Logo, company name

Description automatically generated

**Faculty of Engineering and Applied Science**

**SOFE 3950U / CSCI 3020U: Operating Systems**

**Tutorial 4**

**Group Member 1**

**Name: Alexander Campbell**

**Student ID:  100703650**

**Group Member 2**

**Name: Atharshan Kennedy**

**Student ID: 100590243**

**Group Member 3**

**Name: Joey Villafuerte**

**Student ID: 100759003**

**Date: Mar 13, 2022**

# **Conceptual Questions**

1. - pthread\_create() is the function that creates the new thread.

- pthread\_join() will wait for the thread that’s specified in the thread argument to terminate before continuing execution of the program.

- pthread\_exit will terminate the current thread that is running and return it with a return

val specified by the retval argument.

1. Processes do not generally share the same memory and to do so requires additional memory schemes. Threads can more easily share the same memory as they are given the same memory state as the process, unless specified otherwise.
2. Threads consume a lot less resources than processes, this makes multithreading a much better choice for most applications. Another difference is the ease with which there is inter-thread or inter-process communication. Threads are much easier to communicate with each other, while processes are much more difficult and take system calls in order to do so. Processes have their own memory space, while threads share memory unless otherwise specified. This makes it so heavyweight; memory intensive processes are generally better in multiprocessing, and it is also an advantage when you want separate memory.
3. Mutual exclusion is a technique in multithreading that is used to prevent race conditions between threads, meaning that one thread may access a resource before it’s ready and cause undefined behaviors. This means that it cannot access a critical section before it’s ready. The critical section is a section of the code that is a shared resource between threads or processes.
4. - pthread\_cond\_wait will make a thread wait until a certain condition is met

- pthread\_cond\_timedwait will make a thread wait for a specified time

- pthread\_rwlock\_init will lock a pthread until the thread is unlocked by another thread or process

- pthread\_mutex\_init will create a mutex, or a mutually exclusive flag, which is an object that allows threads to take turns accessing a certain resource

All of these functions can be used to control what the thread is working on and ensure that it will not access a critical section.

# **Code Explanation**

## Graphical user interface, text, application Description automatically generatedQ1:

The first program creates two threads one after another with one exit before the next thread. The first thread will print ‘hello world’ then next thread will print ‘goodbye’. Th sleep random from 0 to 2 sec is used to change the order of the which will print first which is done randomly for both prints.

## Text Description automatically generatedQ2:

Th second program will take 5 grade inputs. The grades are first stored into an array that use the struct called grades as it’s framework. Afterwards the 5 threads are created one after another with the second for loop. Within the thread function a counter before bell curved grade is printed and then changes. Then the bell curved grade is printed before the thread exits. Each grade is multiplied by 1.5 before being printed. The counter is used to help list the grades. And sleep of two secs is used before thread creation to make sure the previous thread prints and exit before the next thread.

## Graphical user interface, text, application Description automatically generatedQ3:

The third question takes 5 names, ID, grades as input and saves the data to an array that use the struct called student as its framework. Afterwards the screen will clear and will start creating 5 threads with the second for loop. Sleep with two seconds is used again to make sure that the previous thread is finished and terminated before the next thread. And each thread will print out the name of the student followed by the student ID which is then followed by the grade before bell curve and grade after bell curve, where 1.5 is multiplied to the current grade input that is to be outputted.

## Text Description automatically generatedQ4:

The fourth question will take 10 grades as input. The grades are stored in an array that uses the struct Grade as its framework. Once that is done the second for loop will create 10 threads with each thread adding the grade to the total grade global variable. Here, in order prevent two or more threads from accessing the same global variable, Mutex is used to lock other threads out while still allowing one thread to go in. After the total grade has been calculated, it is finally printed out.

## Text Description automatically generatedQ5:

In the final program a thread barrier is first initiated before the while loop. This will be used mainly with a thread function read\_grades(). Then a file called grade.txt is opened. The file contains 10 grades that must be read line by line. After a line is extracted with the while loop fgets it is sent to read\_grades() thread function. In read\_grades() the grade value is saved into a global array that uses the Grade struct as it’s framework. After each thread saves a grade value to the global array it waits at the barrier in read\_grades() until 10 threads as arrived. After 10 threads as arrived the barrier opens allowing for all threads to flow out and exit synchronously. After the while loop has finished the file is closed and the barrier is destroyed.

Graphical user interface, text, application, email

Description automatically generated

The next part of question 5 will create a file and write to bellcurve.txt. The for loop will be used with save\_bellcurve() thread function when creating the threads. Each grade from the global array is sent to this thread function to be added to both total\_grade and total\_bullcurve global variables. As a thread comes in it locks others out until it has finished with Mutex lock. Going back to the first for loop in int mian(), after a creation of thread is done the new bell curved grade is saved to the bellcurve.txt file. After for loop has finished the file bellcurver.txt is closed. Finally, the Total grade and Avg of before and after the bell curve is outputted.

## Make File:

Graphical user interface, text, email

Description automatically generated

Make file used for compiling questions 1 to 5.

# **Sample Runs**

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedGraphical user interface, text, chat or text message

Description automatically generatedText

Description automatically generated