Study Set for Lecture 04: Inter-Process Communication

Due Sep 21 at 11:05pm **Points** 10 **Questions** 7
Available Sep 15 at 12pm - Dec 8 at 8pm 3 months **Time Limit** None
Allowed Attempts Unlimited

Instructions

Download **lect04code.zip** that contains the code shown in the class, unzip it, build and run all examples.

Review lecture notes from **lect04_IPC.pdf**.

Then answer the questions from this study set and submit them to gain access to the further part of the course.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	2,636 minutes	10 out of 10

Correct answers are hidden.

Score for this attempt: **10** out of 10
Submitted Sep 17 at 10:12am
This attempt took 2,636 minutes.

Question 1

0 / 0 pts

What are the main mechanisms for IPC (Inter-Process Communication)? Describe each of them.

Your Answer:

Inter process communication (IPC) is a mechanism which allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other through both:

1. Shared Memory
2. Message passing

These are the methods in IPC:

1. **Pipes (Same Process) –**
   This allows flow of data in one direction only. Analogous to simplex systems (Keyboard). Data from the output is usually buffered until input process receives it which must have a common origin.
2. **Names Pipes (Different Processes) –**
   This is a pipe with a specific name it can be used in processes that don’t have a shared common process origin. E.g. is FIFO where the details written to a pipe is first named.
3. **Message Queuing –**
   This allows messages to be passed between processes using either a single queue or several message queue. This is managed by system kernel these messages are coordinated using an API.
4. **Semaphores –**
   This is used in solving problems associated with synchronization and to avoid race condition. These are integer values which are greater than or equal to 0.
5. **Shared memory –**
   This allows the interchange of data through a defined area of memory. Semaphore values have to be obtained before data can get access to shared memory.
6. **Sockets –**
   This method is mostly used to communicate over a network between a client and a server. It allows for a standard connection which is computer and OS independent.

Question 2

0 / 0 pts

List at least four common uses of IPC (Inter-Process Communication).

Describe each of them.

Your Answer:

Pipes

Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.

Message Passing:

It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables.

IPC mechanism provides two operations:

- Send (message)- message size fixed or variable
- Received (message)

Message Queues:

A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.

Direct Communication:

In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.

Indirect Communication:

Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.

Shared Memory:

Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protected from each other by synchronizing access across all the processes.

FIFO:

Communication between two unrelated processes. It is a full-duplex method, which means that the first process can communicate with the second process, and the opposite can also happen.

Question 3

0 / 0 pts

Describe producer-consumer paradigm explaining the difference between using bounded and unbounded buffers in the communication link.

Your Answer:

Bounded Capacity: The queue has finite length n; thus, at most n message can reside in it. If the queue is not full when the message is sent,, the message is placed in the queue, and the sender can continue executon without waiting.
unbound capacity: the queue's length is potentially infinite; thus, any number of messages can wait in it. The sender never blocks.

In a producer-consumer paradigm, we get that with these two, bounded capacity is better for a instant messaging style, while the unbounded capacity is much better in an email type environment since the instant messaging, people generally have short conversations or the conversations dont require many words to fit in the box.

Question 4

0 / 0 pts

Describe direct and indirect Interprocess Communication.

Your Answer:

Direct Communication: Each process that wants to communicate must explicitly name the recipient or sender of the communcation.
Indirect Interprocess Communication: The messages are sent to and received from mailboxes, or ports.

In a sense, instant messaging vs email.

Question 5

0 / 0 pts

Describe blocking and non-blocking communication. What is a rendezvous?

Your Answer:

Blocking communication is done using `MPI_Send()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Send.html) and `MPI_Recv()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Recv.html) . These functions do not return (i.e., they block) until the communication is finished. Simplifying somewhat, this means that the buffer passed to `MPI_Send()` can be reused, either because MPI saved it somewhere, or because it has been received by the destination.
Similarly, `MPI_Recv()` returns when the receive buffer has been filled with valid data.

In contrast, non-blocking communication is done using `MPI_Isend()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Isend.html) and `MPI_Irecv()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Irecv.html) . These function return immediately (i.e., they do not block) even if the communication is not finished yet. You must call `MPI_Wait()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Wait.html) or `MPI_Test()` (https://www.mpich.org/static/docs/v3.2/www3/MPI_Test.html) to see whether the communication has finished.

Blocking communication is used when it is sufficient, since it is somewhat easier to use. Non-blocking communication is used when necessary, for example, you may call `MPI_Isend()`, do some computations, then do `MPI_Wait()`. This allows computations and communication to overlap, which generally leads to improved performance.

Question 6

0 / 0 pts

Let's a flags `byte` be a collection of 8 one-bit flags.

Explain with details how a one-bit flag that is located in the 5th bit from the right can be set and cleared in the `byte` byte.

Note that the bits are counted from 0.

Your Answer:

Setting

n = 01001001

left shift 5 = 00100000

OR

```
n | (left shift 5) = 01001001
                     00100000
                  =  01101001
```

set the 5th bit from the right.

Clearing

So say we start with a collection of 8 one-bit flags.

n = 01101001

<<

(left shift 5) = 00100000

~(left shift 5) = 11011111

AND

n & ~(left shift 5) = 01101001

11011111

= **01001001**

cleared the 5th bit from right.

Question 7

10 / 10 pts

I have submitted answers to all questions in this study set.

[✓ True]

False

Quiz Score: **10** out of 10