L11: Message passing Model

Dr. Rajashree Dash

Associate Professor,
Department of CSE,
ITER, Siksha O Anusandhan Deemed to be University

Outline

Message Passing Model

- It provides a mechanism for processes to communicate and to synchronize their actions without sharing the same address space.
- It is particularly useful in distributed environment.
- Message passing facility provides two operations:
  send(message)
  receive(message)
- Message passing systems are typically implemented using system calls and thus require the more time-consuming task of kernel intervention
- Message size is either fixed or variable.
- If processes P and Q want to communicate, they must send messages to and receive messages from each other: a communication link must exist between them.

Logical implementation issues of the link

- 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?

Methods used for logically implementing a link and the send()/receive() operations

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

Direct Communication

- Under direct communication, each process that wants to communicate must explicitly name the recipient or sender of the communication.
- Properties of communication link:
  - Links are established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.
  - A link is associated with exactly one pair of communicating processes.
  - Between each pair there exists exactly one link.

Direct Communication

- It may use symmetry in addressing or asymmetry in addressing.

Symmetry in addressing	Asymmetry in addressing
Both the sender process and the receiver process must name the other to communicate.	Only the sender names the recipient; the recipient is not required to name the sender.
send (P, message) – send a message to process P.	send (P, message) – send a message to process P.
receive(Q, message) – receive a message from process Q.	receive(id, message) —Receive a message from any process. The variable id is set to the name of the process with which communication has taken place.

Direct Communication

- Disadvantage: Changing the identifier of a process may necessitate examining all other process definitions. All references to the old identifier must be found, so that they can be modified to the new identifier.

Indirect Communication

- Messages are sent to and received from mailboxes (also referred to as ports).
- Each mailbox has a unique id.
- Processes can communicate only if they share a mailbox.
- Send and receive primitives are as follows:
  send(A, message) – send a message to mailbox A
  receive(A, message) – receive a message from mailbox A
- Properties of communication link:
  - Link is 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.

Indirect Communication

- A mailbox may be owned either by a process or by the operating system.
- If the mailbox is owned by a process, then owner and user should be distinguished. The owner is the process which can only receive messages through this mailbox and the user is the process which can only send messages to the mailbox.
- Since each mailbox has a unique owner, there can be no confusion about which process should receive a message sent to this mailbox.
- When a process that owns a mailbox terminates, the mailbox disappears. Any process that subsequently sends a message to this mailbox must be notified that the mailbox no longer exists.

Indirect Communication

- The process that creates a new mailbox is that mailbox's owner by default.
- Initially, the owner is the only process that can receive messages through this mailbox. However, the ownership and receiving privilege may be passed to other processes through appropriate system calls. Of course, this provision could result in multiple receivers for each mailbox.
- A mailbox that is owned by the operating system has an existence of its own.
- The operating system then must provide a mechanism that allows a process to do the following:
  - Create a new mailbox.
  - Send and receive messages through the mailbox.
  - Delete a mailbox.

Synchronous or asynchronous communication

Message passing may be either blocking or non-blocking.

- Blocking is considered synchronous.
  - Blocking send: The sender is blocked until the message is received by the receiving process or by the mailbox.
  - Blocking receive: The receiver is blocked until a message is available.
- Non-blocking is considered asynchronous.
  - Non-blocking send: The sender sends the message and continues
  - Non-blocking receive: The receiver receives a valid message or Null message.

Buffering

- A link has some capacity that determines the number of messages that can reside in it temporarily. This can be viewed as a queue of messages attached to the link.
- The queue can be implemented in one of three ways:
  - Zero capacity – The queue has maximum length of zero. No messages are queued on a link. Sender must wait for receiver.
  - Bounded capacity – The queue has finite length of n messages. Sender must wait if link is full.
  - Unbounded capacity – The queue has infinite length . Sender never waits.
- The zero-capacity case is sometimes referred to as a message system with no buffering. The other cases are referred to as systems with automatic buffering.