Week3 reading:

Inter process communication:

1. IPC is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of cooperation between them. Processes can communicate with each other through:
    a. Share memory
    b. Message passing
2. Communication between processors using shared memory requires processes to share some variable, and it completely depends on how the programmer will implement it. The example will be like this: processorA and processorB execute simultaneously, they share some resources from another process. ProcessA generates some information about certain computations and keeps it as a record in shared memory. When processB wants the shared information, it will check in the record stored in shared memory and take note of the information generated by processA and act accordingly.
3. Shared memory example: producer-consumer problem:
    a. 假设在一个buffer里面，consumer和producer，一个消耗东西，一个生产东西，假设这个buffer是bounded,producer生产的东西是有限的，必须是consumer先消耗掉一部分，producer才能接着生产。对于consumer来说也是一样的，必须是buffer里面有东西，consumer才能消耗。
4. Messaging passing Method: there is no shared memory. If they communicate with each other, they proceed as follows:
    a. Establish the communication link(if a link already exists, no need to establish it again)
    b. Start exchanging messages using basic primitives:
        i. send(message, destination) or send(message)
        ii. receive(message, host)or receive(message)
5. The message can be variable size or fixed size. A standard message can have two parts: header and body. The header part is used for storing message type, destination id, source id, message length and control information. The control information contains information like what to do if run out of buffer space, sequence number, priority. Generally, messages are sent using the FIFO style.
6. How to implement communication links? We need to figure out the following questions
    a. How are links established?
    b. Can a link be associated with more than 2 processes?
    c. How many links can there be between every pair of communicating processes?
    d. What is the capacity of a link? Is the size of a message that the link can accommodate fixed or variable?
    e. Is a link unidirectional or bi-directional?

7. 每一个link都有一个capacity规定message的数量。every link has a queue associated with it which can be zero capacity, bounded capacity or unbounded capacity. In zero capacity, the sender waits until the receiver informs the sender that it has received the message. In non-zero capacity, a process does not know whether a message has been received or not after the send operation. For this, the sender must communicate with the receiver explicitly. Implementation of the link depends on the situation, it can either be a direct communication link or an in-directed communication link.
8. Direct communication links are implemented when the processes use a specific identifier for the communication, but it's hard to identify the sender ahead of time. In-direct communication is done via a shared mailbox(port), which consists of a queue of messages. The sender keeps the message in the mailbox and the receiver picks them up.
9. IPC is possible between processes on the same computer or on the processes running on different computers. A process is blocked when it's waiting for some event. Message passing could be blocking or non-blocking.
    a. Blocking is considered synchronous and blocking send means the sender will be blocked until the message is received by the receiver. Blocking receive has the receiver block until a message is available.
    b. Non-blocking is considered asynchronous and non-blocking send has the sender sends the message and continues. Non-blocking receive has the receiver receive a valid message or null.
    c. We can come to a conclusion that for a sender it is more natural to be non-blocking after message passing as there may be a need to send the message to different processes. However, the sender expects acknowledgement from the receiver in case the send fails.
    d. Similarly, it is more natural for a receiver to be blocked after issuing the receiver as the information from the received message may be used for further execution.
    e. There are basically three preferred combinations:
        i. Blocking send and blocking receive
        ii. Non-blocking send and non-blocking receive
        iii. Non-blocking send and blocking receive(mostly used)
10. In direct message passing, the process which wants to communicate must explicitly name the recipient or sender of the communication
    a. send(p1, message) means send the message to p1.
    b. receive(p2, message) means receive the message from p2.

在direct message passing情况下， the commnucation link get established automatically, which can be either unidirectional or bidirectional. One link can be used between a pair of sender and receiver and this pair should not possess more than one pair of links.

11. For indirect message passing, processors use mailbox (ports) for sending and receiving messages. Each mailbox has a unique ID, and processors can communicate only if they share the same mailbox. Each pair of processors can share several communication links and these links may be unidirectional or directional.
    a. The operations for indirect communications are:

        i. Create a mailbox
        ii. Use this mailbox for sending and receiving messages.
        iii. Destroy the mailbox
    b. 假设多个pair of processors share the same box, after sending the message,怎么知道receiver是谁？有三种策略：
        i. 只允许一对processor share the same mailbox
        ii. 在同一时间，只有一个processor可以接受
        iii. Randomly选一个receiver接受，然后告诉sender receiver是谁
12. The port is owned by the receiving process and created by OS on the request of the receiver process and can be destroyed either by the sender process or the receiver process.
13. 如果用上述第二种机制的话，可以用mutex (mutual exclusion). Mutex box is created which is shared by n processes. The sender is non-blocking and sends the message. The first process which executes the receive will enter in the critical section and all other processes will be blocking and waiting.
14. 如果用message passing, consumer -producer problem将会这么解决：
    a. The producer places items(messages) in the mailbox
    b. The consumer can consume an item when at least one message is present in the mailbox.
15. Examples of IPC systems:
    a. Posix
    b. Mach
    c. Windows XP:
16. Communication in client/ server architecture:
    a. Pipe
    b. Socket
    c. RPC

Difference between Mac and IP address:

1. An internet is made up of a combination of physical networks (LAN or WAN) connected by routers. When a host communicates with another host, the packets may travel from one physical network to another using these routers. This suggests that communication at this level also needs some global identification system. A host must be able to communicate with any other host without worrying about which physical network must be passed through. This means that the hosts must be identified uniquely and globally at this layer also. In addition, for efficient and optimum routing, each router must also be identified uniquely and globally at this layer.
2. The identifier that is used in the IP layer of the TCP/IP protocol is called the INTERNET PROTOCOL ADDRESS OR THE IP ADDRESS.
3. Mac address: However , packets must pass through physical networks to reach these hosts and routers. At the physical level, the hosts and routers are recognized by their

physical address. A physical address is a local address. Its jurisdiction is a local network. It should be unique locally, but not necessary universally.

Java serialization and deserialization:

1. Classes ObjectInputStream and ObjectOutputStream are high-level streams that contain the methods for serializing and deserializing an object.
2. Notice that for a class to be serialized successfully, two conditions must be met: The class must implement the java.io.Serializable interface. All of the fields in the class must be serializable. If a field is not serializable, it must be marked transient.

IPC overview:

1. Methods for effective sharing of information among cooperating processes are collectively known as interprocess communication (IPC). Two basic models are used: – shared memory—"shared data" are directly available to each process in their address spaces. – message passing—"shared data" are explicitly exchanged.
2. A common approach in communication is where one process sends some information to another. The information exchanged among processes in this way is called a message. A message can be a structured (language) object, specified by its type, or it is specified by its size: fixed length or variable length. There are two basic operations on messages: – send()—transmission of a message. – receive()—receipt of a message.
3. • Form of communication—messages can be send directly to its recipient or indirectly through an intermediate named object. • Buffering—how and where the messages are stored. • Error handling—how to deal with exception conditions.
4. Direct message communication: The sender and receiver can communicate in either of the following forms: • synchronous—involved processes synchronize at every message. Both send and receive are blocking operations. This form is also known as a rendezvous. • asynchronous—the send operation is almost always non-blocking. The receive operation, however, can have blocking (waiting) or non-blocking (polling) variants. A link is established automatically, but the processes need to know each other's identity. • A unique link is associated with the two processes. • Each pair of processes has only one link between them. • The link is usually bi-directional.
5. Indirect message communication: In case of indirect communication, messages are sent to mailboxes, which are special repositories. A message can then be retrieved from this repository. – send (A, message). Send a message to mailbox A. – receive (A, message). Receive a message from mailbox A. This form of communication decouples the sender and receiver, thus allowing greater flexibility. Generally, a mailbox is associated with many senders and receivers. In some systems, only one receiver is (statically) associated with a particular mailbox; such a mailbox is often called a port.

6. The interconnection between the sender and receiver has the following characteristics: • A link is established between two processes only if they “share” a mailbox. • A link may be associated with more than two processes. • Communicating processes may have different links between them, each corresponding to one mailbox. • The link is usually bi-directional, but it can be unidirectional.
7. Error handling: In distributed systems, message passing mechanisms extend inter-process communication beyond the machine boundaries. Consequently, messages are occasionally lost, duplicated, delayed, or delivered out of order. The following are the most common ‘‘error’’ conditions which requires proper handling: • Process terminates—either a sender or a receiver may terminate before a message is processed. • Lost or delayed messages—a message may be lost (or delayed) in the communications network. • Scrambled messages—a message arrives in an unprocessable state.

Core Concepts and Terminology:

1. An ORB or RPC is a mechanism for invoking operations on an object (or calling a procedure) in a different (“remote”) process that may be running on the same, or a different, computer.
2. Many people refer to CORBA as middleware or integration software. This is because CORBA is often used to get existing, stand-alone applications communicating with each other.
3. One of CORBA’s strong points is that it is distributed middleware. In particular, it allows applications to talk to each other even if the applications are: • On different computers, for example, across a network. • On different operating systems. CORBA products are available for many operating systems, including Windows, UNIX, IBM mainframes and embedded systems. • On different CPU types or Implemented with different programming languages, such as, C, C++, Java, Smalltalk, Ada, COBOL, PL/I, LISP, Python and IDLScript1
4. CORBA is also an object-oriented, distributed middleware. This means that a client does not make calls to a server process. Instead, a CORBA client makes calls to objects (which happen to live in a server process).
5. In CORBA terminology, a server is a process that contains objects, and a client is a process that makes calls to objects. A CORBA application can be both a client and a server at the same time.

IPC tutorial:没看完 有点看不懂

1. Programs can also have a point where control splits into two independent lines, an action called forking. A call to the system routine fork(), causes a process to split in this way. The result of this call is that two independent processes will be running, executing exactly the same code.
2. A process views the rest of the system through a private table of descriptors. The descriptors can represent open files or sockets. Descriptors are referred to by their index numbers in the table. The first three descriptors are often known by special names, stdin, stdout and stderr. These are the standard input, output and error. When a process forks, its descriptor table is copied to the child. Thus, if the parent's standard input is being taken from a terminal (devices are also treated as files in UNIX), the child's input will be taken from the same terminal.
3.