CS240 - Operating Systems
Assignment #5
Spring 2024

The purpose of this assignment is to implement an algorithm using threads in order to take advantage of parallelism in the calculations. We will be using the *pthreads* library for this assignment.

Matrix multiplication is a common mathematical calculation. For example, assume that we want to calculate a matrix $C = A \times B$, where A is a 3×5 matrix, and B is a 5×4 matrix. Note that matrix multiplication requires that the number of rows in the first matrix be equal to the number of columns in the second. The element $C_{1,3}$ in this example would be computed by:

$$C_{1,3} = A_{1,0} * B_{0,3} + A_{1,1} * B_{1,3} + A_{1,2} * B_{2,3} + A_{1,3} * B_{3,3} + A_{1,4} * B_{4,3}$$

The resulting matrix C would be a 3×4 matrix. To generalize the equation, if A is of size $i \times k$, an B is of size $k \times j$, the result C will be of size $i \times j$, where each element can be computed as:

$$C_{i,j} = \sum_{n=0}^{k-1} (A_{i,n} * B_{n,j})$$

While large matrices involve a fair amount of calculation, the calculation of each element is independent of the other elements, a situation called *embarrasingly parallel*. This means that perhaps we can speed up the operation by having separate threads working on smaller pieces of large matrices, thereby taking advantage of any parallelism in the system. Another feature of such a computation is that no synchronization is necessary between threads - we can compute the bottom half of a matrix without having to wait for the top half to complete.

For this assignment, you are to write a threads solution to this problem. Your solution should handle arrays up to size 50×50. The arrays A and B should be input from a file, where the first line in the file specifies the size of each matrix (the three values i, j, k as specified above). This should be followed by the matrix element values themselves.

The command line to run your program should include the file name, and then a number that specifies the number of threads to be used in the calculation. The matrices should be floating point, and made global so each thread has access to them.

Based on the command line input, your program should then determine for each thread which range of rows it should process, and pass these values as arguments to the thread. For example, for a 50 row matrix, if the command line specifies to distribute the matrix among 5 threads, the first thread should process rows $0 - 9$, the second thread rows $10 - 19$, etc.

As a small example, assume that a file named `matmulfile` has been created, that looks like:

```
9 3 5
1.2 4.2 0.4
4.4 0.9 5.3
5.1 0.9 6.1
4.2 5.9 1.2
6.6 3.1 2.3
4.1 5.6 7.1
4.3 0.6 8.2
5.0 0.7 8.5
5.1 3.5 9.9

4.6 3.0 1.5 7.8 4.1
1.1 3.3 8.1 9.0 0.1
3.4 5.7 5.9 9.1 4.3
```

Assume that you run your program, named `matmul`, with the following command:

```
matmul matmulfile 3
```

Your program should then create three threads, with the first thread processing rows $0-2$, the second thread rows $3-5$ and the third thread rows $6-8$ of the first matrix. These parameters should be passed as arguments to each thread.

NOTE: Our system has been limited to no more than 20 threads, so attempting more will cause an error.