Homework #2

1. Explain the tradeoffs between using multiple processes and using multiple threads

2. Does a multi-threading solution always improve performance? Please explain your answer and give reasons.

3. Explain the tradeoffs between preemptive scheduling and nonpreemptive scheduling

4. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

5. What would be a possible problem if you executed the following program (and intend for it to run forever)? How can you solve it?

```c
#include <signal.h>
#include <sys/wait.h>
int main()
{
    for (;;) {
        if (!fork()) { exit(0); }
        sleep(1);
    }
    return 0;
}
```

6. Consider the following program:

```c
#include <stdlib.h>
int main ()
{
    fork ();
    if (fork ()) {
        fork ();
    }
    fork ();
```
Draw a tree diagram showing the hierarchy of processes created when
the program executes. How many total processes are created (including
the first process running the program)?

Hint: You can always add debugging code, compile it, and
run the program to experiment with what happens.

7. Show how a lock, and acquire() and release() functions can be
implemented using atomic SWAP instruction. The following is the
definition of swap instruction:

```c
void swap (char* x, char * y) // All done atomically
{
    char temp = *x;
    *x = *y;
    *y = temp
}
```

```c
struct lock {
}
```

```c
void acquire (struct lock *) {
}
```

```c
void release (struct lock *) {
}
```

8. Suppose you have an operating system that has only binary semaphores.
You wish to use counting semaphores. Show how you can implement
counting semaphores using binary semaphores.

Hints: You will need two binary semaphores to implement
one counting semaphore. There is no need to use a queue
the queuing on the binary semaphores is all you’ll need.
You should not use busy waiting. The wait() operation for
the counting semaphore will first wait on one of the two bi-
nary semaphores, and then on the other. The wait on the
first semaphore implements the queueing on the counting semaphore and the wait on the second semaphore is for mutual exclusion.