starve-free

kranthikumarrambha

March 2021

1 Working and Correctness of Algorithm

Explanation on how it works The starve-free solution works on this method : Any number of readers can simultaneously read the data. A writer will make its presence known once it has started waiting by setting wait to true.

Once a writer has come, no new process that comes after it can start reading. This is ensured by the writer acquiring the in semaphore and not leaving it until the end of its process. This ensures that any new process that comes after this (be it reader or writer) will be queued up on in. Now, from the looks of it, it might seem like this is a method that is biased towards writer and may lead to starvation of readers. However, this is where the FIFO nature of semaphore comes into picture.

Suppose processes come as RRRWRWRRR. Now, by our method the first three readers will first read. Then, when the next process which is a writer takes the in semaphore, it won't give it up until it's done. Now, suppose by the time this writer is done writing, the next writer has already arrived. However, it will be queued up on the in semaphore. Thus, it won't get to start writing before the process before it, the reader process has completed. Thus, the reader process will get done and then the writer process will again block any more processes from starting and so on.

To explain the writer process further, in case there's no other reader, it doesn't wait at all. Otherwise, it sets wait to true and waits on the writer semaphore. Also, to make the writer semaphore synchronizable across both the process where one process only executes wait() and other only executes signal(), we initialize it to 0.

To explain the reader process further, it firsts increments the var-start then reads the data and then increments the var-completed. After this, it checks both these variables are equal. If they are and a writer is waiting (found from wait), it signals the writer semaphore and finishes.

Thus, it will be pretty clear how we manage to make a starve-free solution for the readers-writers problem with a FIFO semaphore. We can say that all processes will be handled in a FIFO manner. Readers will allow other readers in the queue to start reading with them but writers will block all other processes waiting in the queue from executing before it finishes. In this way, we can

implement a reader-writer solution where no process will have to indefinitely wait leading to starvation.

The main idea here is that a Writer indicates to Readers its necessity to access the working area. At the same time no new Readers can start working. Every Reader leaving the working area checks if there is a Writer waiting and the last leaving Reader signals Writer that it is safe to proceed now. Upon completing access to the working area Writer signals waiting Readers that it finished allowing them to access the working area again.