Effective and Modern C++ Programming

Lab 12 – Threads

Exercise 1. Threads and mutex.

Program random.cpp fills an array with random values then replaces each element x with f(x). Finally it computes sum of elements in the array.

When generateArray is called in parallel by two threads the output changes. The order of elements can change but the sum should remain the same. Correct the algorithm to avoid data races.

Make the number of threads used a parameter of the algorithm.

Change computeSum function so that it could be run in parallel by several threads.

Exercise 2. Mandelbrot set

Program mandelbrot.cpp generates bitmap file with part of Mandelbrot set contained in the rectangle $[x_1, x_2]x[y_1, y_2]$ on the complex plane. Each pixel corresponds to some complex number c. It defines complex function $f(z) = z^2+c$. We generate sequence $z_{n+1} = f(z_n)$ starting from $z_0=0$. If sequence is bounded than c belongs to Mandelbrot set.

In practice we iterate f(z). if $|z_n|> M$ (where $M >2$) then sequence is unbounded and c is not in Mandelbrot set. We use first such n to choose color for corresponding point.

If we reach maximal number of iterations N and $|z_n| <= M$ for all $n<=N$ then we assume that c is in the Mandelbrot set.

Each point can be checked independently and it can be done in parallel.

Change the part of the program that generates bitmap from sequential to asynchronous to get maximal speed up. Find optimal number of threads for your processor.

Exercise 3. Futures and Promises

Write function that for a given file name computes the sum of ASCII codes of all characters in that file and returns it using std::promise.

In the main thread create separate thread for each file name given as command line argument. Thread should compute the sum of characters in a given file.

Use futures to get those sums and find all pairs of files with the same sums.