PA05

Jayson Grace

Sunday 18th October, 2015

Jayson Grace

Problem 1a

Upon running the code, I was able to determine that the balance was not what I was expecting it to be. Instead of staying at the initial balance of 200, it would either be too big or too small. After inspecting the code, I saw that there were a number of floating point calculations happening between the calculation of balances A and B:

```
25        Bank.balance[0] = tmp1 + rint;
26        for (j = 0; j < rint * 100; j++)
27        {
28          dummy = 2.345 * 8.765 / 1.234;
29        }
30        Bank.balance[1] = tmp2 - rint;
```

To determine if this was causing an issue, I moved the Balance calculation for B above the loop:

```
25        Bank.balance[0] = tmp1 + rint;
26        Bank.balance[1] = tmp2 - rint;
27        for (j = 0; j < rint * 100; j++)
28        {
29          dummy = 2.345 * 8.765 / 1.234;
30        }
```

This almost nearly eliminated any issues that I was seeing before, with a few exceptions that were very small discrepancies.

After looking at the code a bit more, I came to realize that the variable balance is shared between the two threads that we create while running the program.

To confirm this might be causing an issue, I used valgrind:

```
valgrind --tool=helgrind ./race -v
```

The threads accessing the variable without any protection appear to be the catalyst of the sporadic behavior I saw when executing the program. The behavior became less prevalent when I took the latency being introduced by all of the floating point calculations out of the equation, but it did not completely resolve the issue. In fact, this issue will persist until a form of protection is introduced into the program.

Problem 1b