9/13/2018

Programming Lab

Assignment 1

Concurrent Programming

Piyush Jain 150101046
Shivam Gupta 150101068

Problem 1 – Sock Matching

1. The role of concurrency and synchronization in the above system:

- **There are multiple robotic arms working simultaneously, so we need some concurrency.**
- **We need some synchronization here to ensure more than one robotic arms do not pick up that same sock simultaneously.**

2. How we handled it?

- **For each robotic arm, we created a thread. Each thread will pick a sock and pass to the matching machine for matching with same-coloured pair and then passing it to shelf. For this, Thread class of JAVA has been extended in our class.**
- **We handled synchronization on socks by using synchronized keyword**

Problem 2 – Data Modification in Distributed System

1. **Why concurrency is important here?**

   To ensure that CC, TA1 and TA2, all three of them can work simultaneously.

2. **What are the shared resources?**

   - **Records in the file Stud_Info.txt**
   - **Files**

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

   - **One record may be accessed and updated simultaneously by a TA and CC since they both can access it concurrently. This may give incorrect final marks.**
   - **Files may be accessed and updated simultaneously by multiple threads leading to incorrect marks.**

4. **How you handled concurrency and synchronization?**

   - **We created different threads for each of TA1, TA2 and CC, thus they are working concurrently. This is done by extending Thread class of Java in our class.**

- **For synchronization, we have created locks:**
  - **concurrent.locks for the record level locks**
  - **normal semaphore locks (self-implemented) for the file level lock**

PROBLEM 3 – ROOM DELIVERY SERVICE OF TEA/SNACKS

1. What role concurrency plays here?

There can be multiple customers placing orders simultaneously. So, system should be able to record all the orders concurrently.

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

Yes, since there are multiple threads placing orders simultaneously, we need to ensure synchronization on items before showing them the availability. For example, say there are 10 cookies, and 1 order thread asks for 5 cookies and another asks for 6 cookies and both these threads reach simultaneously, had there not been synchronization, the system would show that the cookies are available to each of them, although it is not. We can serve only one of them.

3. How we handled both?

For concurrency, we did multithreading in our network implementation of socket. We created a new order thread for every customer and sent it to the canteen server. This is done by implementing *run* method of *Runnable* interface.

For synchronization, we used a FIFO queue on the received orders so that the availability status checking and updation is done by only one thread at a time. Thus, each new order will see only the latest updated status exclusively.