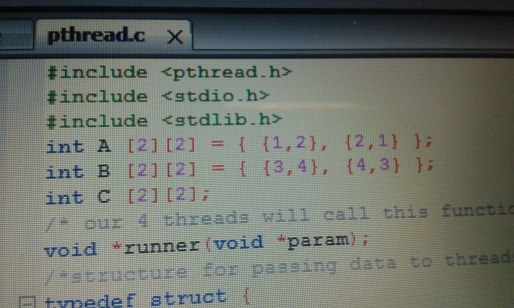
1) At all, thread is like a flow of control within a process, while multithreading contains a lot of threads within the same address space. In order to multiply matrices we create 4 child threads with pthread\_create() function call passing him thread identifier, attributes for the thread, name of the function (in our case runner()) and data as parameter. In function runner() our threads will begin execution performing formula for multiplication of matrices. Also we have our parent thread in main() which will wait its children to be executed and with their identifiers will return value and collect them by calling pthread\_join() function. Through function pthread\_ exit() threads will terminate. Also in order to calculate each element of our C matrix we should pass several parameters to each thread. For it we created data structure using a struct to pass the row and column where a thread will start validating.

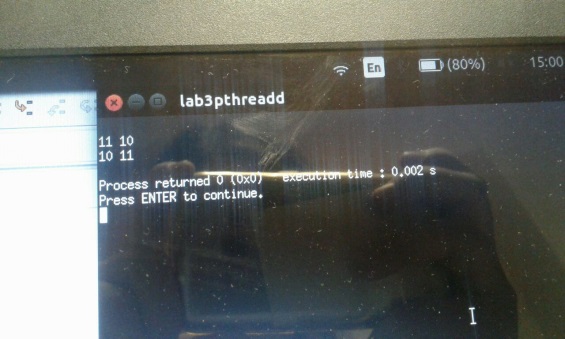
2)Logic of implementation this program using Java threads is almost same. Java API provides different features for creating threads. In our case we have class Threadlab3 which extends main class Thread and overrides its run() method, in which every thread will implement multiplication using known formula. Also in main we have start() which will initialize new thread in JVM and call run() method of our class. JVM will create parent thread which executes in main() method and its 4 children threads which will be created after start() will be invoked. They will be executed in run() method, and as soon as they exit from this method they will terminate. And at this time their parent will wait its children to be executed using join() method.

3)Program implemented using Windows thread library is also similar. In program we also define Multiplication() function which will be performed by each thread and which is passed a pointer to void (LPVOID). As in Pthreads we have CreateThread() function, to which passed: Null(default security attr), 0(default stack size), Multiplication(thread function), &data[tid](parameter to thread function), 0(default creation flags) and &ThreadId[tid](thread identifier). Similar to pthread\_join() we have WaitForMultipleObjects() function which will 4 threads to be executed. Parameters are 4(number of threads), ThreadHandle, True(determines whether all threads have been signaled) and INFINITE(timeout duration). Through function CloseHandle() thread handles will be closed.

In all cases I tested matrices with following elements:



and result was as:



which is true based on matrix multiplication rule.