OPERATING SYSTEM

Inter-Process Communication

Introduction

IPC Mechanisms

Introduction

- Communication between processes following particular mechanism
- Processes can be dependent or independent
- Reasons:
  - Sharing Information
  - Task distribution or computation speedup
  - Communication b/w layers of a system
  - Process Synchronization/cooperation

Inter-process Communication

- Wide Variety of inter-process communication (IPC) mechanisms – e.g.,
  - Pipes & streams
  - Sockets & Messages
  - Remote Procedure Call
  - Shared memory techniques
  - OS dependent
- Depends on whether the communicating processes share all, part, or none of an address space

Common IPC mechanisms

- Shared memory – read/write to shared region
  - Need critical section management
  - Shared memory requires that two or more processes agree to remove this restriction of OS not allowing one process to access memory of the other process.
- Semaphores & Monitors – notifies waiting process
  - Shared memory or not, but semaphores need to be shared
- Software interrupts - process notified asynchronously
  - signal ()
- Pipes - unidirectional stream communication
- Message passing - processes send and receive messages
  - Across address spaces
- Remote procedure call – processes call functions in other address spaces
  - Same or different machines

Shared Memory

- Straightforward if processes already share entire address space
  - E.g., threads of one processes
  - E.g., all processes of some operating systems
- Critical section management
  - Semaphores
  - Monitors

Semaphores

- Semaphores can help solve many traditional synchronization problems, BUT:
  - Have no direct relationship to the data being controlled
  - Difficult to use correctly; easily misused
    - Global variables
    - Proper usage requires attention to detail
- Another approach – use programming language support

Monitors

- Programming language construct that supports controlled access to shared data
  - Compiler adds synchronization automatically
  - Enforced at runtime
- Provides
  - Shared data structures
  - Procedures/functions that operate on the data
  - Synchronization between processes calling those procedures
- Only one process *active* inside a monitor at any instant
  - All procedures are part of critical section

Monitors

- Mutual exclusion
  - only one process can be executing inside at any time
  - if a second process tries to enter a monitor procedure, it blocks until the first has relinquished the monitor
- Once inside a monitor, process may discover it is not able to continue
- condition variables provided within monitor
  - processes can **`wait`** or **`signal`** others to continue
  - condition variable can only be accessed from inside monitor
  - **`wait`**'ing process relinquishes monitor temporarily

IPC – Software Interrupts

- Similar to hardware interrupt.
  - Processes interrupt each other
  - Non-process activities interrupt processes
- Asynchronous
  - Keyboard driven
  - An alarm scheduled by the process
  - resource limit exceeded (disk quota, CPU time...)
  - programming errors: invalid data, divide by zero, etc.

IPC – Pipes

- A *pipe* is a unidirectional stream connection between 2 processes
- Introduced in Unix
- A buffer(fixed length messages)/pipe (variable length messages) is established b/w processes & data is shared

IPC – Message Passing

- Communicate information from one process to another via:–
- send(dest, &message)
- receive(source, &message)
- Receiver can specify ANY
- Receiver can choose to block or not
- Applicable to single- and multi-processor and distributed systems
- Does not require shared address spaces!

IPC – Message Passing

- Indirect Communication – mailboxes
  - Messages are sent to a named area – *mailbox*
  - Processes read messages from the mailbox
  - Mailbox must be created and managed
  - Sender blocks if mailbox is full
  - Enables many-to-many communication
  - Acknowledgements:
  - Receiver can reply for the acknowledgement for received message
  - Can be synchronous or asynchronous

Message Passing issues

- Scrambled messages (checksum)
- Lost messages (acknowledgements)
- Lost acknowledgements (sequence no.)
- Destination unreachable (down, terminates)
  - Mailbox full
- Authentication
- Performance

Remote Procedure Call (RPC)

- *The* most common means for communicating among processes of different address spaces
- Used both by operating systems and by applications
  - NFS is implemented as a set of RPCs
- Mechanism:
  - Server processes export an *interface* of procedures/functions that can be called by client programs
  - Clients make local procedure/function calls
  - Procedure/function call is converted into a message exchange with remote server process

Questions