Homework 15: Threads

Instructor: Mehmet Emre

CS 32 Spring '22

Due: 6/1 12:30pm
Name & Perm #: Bharat Kathi (5938444)

Homework buddy (leave blank if you worked alone):

Reading: Chapters 25, 26, and 28.1 of Operating Systems: Three Easy Pieces (you can download them from the web page). The book uses a slightly different API for threads (POSIX threads rather than C++ thread API), and you don't need to learn the details of that API besides the main points the book explains.

1 (2 pts) What is the advantage of using threads over processes?

Answer: Threads are a lot less resource intensive and generate less overhead when compared to proccesses. Context switches are also much quicker between threads than between proccesses.

2

Arpaci-Dusseaus give an example program in Figure 26.6, and discuss that program in detail throughout section 3.

1. (1 pt) Their example has the same issue as the first version of the bank account program we looked at in the lecture. What is the name of the specific problem both programs share?

   Answer: A race condition, specifically a data race.

2. (2 pts) What is the root cause of this problem?

   Answer: Multiple threads are trying to access a shared resource in a critical section of code, resulting in a data race.

3. (2 pts) How do locks solve the problem?

   Answer: Locks ensure that the first thread finishes executing in a critical section before allowing another thread to enter that section.

3

Suppose we have a mutex lock `m` and a piece of code like the following:

```
m.lock(); // (*)
// ...
m.unlock(); // (**)
```

Also assume that there are two threads in the program: `T1` and `T2`. Suppose `T1` starts running first and reaches the code marked with `...` (so it has already acquired the lock).

1. (1 pt) Suppose `T2` starts running the line marked with `(*)` while `T1` is running the code marked with `....` What happens when `T2` runs the first line?

   Answer: T2 will be waiting at the (*) line since the .lock() function will not return until T1 releases it.

2. (1 pt) After `T2` runs the first line of code, `T1` reaches the last line (marked with `(**)`). What will happen with respect to the execution of `T2`?

   Answer: Now that T1 has reached the (**) line, the lock is released and T2 can now aquire the lock and continue its execution into the ... section.