CS : 431

Programming Language Lab

Report

Assignment - 1

Concurrent Programming

Group-2

Abhishek Goyal : 150101002

Roopansh Bansal : 150101053

❖ Question 1

➢ **The role of concurrency and synchronization in the given system.**

Concurrency:

There are multiple robotic arms available in the system and they are executing concurrently to pick a sock from the heap and pass it to the matching machine. Along with that, the sock matcher and shelf manager are working simultaneously with the robots arms.

Synchronization:

The heap of socks can be accessed simultaneously by all the available robotic arms but no two robotic arms are picking the same sock. Hence picking up of a sock by the robotic arms is being synchronized. Matching of the socks present in the matching machine's buffer and addition of a new sock in the matching machine's buffer is done synchronously. Arrangement of the pair of socks from the shelf manager's buffer to the correct shelf and addition of a new sock to the shelf manager's buffer is done synchronously.

➢ **How you handled it?**

Concurrency:

Concurrency is achieved by multithreading. A thread is created for each robotic arm and are executed concurrently.

Synchronization:

Synchronization for accessing the socks by the multiple arms is done by using **semaphores**. A semaphore controls access to a shared resource through the use of a counter. Here socks are the shared resources and we have used individual semaphores to lock each of the sock and set the semaphore counter value equal to 1 i.e. each sock can be picked up by only one robotic arm. We used block synchronization method to put a sock in the matching machine's buffer and similarly to put a sock in the shelf manager's buffer.

❖ Question 2

➢ **Why concurrency is important here?**

TA1,TA2 and CC are independent of each other and they can simultaneously update the marks of the students. In absence of concurrency, while one of them is updating the marks of any one student, others have to wait for even if they want to update the marks of some other student which can be done independently. So concurrency is required for allowing them to concurrently update the marks of different students simultaneously to speed up the complete process.

➢ **What are the shared resources?**

The files containing the student marks are the shared resources here. The marks of the individual students is a shared resource which needs to be handled with care.

➢ **What may happen if synchronization is not taken care of? Give examples.**

If synchronization is not taken care of, some updation of marks may not happen because of independent threads for TA's and CC.

Example:

X's initial marks = 10

Query-1: TA1 increasing X's marks by 10

Query-2: TA2 increasing X's marks by 20

If both Queries are executed simultaneously then both TA1 will see the initial marks of X's to be 10 and increase it by 10. Now TA2 will also see the initial marks of X's to be 10 as TA1 haven't yet written back the

updated marks. And TA2 increments X's marks by 20. Both TA1 and TA2 write back their updated marks for the student.

Finally X's marks in the records will be 30 (TA2 writes back the updated marks after TA1) , but actually it should be 40. So the Query-1 execution haven't affected the X's marks.

➢ **How you handled concurrency and synchronization?**

Concurrency:

Concurrency is achieved by multithreading. Individual threads are created for TA1,TA2 and CC and are being executed concurrently for updating the marks of the students.

Synchronization:

We used **block synchronization** on accessing the student's record, so that at a time only one teacher either of TA1, TA2 or CC can access the student's record.

❖ Question 3

➢ **What role concurrency plays here?**

Multiple customers can connect to the server simultaneously. Multiple customers can place the order at same time.Multiple items in single order can be processed concurrently to calculate the total processing time of the order.

➢ **Do we need to bother about synchronization? Why? Illustrate with example.**

Yes,

As multiple orders can be placed together. To ensure FCFS manner, all the items of an order should be processed either before or after all the corresponding items of other orders and for that there should be synchronization among orders while adding items to the processing queue of that item.

Example :

Two orders for one tea and one coffee each are placed simultaneously. After adding the items in the corresponding item's processing queue, in absence of synchronization one possible scenario for processing queue of tea and coffee is

1 Tea(customer-1)	1 Tea(customer-2)	....	

1 Coffee(customer-2)	1 Coffee(customer-1)	....	

So as we can see tea for customer-1 is processed before the tea for customer-2 but coffee of customer-1 is processed after coffee of customer-2, hence FCFS manner isn't ensured.

➢ **How you handled both (Concurrency and Synchronization) ?**

Concurrency :

We used multithreading to receive orders from multiple customers simultaneously. Similarly we used multithreading to process multiple items of an order concurrently.

Synchronization:

Multiple orders can be placed for the same item simultaneously. To add those items in the server's queue of that item synchronously we used **reentrantlock**. One has to acquire reentrantlock before adding that item into server's queue of that item.

To calculate delivery time of an order we used **countdownlatch** to make sure that delivery time of an order is calculated after processing time of each item in that order is calculated.