Øving 2

1. Explain the difference between a process and a thread.

A process refers to a program currently under execution. A process essentially allows you to execute all the important parts of a program. A thread on the other hand refers to the smallest segment of code that can be handled independently. A process may consist of multiple threads.

2. Describe a scenario where it is desirable to:
    a. Write a program that uses threads for parallel processing

A scenario in which it is desirable to write a program that uses threads for parallel processing could be where we need multiple parts of a process to execute simultaneously whilst also sharing the same segment of memory. This could for example be used in videogames.

   b. Write a program that uses processes for parallel processing

A scenario in which it is desirable to write a program that uses processes for parallel processing could be where we need to handle large datasets. By using processes for parallel processing we can run one process over multiple cores simultaneously resulting in faster execution times. This is useful in, for example, weather forecasting.

3. Explain why each thread requires a thread control block (TCB)

Each tread requires a TCB in order to keep track of information concerning the thread such as the thread id and state.

4. What is the difference between cooperative (voluntary) threading and pre-emptive (involuntary) threading? Briefly describe the steps necessary for a context switch for each case.

The difference between cooperative threading and pre-emptive threading is that the cooperative threading has exclusive access to the CPU until it gives it up, whilst preemptive threading also can give up access, however, when they don't, access can be taken from them. The scheduler will then start the next thread.

1. Which part of the code (e.g., the task) is executed when a thread runs? Identify the function and describe briefly what it does.

The part of the code that is executed when a thread runs is the go function.The method prints out which thread it is on.

2. Why does the order of the “Hello from thread X” messages change each time you run the program?

The order of the message changes as the threads finish in different orders between each time the program runs. It is possible for the output to be the same for two consecutive runs, this is however very unlikely.

3. What is the minimum and maximum number of threads that could exist when thread 8 prints “Hello”?

The maximum amount of threads that can exist when thread 8 prints “hello” are 11, whilst the minimum amount of threads that can exist is 2

4. Explain the use of pthread join function call.

The pthread_join methods use is as a wait function that waits on the given thread to finish running.

5. What would happen if the function go is changed to behave like this

With the altered code the program would wait an additional 2 seconds before printing for the 5th thread. After the 2 second delay “Thread 5 returned with 105”

6. When pthread join returns for thread X, in what state is thread X?

The thread could either be finished running or notifying the other threads that it is finished.