**Chapter 9**

POSIX threads and mutexes

As you read Chapter 9, you should compile and run the example code.  By the nature of multi-threaded programs, you results will probably be different from mine, possibly in interesting ways.

1) Why does each thread have its own stack?

To keep track of individual identification information as well as allow threads to access functions without causing simultaneous access problems, thus making memory access time quicker.

2) What does the gcc flag -lpthread do?

Accesses/ links to the pthread library.

3) What does the argument of pthread\_exit do?

Allows transference of computed information to the next joined thread.

4) Normally the same thread that created a thread also waits to join it.  What happens if another thread tries to join a thread it did not create?

Threads can be joined regardless of its creator, though the most common pattern is this aforementioned parent-child joining.

5) What goes wrong if several threads try to increment a shared integer at the same time?

Synchronization goes out of wack and you get incorrect answers since the various threads will read and write in unpredictable order.

6) What does it mean to "lock a mutex"?

A locked mutex prevents other threads from executing the following lines of code. This allows the programmer to control the number of simultaneous thread executions.