For Prog06 I created locks for each square in the grid as well as locks for each ink tank. I thought this was preferable to simply locking an entire row or column, because more travellers means more crossing paths. One thing I thought was somewhat clever was my use of casting to lock the color before calling any functions. What would be 9 or 10 lines of code gets reduced to just 5. The instructions specifically say “a red traveller travelling 5 squares must acquire 5 units of ink.” I interpreted this to mean that a traveller must acquire *all* the ink it needs at once. Reading the report instructions I see I was incorrect, so I went back and quickly modified the code to acquire a single unit at once.

When my travellers require ink, they first lock their associated color using casting. Then there is an if statement that checks the color. When the color is determined, the program enters a body-less while loop that tries to acquire the required amount of ink. Only one traveller of a given color will ever be in the loop to check the ink level, all the rest have to wait for the lock.

In my original program, if the first traveller to access the ink needs 10 units, and there are only 9, it would loop until there is enough ink. No other traveller would be able to access those 9 units even if they only needed 1. On top of this, acquiring all the ink at once required my program to separate painting and acquisition. This meant that a traveller could not paint part f it’s path, it had to either paint all of it or wait for the ink. After the modifications, this is not the case. Each traveller gets a single unit of ink every time it paints a square. Because of this, every last drop of ink is used before requiring a refill.

The program is admittedly much slower with the updates. Locks and checks happen per square rather than per path. At the maximum distance, 19 times the work goes into acquiring ink. I felt that the point of the assignment was not the speed of the travelers but how efficiently they painted, so I switched to the current program.

Difficulties

There were not many difficulties in my program that were not simple errors on my part. For instance, I spent awhile trying to figure out why my program wouldn’t compile. It turns out I hadn’t added pthreads to the compile command, which took my quite a while to figure out. Another issue; after casting to unsigned char*, I was trying to access column 0 but would always end up accessing column 1. Turns out I was using indexes 1 - 4 rather than 0 - 3 and I was accessing the 4th index to change the color red. Because of this I was accessing the red color channel in the next column from the current column.

The last issue worth mentioning was with the random number generator. I moved it to the threads for efficiency, but I did not account for the fact that the seed was based on the current computer time. I also did not realize the built in generator was casting to unsigned int. I thought that was a designation for the generated value. Because of this I spent a while trying to figure out the best place for the generator. Once I realized my error I just added an index value to the traveller struct. When I make a thread; I make a traveller struct, add a reference to grid, add the index, and pass the struct to the thread. Once in the thread I make a random generator based off the index. That way the generator is not a shared resource, each generator is unique, and the work of randomly initializing a traveler is done in the thread rather than in InitializeApplication.

Limitations

I discussed how color acquisition is slightly slower than the original version but much more efficient at using all the available resources. Obviously, I did not institute any of the extra credit options, but my program completes every desired aspect for the base program in as concise a manner as I can hope. I would have liked to split the color acquisition and tile painting into separate functions, but I was not sure if this would cause adverse effects on the threads. I suppose this is a slight limitation, because I could have reduced my code by about 30 lines.