Inter-Process Communication (IPC)

By
Ravindra Raju Kolahalam

What is IPC

- Processes within a system may be independent or cooperating
- Mechanism whereby one process can communicate or exchange data, with another process
- Set of techniques for the exchange of data among multiple threads in one or more process
- Processes may be running on one or more computers connected by a network

Why Should I listen to You?

- Familiarity with the different mechanisms that are available for communicating between two or more processes
- Even quite small applications can benefit from this approach particularly where there could be many users accessing a common resource, such as a database.
- Client/Server computing always uses IPC.
- Net work programming can be easy

Why IPC

- Information sharing
- Resource sharing
- Computation speedup
- Synchronization
- Modularity
- Convenience

IPC Mechanisms

- **Message passing**: Send messages to other processes or receive messages from them.
- **Share a memory** area with other processes.
- **Synchronize** itself with other processes by means of 'Semaphores'.

Message Passing Mechanism

- Establish a communication link
  - physical
  - logical
- Exchange messages two primitive operations for fixed or variable sized message passing
  - Send
  - Receive

Message Passing Systems

- Direct or Indirect communications
- Symmetric or Asymmetric communications
- Automatic or Explicit buffering

Synchronizing Messages

- Message passing may be either blocking or non-blocking.
- blocking send: sender blocked until message received by mailbox or process
- Non blocking send: sender resumes operation immediately after sending
- blocking receive: receiver blocks until a message is available
- Non blocking receive: receiver returns immediately with either a valid or null message.

Synchronizing Messages

- Message passing may be either blocking or non-blocking.
- blocking send: sender blocked until message received by mailbox or process
- Non blocking send: sender resumes operation immediately after sending
- blocking receive: receiver blocks until a message is available
- Non blocking receive: receiver returns immediately with either a valid or null message.

Buffering

- All messaging system require framework to temporarily buffer messages. These queues are implemented in one of three ways:
- Zero capacity – No messages may be queued within the link, requires sender to block until receives retrieves message.
- Bounded capacity – Link has finite number of message buffers. If no buffers are available then sender must block until one is freed up.
- Unbounded capacity – Link has unlimited buffer space, consequently send never needs to block.

IPC Methods

- File
- Signal
- Socket
- Message queue
- Pipe
- Semaphore

File

- A **computer file** is a block of arbitrary information, or resource for storing information.

- A file is durable in the sense that it remains available for programs to use after the current program has finished.

Signal

- **Signal** is a limited form of IPC used in Unix and Unix-like operating systems.
- Essentially it is an asynchronous notification sent to a process in order to notify it of an event that occurred.
- When a signal is sent to a process, the operating system interrupts the process's normal flow of execution.
- Execution can be interrupted during any instruction.

Socket

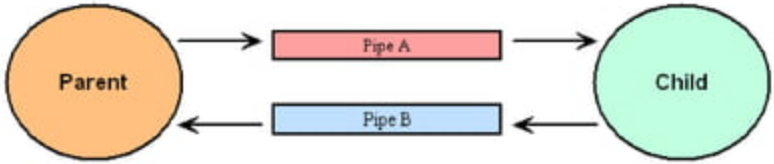
- Sockets provide point-to-point, two-way communication between two processes.
- Sockets are very versatile and are a basic component of inter process and intersystem communication.
- A socket is an endpoint of communication to which a name can be bound.
- It has a type and one or more associated processes.
- Sockets exist in communication domains.
- A socket domain is an abstraction that provides an addressing structure and a set of protocols. Sockets connect only with sockets in the same domain

Message Queues

- Message queues provides an asynchronous communication protocol, meaning that the sender and receiver of the message do not need to interact with the message queue at the same time.
- Messages placed onto the queue are stored until the recipient retrieves them.
- Most message queues have set limits on the size of data that can be transmitted in a single message.

What is PIPE

- A **pipeline** is a set of processes chained by their standard streams, so that the output of each process (*stdout*) feeds directly as input (*stdin*) to the next one.
- Each connection is implemented by an anonymous pipe.



Resources

- http://www.freenetpages.co.uk/hp/alan.gauld/tutipc.htm
- http://linuxgazette.net/104/ramankutty.html

Special Thanks To

- Dr. Way
- Google :P
- All those intelligent guys who published their articles on the internet.