Graham Leslie

CSCE 313

Dr. Bettati

5 March 2013

Homework 2

1. **What are asynchronous, deferred, and disabled cancellation in pthreads? What makes them different?**

   Asynchronous cancellation can occur at any time, and is often used to cancel threads immediately. Deferred cancellation will be delivered when the thread reaches it the next function that is classified as a cancellation point. Disabled cancellation will be queued by the thread until it returns to enabled cancellation.

2. **What is the role of the *contentionscope* for pthreads? How does it affect the execution of pthreads?**

   contentionscope is the scope of a *pthread* as either user-level or kernel-level; it can be changed after a *pthread* is created to change its scope. The execution will change depending on the contention scope in several ways. First, there is a limit on the number of available kernel threads, so it may not be possible to change a pthread to *PTHREAD_SCOPE_SYSTEM* if the number of kernel threads is already full.

3. **usleep() C++-style snippit**

```
// pass in time with arg _time, assume tv_musec
exists, condition variable is c and mutex is m

pthread_mutex_lock(&m)
cond_relative_timedwait(&c, &m, &time);
pthread_mutex_unlock(&m);
```

4. **The *execve* system call**

   a. What happens to the file descriptors of open files?

      Normally, the file descriptors remain open across an *execve()* call.

   b. How can this behavior be controlled?

      Using the flag *FD_CLOEXEC* will cause the file descriptors to close across an *execve()* call.

5. **Ten processes share a critical section implemented by using a semaphore *x*. Nine of these process use the code P(x); <critical section>; V(x). However, one process erroneously uses the code V(x); <critical section; P(x). What is the maximum number of processes that can be in the critical section at one time?**

   Given the problem, at most three threads can enter the critical section at one time. Assume it is the erroneous process's turn and $x = 1$: the erroneous function calls V(x), and $x = 2$. Now, two more processes will come along and get to enter as $x = 1$ and then $x = 0$.

6. **SafeAccount Java Class:**

```
public class SafeAccount {

    // members
    private static int balance;
    private static final int N;
```

```
     private final Semaphore lock = new Semaphore(100,
true);

     // methods
     public SafeAccount(int _n) {
          N = _n;
     }

     // do not credit if balance exceeds n
     public void credit(int _amount) {
          lock.acquire();
          /* critical section */
          if (balance < N) balance += _amount;
          /* /critical section */
          lock.release();
     }

     // hold debits until acct is large enough to
permit
     public void debit(int _amount) {
          lock.acquire();
          /* critical section */
          if (balance >= _amount) balance -= _amount;
          /* /critical section */
          lock.release();
     }
}
```