Homework  Chapter 3 solutions

1.  
   A.  **Synchronous and asynchronous communication**—A benefit of synchronous communication is that it allows a rendezvous between the sender and receiver. A disadvantage of a blocking send is that a rendezvous may not be required and the message could be delivered asynchronously. As a result, message-passing systems often provide both forms of synchronization.
   
   B.  **Automatic and explicit buffering**—Automatic buffering provides a queue with indefinite length, thus ensuring the sender will never have to block while waiting to copy a message. There are no specifications on how automatic buffering will be provided; one scheme may reserve sufficiently large memory where much of the memory is wasted. Explicit buffering specifies how large the buffer is. In this situation, the sender may be blocked while waiting for available space in the queue. However, it is less likely that memory will be wasted with explicit buffering.
   
   C.  **Send by copy and send by reference**—Send by copy does not allow the receiver to alter the state of the parameter; send by reference does allow it. A benefit of send by reference is that it allows the programmer to write a distributed version of a centralized application. Java’s RMI provides both; however, passing a parameter by reference requires declaring the parameter as a remote object as well.
   
   D.  **Fixed-sized and variable-sized messages**—The implications of this are mostly related to buffering issues; with fixed-size messages, a buffer with a specific size can hold a known number of messages. The number of variable-sized messages that can be held by such a buffer is unknown. Consider how Windows 2000 handles this situation: with fixed-sized messages (anything < 256 bytes), the messages are copied from the address space of the sender to the address space of the receiving process. Larger messages (i.e. variable-sized messages) use shared memory to pass the message.

2.  
   If an RPC mechanism cannot support either the “at most once” or “at least once” semantics, then the RPC server cannot guarantee that a remote procedure will not be invoked multiple occurrences. Consider if a remote procedure were withdrawing money from a bank account on a system that did not support these semantics. It is possible that a single invocation of the remote procedure might lead to multiple withdrawals on the server. For a system to support either of these semantics generally requires the server maintain some form of client state such as the timestamp described in the text. If a system were unable to support either of these semantics, then such a system could only safely provide remote procedures that do not alter data or provide time-sensitive results.
Using our bank account as an example, we certainly require “at most once” or “at least once” semantics for performing a withdrawal (or deposit!). However, an inquiry into an account balance or other account information such as name, address, etc. does not require these semantics.

6. The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time. If the context is in memory, one of the contexts in a register set must be chosen and be moved to memory, and the new context must be loaded from memory into the set. This process takes a little more time than on systems with one set of registers, depending on how a replacement victim is selected.

9.
A. **Short-term** (CPU scheduler)—selects from jobs in memory those jobs that are ready to execute and allocates the CPU to them.
B. **Medium-term**—used especially with time-sharing systems as an intermediate scheduling level. A swapping scheme is implemented to remove partially run programs from memory and reinstate them later to continue where they left off.
C. **Long-term** (job scheduler)—determines which jobs are brought into memory for processing.

The primary difference is in the frequency of their execution. The short-term must select a new process quite often. Long-term is used much less often since it handles placing jobs in the system and may wait a while for a job to finish before it admits another one.

11. \[A = 0, B = 2603, C = 2603, D = 2600\]

12. The result is still 5 as the child updates its copy of value. When control returns to the parent, its value remains at 5.