REPORT

In this assignment ,I have implemented the concept of futures in Xinu Operating System over the producer consumer model.

Future is used for synchronizing a set of threads enabling them to perform concurrently. For instance in the producer consumer model, both of them are accessing one variable. The producer thread produces the data which is then consumed by the consumer thread. But if consumer thread is called before producer thread , then the consumer will not have any data to consume and hence it will exit without performing it's operations. So, the correct way for this model to run will be for the consumer thread to wait until the producer thread has set the value and then resume it's functionality. In order to implement the above idea, I use future.

I have developed three modes in future. Using these modes , I have performed synchronization for different cases of thread execution.

FUTURE_EXCLUSIVE:-
In this mode, we have only one consumer thread and on producer thread. If the producer thread runs first , then it sets the value first . So when the consumer thread runs , it can get the data from future. If the consumer thread runs first , then it changes it's state to waiting and suspends itself waiting for the producer thread to execute. When the producer thread runs, it set the value and resumes the consumer thread to get the value.

FUTURE_SHARED:-
This mode is used when we have a case where there is one producer thread but more than one consumer thread. In this case, I have implemented a queue using single linked list. Multiple consumers are enqueud so that , they get the data set by the producer in FIFO format. So, once the producer sets the data, we dequeue the consumer threads one by one to access the data from future.

FUTURE_QUEUE:-
This mode is used for multiple consumer and producer threads. For this case, two queues are required to handle consumer threads and producer threads. The two queues are named as get_queue and set_queue. get_queue is used by consumer while set_queue is used by the producer. We save the pids of threads in their respective queues.
Case 1:- If a producer thread executes, then, it checks if if the get_queue has any consumer threads suspended. If it does, then it sets that value,changes the state to READY and resumes the consumer thread. If there is no consumer thread in the get_queue , then the producer thread enqueues itself in the set_queue and suspends itself.

Case 2:- If a consumer thread executes, then , it checks if the set_queue has any producer thread waiting. If it does then, the consumer dequeues the producer thread pid and resumes its execution.That producer thread sets the value , changes the thread state to READY and exits giving the control back to the consumer thread which gets the value from that particular producer thread and changes the state to EMPTY .If there is no producer thread in the set_queue, that means there are no producer thread waiting to set the value. Hence, the consumer enqueues itself in the get_queue and changes it's state to WAITING.