**Practice Exercises**

3.1 Palm OS provides no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

**Answer:**

a. A method of time sharing must be implemented to allow each of several processes to have access to the system. This method involves the preemption of processes that do not voluntarily give up the CPU (by using a system call, for instance) and the kernel being reentrant (so more than one process may be executing kernel code concurrently).

b. Processes and system resources must have protections and must be protected from each other. Any given process must be limited in the amount of memory it can use and the operations it can perform on devices like disks.

c. Care must be taken in the kernel to prevent deadlocks between processes, so processes aren’t waiting for each other’s allocated resources.

3.2 The Sun UltraSPARC processor has multiple register sets. Describe the actions of a context switch if the new context is already loaded into one of the register sets. What else must happen if the new context is in memory rather than in a register set and all the register sets are in use?

**Answer:** 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.

3.3 When a process creates a new process using the fork() operation, which
of the following state is shared between the parent process and the child
process?

a. Stack
b. Heap
c. Shared memory segments

**Answer:** Only the shared memory segments are shared between the
parent process and the newly forked child process. Copies of the stack
and the heap are made for the newly created process.

3.4 Again considering the RPC mechanism, consider the “exactly once” se-
monic. Does the algorithm for implementing this semantic execute cor-
rectly even if the “ACK” message back to the client is lost due to a network
problem? Describe the sequence of messages and whether “exactly once”
is still preserved.

**Answer:** The “exactly once” semantics ensure that a remote procedure
will be executed exactly once and only once. The general algorithm for
ensuring this combines an acknowledgment (ACK) scheme combined
with timestamps (or some other incremental counter that allows the
server to distinguish between duplicate messages).

The general strategy is for the client to send the RPC to the server along
with a timestamp. The client will also start a timeout clock. The client
will then wait for one of two occurrences: (1) it will receive an ACK from
the server indicating that the remote procedure was performed, or (2) it
will time out. If the client times out, it assumes the server was unable
to perform the remote procedure so the client invokes the RPC a second
time, sending a later timestamp. The client may not receive the ACK for
one of two reasons: (1) the original RPC was never received by the server,
or (2) the RPC was correctly received—and performed—by the server
but the ACK was lost. In situation (1), the use of ACKs allows the server
ultimately to receive and perform the RPC. In situation (2), the server will
receive a duplicate RPC and it will use the timestamp to identify it as a
duplicate so as not to perform the RPC a second time. It is important to
note that the server must send a second ACK back to the client to inform
the client the RPC has been performed.

3.5 Assume that a distributed system is susceptible to server failure. What
mechanisms would be required to guarantee the “exactly once” seman-
tics for execution of RPCs?

**Answer:** The server should keep track in stable storage (such as a
disk log) information regarding what RPC operations were received,
whether they were successfully performed, and the results associated
with the operations. When a server crash takes place and a RPC message
is received, the server can check whether the RPC had been previously
performed and therefore guarantee “exactly once” semantics for the
execution of RPCs.