Critical Section Problem | Operating System – M03 P04

This is a multipart blog article series, and in this series I am going to explain you the concepts of operating system. This article series is divided into multiple modules and this is the third module which consists of 10 articles.

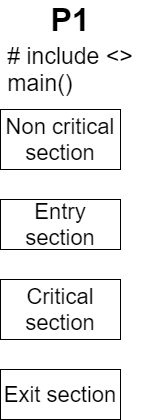
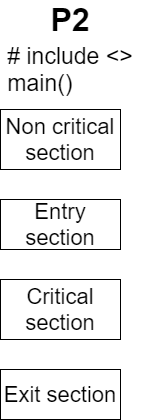
In this article we are going to discuss the critical section problem. First we will see that what is critical section, then what are the problems related to critical section and at last we will see some basic solutions to overcome that problem.

**Critical section**

* It is the part of the program where shared resources are accessed by various processes.
* Here, the various processes means “cooperative processes” i.e. which share something or which have something in common, it could be code, data, resources, buffer etc.
* We can also say that critical section is a place where shared variable or resources are placed.

Now, when we have seen that what is critical section then let’s understand about critical section in detail.

* Suppose we have two C program, some part of both program will be common and will be placed in the critical section.
* Where, some part of both the program is not common so it will get placed in “non-critical section”.
* If both the program will starts executing the code present in the critical section, then a problem will occur and this problem is known as “race condition.”
* **Note:** *Code present in critical section can be executed by only one program at a time.*
* To avoid this problem (race condition) we have to synchronize both programs.
* One way to do that is by writing a “entry section” in both the programs.
* First the program has to execute the code written in entry section and if it clears then only it can execute the code of the critical section.
* And if one program has cleared the entry section code and is present in critical section then the other program will get blocked at entry section (means it cannot execute the code of the entry section, until first program comes out of critical section).
* One more thing, when the program comes out of critical section it has to execute “exit section” code.

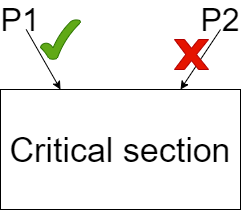
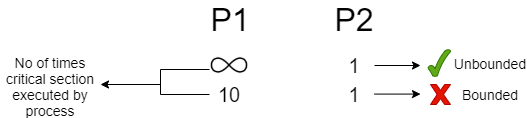
* We can synchronize it by using any predefined method, or we can write our own program to do so.

To achieve synchronization mechanism we need to fulfil these four conditions.

1. Mutual exclusion
2. Progress
3. Bounded wait
4. No assumption related to hardware or speed.

“Mutual exclusion” and “progress” are primary (mandatory) rules, while “bounded wait” and “no assumption related to hardware or speed” are secondary rule.

Primary rules have to be followed at any cost, if secondary rules are not possible to follow its okay, but we have to try to follow all four rules.

1. Mutual exclusion:
   * Suppose if P1 wants to execute critical section and start executing it, and then at the same time (while P1 was executing critical section) P2 wants to enter critical section in that case P2 will be blocked.
   * And by that we will achieve “mutual exclusion”.
   * 
2. Progress
   * Suppose there is some common code written in critical section and at present critical section is empty (means that there is no process in it).
   * So, now P1 wants to enter critical section and P2 is blocking it do so. Maybe because of some code written in the enter section of P2.
   * Therefore, in this case P2 is not going in critical section and also blocking P1 to enter critical section.
   * This means that progress is not there and this should not happen.
3. Bounded wait
   * If P1 is executing the critical section n times, due to which P2 is not getting chance to execute critical section.
   * Then this is a problematic situation and this should not happen.
   * There should be some time limit for every process to execute critical section in order to give chance to other process to execute critical section.
   * 
4. No assumption related to hardware or speed:
   * Suppose we have a solution to achieve synchronization which says that this solution will run on 32-bit system but won’t work on 64-bit system or if we say that “this solution will run if the processor is above 1GHz, and won’t work for the processor below that”. So, this type of solution is not valid.
   * The solution should not be dependent on the hardware or the speed of system, and they should be portable across different operating system.

So this was all about critical section problem. Hope you liked it and learned something new from it.

If you have any doubt, question, queries related to this topic or just want to share something with me then please feel free to contact me.