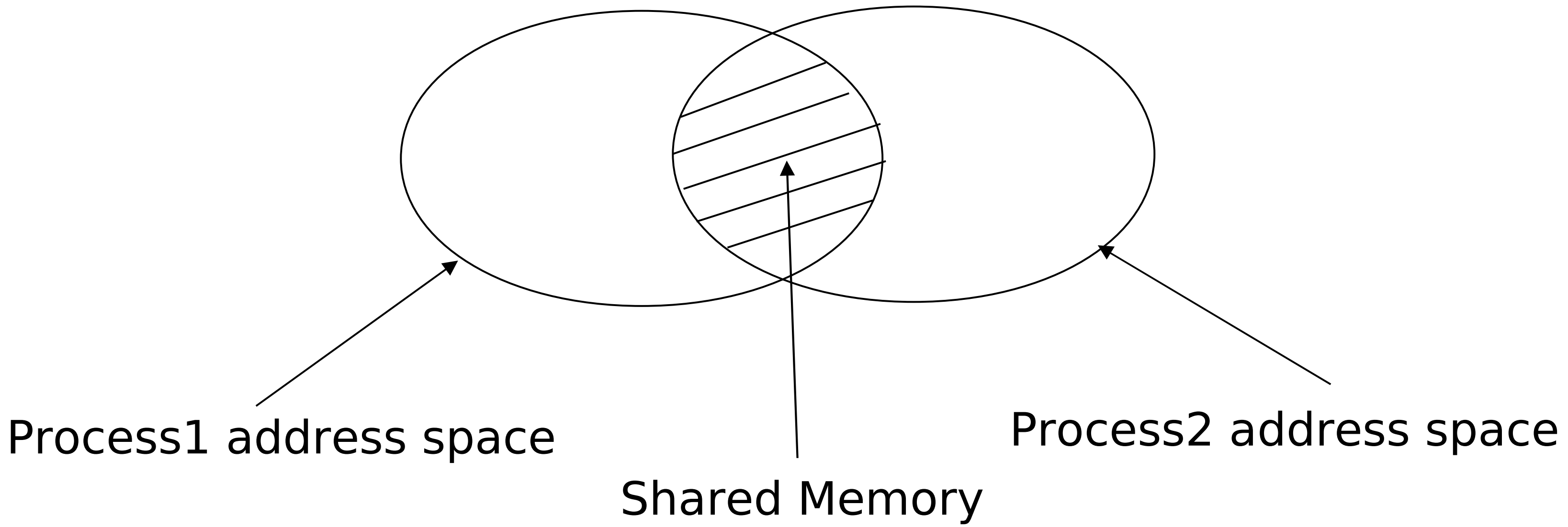
- There are three issues:
  - *How one process can pass information to another.*
  - *How processes are synchronized with making sure one more processes do not get in each other's way. (threads also)*
    - *Two processes in an airline reservation system each trying to grab the last seat on a plane for a different customer.*
  - *How maintain proper sequencing when dependencies are present. (threads also)*
    - *If process A produces data and process B prints them, B has to wait until A has produced some data before starting to print.*

Achieving Data Sharing

- How to achieve data sharing (exchange) between cooperating processes?
  - *Use shared memory*
    - *OS allows different processes to share a portion of the memory by having each process map that memory to their address spaces*
  - *Use message passing*
    - *pipes, mailboxes, sockets, …*
- What about multi-threaded code?
  - *Threads within the same process share everything except the stack space (local variables) and execution state (registers)*
  - *Very easy to implement cooperating threads*
    - *Almost everything is already shared!*
  - *Can't have processes cooperate this way since they don't share the same address space!*

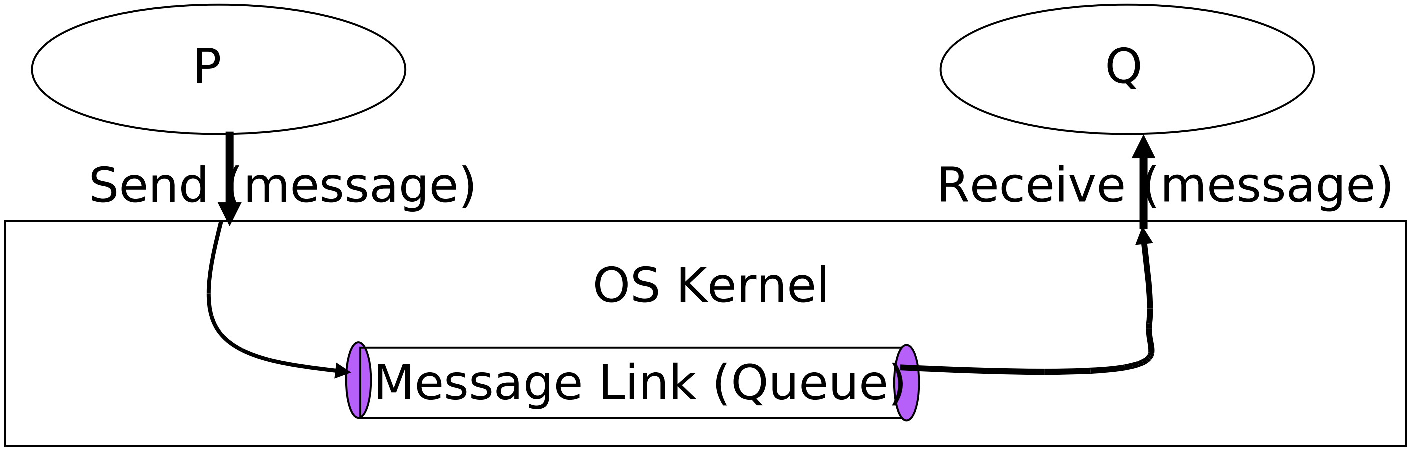
Shared Memory

- OS allows different processes to share a portion of the memory by having each process map that memory to their address spaces
  - *Then all shared data will be kept in the shared memory and all processes sharing that memory segment can access them!*
- We will discuss how shared memory can be implemented in memory management later.



Message Passing (MP)

- Exchange messages using two operations:
  - ***Send***(message) – *message size fixed or variable*
  - ***Receive***(message)
- If *P* and *Q* wish to communicate, they need to:
  - *establish a* communication link *between them*
    - *Typically a logical link implemented in kernel buffers*
  - *exchange messages via send/receive*



MP: Implementation Questions

- How are links established?
- Can a link be associated with more than two processes?
- How many links can there be between every pair of communicating processes?
- What is the capacity of a link?
- Is the size of a message that the link can accommodate fixed or variable?
- Is a link unidirectional or bi-directional?
  - *Pipes* -- *Unidirectional*
  - *Unix Message Queues –- Bi-directional*

MP: Direct Communication

- Processes must name each other explicitly:
  - ***send*** *(*P, message*) – send a message to process P*
  - ***receive****(*Q, message*) – receive a message from process Q*

- Properties of communication link
  - *Links are established automatically.*
  - *A link is associated with exactly one pair of communicating processes.*
  - *Between each pair there exists exactly one link.*
  - *The link may be unidirectional, but is usually bi-directional.*

MP: Indirect Communication

- Messages are directed and received from mailboxes (also referred to as ports, pipes).
  - *Each mailbox has a unique id.*
  - *Processes can communicate only if they share a mailbox.*

- Properties of communication link
  - *Link established only if processes share a common mailbox*
  - *A link may be associated with many processes.*
  - *Each pair of processes may share several communication links.*
  - *Link may be unidirectional or bi-directional.*

MP: Indirect Communication

- Operations
  - *create a new mailbox*
  - *send and receive messages through mailbox*
  - *destroy a mailbox*

- Primitives are defined as:
- **send**(*A*, *message*) – send a message to mailbox A
- **receive**(*A*, *message*) – receive a message from mailbox A

MP: Buffering

- Queue of messages attached to the link; implemented in one of three ways.
  1. *Zero capacity – 0 messages*
     Sender must wait for receiver (rendezvous).
  2. *Bounded capacity – finite length of* n *messages*
     Sender must wait if link full.
  3. *Unbounded capacity – infinite length*
     Sender never waits.