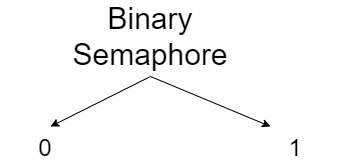
Binary Semaphore | Shell Scripting

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 see that what is binary semaphore and how does it work, in order to achieve synchronization and remove race condition.

**Binary condition**

* The counting semaphore has a range of – infinity to + infinity but in binary semaphore there are only two values 0 and 1.
* Binary semaphores are used more in systems then counting semaphore.



**Entry section code**

Down(Semaphore S)

{

If (Svalue = 1)

{

Svalue = 0;

}

Else

{

Block this process and place in suspend list, Sleep();

}

}

**Exit section pseudo-code**

Up(Semaphore S)

{

If(Suspend list is empty)

{

Svalue = 1;

}

Else

{

Select a process from suspend list and wake up();

}

{

* If the value of S is 1 and then it become 0 after performing Down operation, then it is considered as successful operation.
* If the value of S is 0 and then process execute Down operation, then in that case the process will be block/sleep and it will be sent to suspended list, it will be considered as unsuccessful operation.
* The Up operation first check that if the suspended list is empty or not, and if it is empty then it will change the value of S to 1, does not matter that what was the value the value of S, if the value of S previously was 0 then it will become 1 and if it 1 then it will remain 1.
* If the suspended list is not empty then “else” part will be executed and the process in the block/sleep state will get wake up and then get a chance to try to go in critical section by executing the Down operation.

So, this was all about binary semaphore. Hope you liked it and learned something new from it.

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