Homework 2
CSCE 313-504

Derek Heidtke

October 14, 2015

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

All pthreads have two settings: cancelability state, and cancelability type. The pthreads state can be either enabled or disabled; this setting determines whether a call to pthread_cancel() will result in the cancellation of the thread or not. Now, if cancellation is enabled, the call to pthread_cancel() can be either deferred (default behavior), or asynchronous. Deferred cancellation means the thread will be canceled the next time it calls a function that contains a cancellation point. Asynchronous cancellation simply cancels the thread immediately.

2. What is the role of the contention scope for pthreads? How does it affect the execution of pthreads?

The contention scope for pthreads controls whether the thread must compete for resourses among threads in its same process, or threads and processes across the whole system; respectively bounded or unbounded.

3. Assume you have a system that does not provide a usleep(unsigned long usec) call to suspend the execution of the thread for a given amount of time, say in usecs. How would you implement this function using condition variables? (Describe your implementation in form of a C/C++ code snippet.)

```
#include <pthread.h>

unsigned WAIT_TIME = 2; // wait time in seconds (e.g. 2 sec)

pthread_cond_t * _wait;
pthread_mutex_t * _mutex;
const struct timespec * reftimes;
struct timeval * reftimev;
```

```
// Convert from timeval to timespec
reftimes.tv_sec = reftimev.tv_sec;
reftimes.tv_nsec = reftimev.tv_usec*1000;
reftimes.tv_sec += WAIT_TIME;
```

4. Which of the scheduling algorithms listed below could result in starvation?

- a) First-come, first-served

No starvation; every process eventually gets processor time.

- b) Shortest job first

Starvation possible; a long process could potentially have a never-ending stream of short processes cut in front of it.

- c) Round robin

No starvation; because of the limited time limit (time quantum), all processes get the same amount of processor time until it finishes.

- d) Fixed priority

Starvation possible; if a process ends up with a low-priority, there is no way to change that, so it will likely get stuck behind a perpetual train of higher-priority processes.

5. Servers can be designed to limit the number of open connections. For example, a server may wish to have only N connections active at any time. As soon as N connections are established, the server will not handle more connections until at least one existing connection is closed. How would you use semaphores to limit the number of concurrent connections? Assume that you have a function called HandleNextIncomingConnection() and one called CloseExistingConnection(), which are called before and at the end of handling a connection. How would you add the necessary synchronization code to limit the number of concurrent connections to at most N?

First, initialize a semaphore, _connections, with the value of N. Then, do:

```
_connections.P();
HandleNextIncomingConnection();
// ...
CloseExistingConnection();
_connection.V();
```

So, the server will continue to handle connections until the critical section is full, at which point the N+1 request causes the server to block (on the P() call). Then, when the critical section becomes not-full, the server becomes unblocked (after the V() call) and is allowed to admit more connections.

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

Because the incorrect code increments the semaphore instead of decrementing it, the count on the semaphore is off by two. Now, N+2 threads are allowed in the critical section when only N were allowed in before. In general, whenever the incorrect code runs, the maximum allowed number of threads will increase by 2.

7. A customer gives the following instructions to a bank manager: Do not credit any funds to my account if the balance in my account exceeds n, and hold any debits until the balance in the account is large enough to permit the debit. Design a class in Java (called SafeAccount, to be initialized with a given value for n) with two methods, credit and debit, that implements this type of account.