#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <iostream>

#include <iomanip>

#include <time.h>

using namespace std;

int num1 = 0;

int num2 = 0;

bool raceDetected = false;

unsigned int seed = 0x000fff;

const int NUM\_THREADS = 2;

const int MAX\_RUN\_COUNT = 50000000;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

bool finished[NUM\_THREADS];

timespec timeDifference(timespec start, timespec end)

{

timespec temp;

if ((end.tv\_nsec-start.tv\_nsec)<0) {

temp.tv\_sec = end.tv\_sec-start.tv\_sec-1;

temp.tv\_nsec = 1000000000+end.tv\_nsec-start.tv\_nsec;

} else {

temp.tv\_sec = end.tv\_sec-start.tv\_sec;

temp.tv\_nsec = end.tv\_nsec-start.tv\_nsec;

}

return temp;

}

void \* thread\_routine (void \*thread) {

int \*id\_ptr, thread\_num;

id\_ptr = (int \*) thread;

thread\_num = \*id\_ptr;

long counter = 0;

int tmp1;

int tmp2;

int r;

while(counter < MAX\_RUN\_COUNT && !raceDetected) {

pthread\_mutex\_lock (&mutex);

tmp1 = num1;

tmp2 = num2;

r = rand\_r ( &seed ) % 5;

num1 = tmp1 + r;

num2 = tmp2 - r;

counter++;

if(num1+num2 != 0 || raceDetected)

{

raceDetected = true;

cout << num1 << '\t' << num2 << '\t' << num1+num2 << endl;

cout << counter << " : " << (raceDetected ? "true" : "false") << " : " << thread\_num << endl;

raceDetected = true;

}

pthread\_mutex\_unlock(&mutex);

}

finished[thread\_num] = true;

}

int main() {

pthread\_t tid[NUM\_THREADS];

int\* threadIdNum[NUM\_THREADS];

int returnCode;

pthread\_attr\_t attr;

void \*status;

pthread\_attr\_init(&attr);

pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);

timespec beginTimer, stopTimer;

clock\_gettime(CLOCK\_MONOTONIC, &beginTimer);

for (int i=0; i<NUM\_THREADS; i++ ) {

threadIdNum[i] = new int;

\*threadIdNum[i] = i;

returnCode = pthread\_create ( &(tid[i]), &attr, thread\_routine, (void \*) threadIdNum[i] ); //create thread and output message if it fails

if (returnCode) {

cerr << "ERROR; return code from pthread\_join() is " << returnCode << endl;

exit(-1);

} else {

cout << "MAIN: completed thread creation of thread " << i << " successfully" << endl;

}

}

pthread\_attr\_destroy(&attr);

cout << "\nMAIN: threads running...\n\n";

for(int i=0; i<NUM\_THREADS; i++) {

returnCode = pthread\_join( tid[i], &status);

if (returnCode) {

cerr <<"ERROR; return code from pthread\_join() is " << returnCode << endl;

exit(-1);

} else {

cout << "MAIN: completed join with thread " << i << " having a status of " << (long)status << endl;

}

}

clock\_gettime(CLOCK\_MONOTONIC, &stopTimer);

cout << "MAIN: all threads completed\n\n";

cout << num1 << " + " << num2 << " = " << num1 + num2 << endl;

cout << "time: " << timeDifference(beginTimer, stopTimer).tv\_sec << "." << timeDifference(beginTimer, stopTimer).tv\_nsec << " seconds" << endl;

return 0;

}