Please fill in each section of this documentation file with the information which we will need to mark your coursework.

**Full Name:** Natalia Sharon

**University Username:** psyns3

**Computer Science Username:** nxs13u

We will try to compile your program by opening the project files, replacing the DoNotChangeThese.h file and recompiling everything.

**Are your projects in one Visual Studio Solution or multiple:**

One visual studio solution

**What are the project names for the different components:**

The part A Client:

Client (Please note, in VS the name is Client, however in my folders, it is called nxs13u\_OSC\_CW1. I hope this does not cause confusion, as when I attempted to rename the files then my program did not work as could not link files together, so I left them as it was)

The part B/C Server:

Server

The part C Server:

Client2

**Are there any special instructions we need in order to get your programs to run?**

As noted before, my part A client is called Client in VS however the folder in my project is called nxs13u\_OSC\_CW1 – the project builds and runs fine despite this.

**What problems do you currently know about with your programs?**

In the general comments section for each requirement, please make comments about what you did for that requirement, the names of the files which you changed and any special comments which you think we should know about.

**Requirement A1:**

General comments:

* Altered the Client class
* Created an array of 5 threads, each calling their corresponding procedure
* Error message are printed to console if there are problems creating the threads

**Requirement A2:**

General comments:

* Altered the Client class
* I used critical sections and mutexes to protect resources.
* These are all created and closed/released in the main
* I tested whether these worked by counting how many times each resource is used and therefore worked out what the count of each resource should be after running all threads and tests.
* I commented out all tests and worked on one test at a time by uncommenting it, working out what the thread count should be, and knowing my methods worked if the resource counts were the same as the ones I had worked out.

Please comment on the method that you used to protect your critical sections and why you believe that it was the most appropriate and/or efficient method:

* I was able to use critical sections for tests 1-10 by using a different critical section for each resource
* So I had 6 critical sections in total
* Each critical section protected its corresponding resource and so each resource could only be used at one time
* When two resources were being used, I protected them using their two critical sections.
* When I got to test 11, I found that this did not work as my output to console showed that threads would deadlock
* I decided to use mutexes where multiple resources started and ended in different orders in the tests. I used a different mutex for each corresponding resource, and the mutex would wait until all the resources were available before continuing to them.
* I chose to use a mutex over a semaphore despite being more familiar with semaphores as they were used in the labs, as a mutex is used when you only want one thread to execute code at a time, whereas a semaphore can be locked by threads concurrently.

**Requirement A3:**

General comments:

* Altered the Client class
* Created volatile resource counters for each resource and incremented them after each corresponding resource is used

Please comment on whether you needed to protect your counts from interference between threads and what method you used to do this if so.

* These counts were placed within the critical sections and mutexes so they could only be incremented when the resource was being used.

**Requirement A4:**

* Altered the Client class
* 7 different messages sent each time the function is called
* 6 messages containing resources counts sent to the window which handles resource count (1st message sending count for resource 1). Contain the resource count for that resource as well as its id – 1 (so it can be placed directly in its place in the array when it is passed through).
* 1 message containing thread id sent to window which handles client count.

Please comment on whether you used PostMessage or SendMessage and briefly say why.

* I used PostMessage to send my messages as Post is asynchronous and returns without waiting for the message to be processed, whereas Send is synchronous and does not return until the window procedure has processed the message. My client does not wait for a response from the server and so I used Post.

**Requirement B1:**

General comments:

* Altered the Server class
* Registered two window classes in the main
* Created two window procedures corresponding to the registered classes
* Error message are printed to console if there are problems creating the windows

**Requirement B2:**

General comments:

* Altered the Server class
* Created switch cases within each windows procedure to handle the messages
  + WM\_PAINT copied from the code given
  + WM\_USER + 1 to handle the messages and to extract parameters and place them in the array or increment array values accordingly
  + WM\_CLOSE to destroy the window
  + WM\_DESTROY to post a quit message
* For the threads, I passed through their ID in the 3rd parameter (same value as their place in the array) and incremented the place in the array that corresponded to their ID. I thought this was the most efficient way.
* For the resources, I passed through their value in the 3rd parameter as well as an ID in the 4th parameter so that their place in the array (corresponding to their ID) could easily be updated.

**Resource protection in the server:**

Please comment on whether you needed to protect your resources from interference. If so, please explain under what circumstances you need to protect them and why. If not, please explain why you believe that this is unnecessary.

* Altered the Server class
* I now needed to protect the resource counts in the Server as 5 processes were updating the same parts of the array which holds the resource counts. I used a critical section to protect passing the resource values through from the message from Client2 (using the socket) into the array where they are all stored.

**Requirement C1:**

General comments:

* Altered the Client2 class
* When processes are created, they call the main and so I parsed through a process ID as its second parameter so that when it entered the main, argc was increased (otherwise processes would keep calling main and keep creating more processes)
* Therefore, if argc was 1 then no processes has been created. So I set up an if statement where if argc > 1 then the appropriate procedure corresponding to the process id was called, else the processes haven’t been created and so will be created
* I placed the createProcess function in a loop to loop as many times as the PROCCOUNT variable defined at the beginning was worth
* Error message are printed to console if there are problems creating the processes
* Structs were created for shared data, as well as mapping files to allocate some shared memory and map it into its address space.

Please explain how you share the resource usage counts between the processes:

* Used mapping files and structs to share resources (the variable counts) between the processes to avoid interference
* Incrementing resource counts was done within the critical sections and mutexes
* After all the processes have finished processes, I printed the resource counts and they were correct

Please explain any changes that you had to make to the entry protocols for your critical sections in your clients:

**Requirement C2:**

General comments:

* Altered the Client2 class and the Server class
* In Client2
  + I Created a socket in the sendDataToServer function to connect to the server
  + The server address was that of the local machine
  + Connect to server and build messages, then send them
  + Check for errors and if they occur, the socket shuts down
  + When message has been sent, close socket
* In the Server
  + Created a thread at the end of main to call a function called Socket, which has the listening socket.
  + A thread in Socket is created which runs HandleConnection where all message values are extracted and passed into the corresponding arrays (this section is protected by a critical section).
  + I check for errors which prints the error to console
  + Windows are updated to fit the values in the now updated array

I had a problem throughout this part of the coursework. When running my server, It would close after telling me that there was a Bind Error 10048. Running my code on other laptops would not result in this error, and so I had to wait until coming back to university to complete this coursework a few days before the deadline.

Please explain the structure of your messages that you pass from the clients to the server and of any messages which are passed back again.

* For each SendDataToServer call, I send one message through to the server. The message is composed of 7 parts
* 1 part being the processID and the other 6 being the resource counts
* They are in the format of message1.a.resource7msg = pMyData -> resource7. pMyData points to resource7 which holds the count of resource 7, and this value is stored in resource7msg in the message message1.a

Please explain any additional critical sections which you now need (if any) and how they have been protected:

* Altered the Server Class
  + I now needed to protect the resource counts in the Server as 5 processes were updating the same parts of the array which holds the resource counts. I used a critical section to protect passing the resource values through from the message from Client2 (using the socket) into the array where they are all stored.
* Altered the Client Class
  + Used structures and mapping files to share the resource counters amongst the processes to avoid interference