Process Synchronization |Operating System – M03 P01

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 will the basic concepts of process synchronization. We will see an example and try to understand the basic of process synchronization and how it is achieved.

**Process synchronization:** There are two types of process

1. Cooperative process:
   * In this kind of process execution of one process will cause some changes on other process.
   * This happens because they may have something in common or share something; it could be variable, memory or buffer, code, resources (CPU, printer, and scanner).
2. Independent process:
   * In this kind of process execution of one process does not affect any other process, because they do not share anything or have anything in common.

Let’s see an example and try to understand the concept the process synchronization. Here both the programs share a variable named “shared” and in this case we are talking about uni-processor system.

Int shared = 5;

|  |  |
| --- | --- |
| P1 | P2 |
| Int x = shared;  X++;  Sleep(1);  Shared = x; | Int y = shared;  y--;  sleep(1);  shared = y; |

* As mentioned earlier we are talking about uniprocessor system.
* We will execute P1 program first, you can also execute P2 first.
* When we execute the P1 program the value of shared get sorted/copied into a variable “x”.
* Then value of x is incremented by one and it becomes 6 from 5.
* For 1 second P1 process will be stop executing (due to sleep(1);) command, when P1 get paused CPU will not pause rather CPU will start executing P2 (CPU will perform context switching by saving the details of P1 in PCB which stands for process control block.)
* Context switching is possible in this case because there is a process in parallel.
* Now, P2 will start executing and value of shared variable which is 5 gets stored in a variable y.
* Now the value of “y” will become 4 after executing the command “y--”.
* After that P2 will also get sleep for 1 second because of the same reason that was present in the P1 program and is “sleep(1)”
* Now, CPU will start executing P1 from we’re it was paused.
* The value of x variable is 6 so according to the code we will load the value in shared variable.
* Now the value in shared variable is 6. And the program P1 will get terminate.
* Now again P2 will start executing from the sate it was paused.
* Here the value of y is 4 and the same will get load in the shared variable.
* Now the value of shared variable is 4.
* But this is wrong answer because we are adding 1 in a value and subtracting 1 from that so the value of variable should not change but here the value of variable is changing.
* This is because both the processes are sharing something so they are cooperative processes but they are not synchronized if they were synchronized then this problem will not occur.
* This problem is known as “Race condition”, and that’s why we need process synchronization.

So, in conclusion we can say that whenever two or more cooperative process run then a problem will occur known as race condition.

This was all about process synchronization. Hope you liked it and learned something new from it.

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