Mandatory Assignment 3

Group 14

a) What is the idea of the application?

The idea of the project is to make a document analyser tool.

The program will take a document and load its content to memory. Then a variety of analysis will be taken, such as word count, character count, line count and the like. The output of all the different analyses will be written to an output file at the end.

b) Why is concurrency needed?

In computing concurrency is needed to avoid starvation of resources which can lead to problems like deadlocks.

We use concurrency in our project when analyzing the input text. Here a thread for each analysis is created to execute the given function which analyzes the input text.

Concurrency has the ability to speed up the process of analyzing the input text, if the input text is of a certain magnitude.

c) What could be the potential issues specifical?

There are two major issues that could happen.

If we have one or more threads being locked out of accessing a resource, because another thread is using said resource, we have a deadlock on our hands. In our case this can happen when we are writing the results of our analysis to the output file.

The other issue could be race conditions. Which is where any number of threads racing amongst themselves have the possibility of causing unexpected side effects.

A problem could occur when 2 processes write to the same variable at the same time. Each process would then read the value of the variable as it was before any of the two made any changes, resulting in the latter finishing process undoing the work of the first one.

In our program this could happen in the end when trying to write to the output file. One thread could have written something to the file and then the next could possibly override the already written content.

d) Address race conditions

When several threads get access to the same data or are using a shared variable, then race conditions can happen.

As mentioned earlier this is a possibility when we are trying to write the results of each of the threads of the analysis to a file at the end.

e) Solution for race conditions

Mutual exclusion is a way to avoid race conditions. The way that mutual exclusion works is when a thread is using shared data only said thread has access to it. And the other threads will have to wait for the first thread to finish, before any other thread can use the shared data.

Our solution to the race conditions is using semaphores to handle access to the shared data, in this program that will be the output file.

A simple analogy would be when shopping and two people want to try on clothes. The changing room has a sign that indicates if the changing room is available or in use. Now one of the two people, person A, enters the changing room because the sign says it's available. The sign changes to occupied and the other person, person B, will have to wait for person A to finish. When person A exits, the sign changes to available and person B can enter.

f) Address deadlocks and starvation

In situations where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process a deadlock occurs.

When high priority processes keep executing and low priority processes get blocked for an indefinite period of time, time starvation is the occurring problem.

g) Solution for deadlocks and starvation

As our program doesn’t contain any chance of a deadlock, we will provide non-specific solutions.

One solution for deadlock and starvation is to use Banker’s algorithm. Banker’s algorithm is a resource allocation and deadlock avoidance algorithm. It simply checks if a system has the capacity to allocate a resource, based on the resources activities, to ensure that the system remains operational at all times.