CS 4323 Design and Implementation of Operating Systems I

Final Group Project: Full Marks 100
(Due Date: 11/30/2021, 11:59 PM CDT)

This is the final group project assignment that must be done in the allocated group and using C programming language. If you do not know the group members, then please refer the module **FinalGroupProject** under **Home** page in Canvas.

This project is designed to meet the following objectives:

- Familiarization of race condition and critical section,
- Understand the need for synchronization,
- Understand and able to use one or more synchronization primitives,
- Familiarization with deadlock and ways to deal with it.

Problem description:

Consider XYZ company wants to build a software for online shopping and they have hired your team for this job. You are required to consider a multi-server network scenario i.e. the company has N number of servers, where each server can support M_i number of clients (or users), where i = 1, 2, …, N. The number of customers supported by each server can be different.

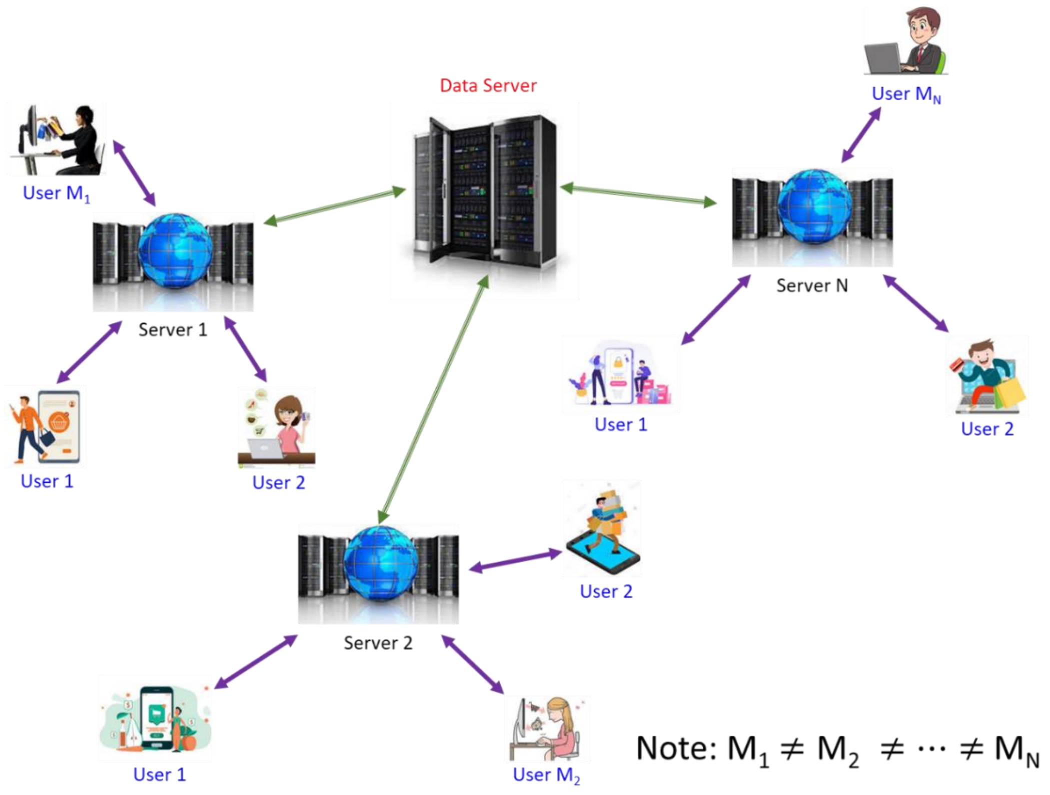

Fig. 1: Conceptual high-level diagram of multi-server network

Since this software is for online shopping, it needs to provide a support to handle two types of clients:

- consumer who buys the product, and
- seller who lists the product.

So, a term “client” can refer to either buyer or a seller.

All the N servers can share data among each other, as shown in fig (1). For example, a seller can list his/her product by logging on to Server “i” and it can be purchased from any other server, including server “i”. Similarly, a buyer can buy a product from one server and all other servers can see the change in the availability of the product.

The software needs to maintain the following five .txt files (given in blue color text, which we will call database, hereafter) to store several information regarding the buyers, sellers and the products:

1. sellerInformation database to store sellers’ information (**Seller ID**, Seller Name, Contact Number, Seller’s Contact Address)
    - Seller ID: ID number of a seller,
    - Seller Name: first and last name of the seller,
    - Contact Number: 10-digits contact number of the seller in the form XXX-XXX-XXXX,
    - Seller’s Contact Address: contact address of the seller
2. customerInformation database to store customers’ information (**Customer ID**, Customers Name, Contact Number, Customer’s Contact Address)
    - Customer ID: ID number of a customer,
    - Customers Name: first and last name of the customer,
    - Contact Number: 10-digits contact number of the seller in the form XXX-XXX-XXXX,
    - Customer’s Contact Address: contact address of the customer
3. productInformation database contains information about the product (**Product ID**, Product Description, Seller ID, Product Quantity Available, Product Price)
    - Product ID: ID number of a product,
    - Product Description: name of the product,
    - Seller ID: ID of the seller who is selling the product,
    - Product Quantity Available: number of quantity available,
    - Product Price: price per unit
4. billingInformation database contains information for the billing purpose (**Order ID**, Customer ID, Customer’s Billing Address, Total Order Price)
    - Order ID: ID number of the order,
    - Customer ID: ID of the customer who made the order,
    - Customer’s Billing Address: Same as customer’s contact address from customerInformation database
    - Total Order Price: total order price i.e. summation of all the items that customer has purchased on that particular order
5. customerOrder database contains information for the billing purpose (**Order ID**, **Product ID**, Quantity Purchased, Delivery Address, Total Price)
    - Order ID: ID number of the order,
    - Product ID: ID number of the product,

- Quantity Purchased: number of items (belonging to that Product ID) bought,
- Delivery Address: Same as customer's contact address from customerInformation database
- Total Price: total price of the product i.e. the product of each product price times the quantity purchased

Note: On each database, the column which is bold is unique for that database. For example,

- In sellerInformation database, **Seller ID** is unique in each row.
- In customerInformation database, **Customer ID** is unique in each row.
- …
- In customerOrder database, together **Order ID** and **Product ID** is unique in each row.

All the clients need to be registered first to use any service.

- The sellers need to register his/her information in the sellerInformation database and his/her product information in the productInformation database
- The buyers need to register his/her information in the customerInformation database

The software needs to handle the following type of operations:

- allow the clients to register in the system,
- allow the clients to update the information in the system:
  - both buyers and sellers can change their personal information
  - sellers can add a new product or remove a product
  - sellers can add or subtract the quantity of the product already existing in the database, including the price of the product
  - buyers can purchase the product or return the product
- allow the clients to see all the relevant information from the productInformation, billingInformation and customerOrder database.
  - sellers can look at the quantity of their products available and its price.
  - buyers can look at the quantity of their product available and its price.
  - sellers can look at all the customers' order belonging to his products.
  - buyers can look at their orders.
  - buyers can look at their billing information.

Program requirements:

The program should support:

- $N \geq 3$
- $M_i > 3 \ \forall i$

You will need to use semaphores as a synchronization mechanism to eliminate any race conditions. Also, you need to make sure there is no deadlocks in the program.

You need to include the following programs:

1. Program(s) that illustrates all the race conditions that can occur in the design. You can have multiple programs, each illustrating race condition for different scenario.
2. A single program that eliminates all the race conditions that have occurred in (1) above.
3. Illustration of deadlock scenario that can occur in the design. You can have multiple programs to illustrates deadlock for different scenarios.
4. A single program that eliminates all the deadlock conditions that have occurred in (3) above.
5. A single program that eliminates all the race conditions and deadlock conditions that have occurred in (1) and (3) above.

Progress Report Submission details:

Please refer to the progress report document for deadline and other relevant information.

Points distribution for this project will be as follows:

• Correct use of Interprocess communication	**[20 points]**
• Identification and illustration of race conditions	**[15 points]**
• Elimination of race conditions	**[15 points]**
• Identification and illustration of deadlock conditions	**[15 points]**
• Elimination of race conditions	**[15 points]**
• Report	**[10 points]**
• Presentation	**[10 points]**

Grading Criteria:

This is a group project. So, grading will focus on students' ability to work in the group and successfully completing the work. Each member in the group should coordinate as a team rather than individual and that's the key to score maximum points.

Points to remember:

- Failure to complete the project as a group will deduct 20 points, even though every individual has completed their part. As this is a group project, you need to learn to work in the group and meet the team's objective, and not individual objective.

- Work must be distributed uniformly among all group member and should be clearly specified in the report.
    - Be sure to submit a progress report that summarizes your design and all the test cases you have performed.
    - The progress report needs to include the code and the output.
        - Each student will write the code in their own file.
    - Include each student's contribution clearly. Failure to be specific on this part will make us difficult to give fair grades. So, it is your responsibility to clearly indicate what specific task each member is undertaking in the project.

- It is not necessary that all group members get equal points. It depends on what responsibility each student has taken, whether or not the student has delivered the task.

- It is the group's responsibility to alert instructor with any issue that is happening in the group.
    - Students are supposed to use the group created in the Canvas for all correspondence. That will serve as the proof, in case there is any dispute among the group members.
    - Time is very important. Group members are expected to respond quickly.
    - You need to have sufficient days to combine individual work. You are going to make one submission as a group and not individually.

- If any of the group member have plagiarized the code, then the complete project will be considered plagiarized, and all members will be awarded zero. So, each group member needs to be very alert of plagiarism issue, not only of their code but fellow member's code. Any plagiarism issue should be brought to the instructor's attention as soon as possible. <u>University and department have very strict policy on plagiarism and will be strictly followed.</u>

Submission Guidelines:

- Each group needs to submit the following files/folders:
  - Make 5 folders (and zip them). These folders contain only the source code.
    - Folder named "Group<A/B/C/D/E/F>_1" contains program(s) to illustrate the race conditions on your design.
    - Folder named "Group<A/B/C/D/E/F>_2" contains a program that eliminates the race conditions specified in the folder "1".
    - Folder named "Group<A/B/C/D/E/F>_3" contains program(s) to illustrate the deadlock scenario that can occur in the design.
    - Folder named "Group<A/B/C/D/E/F>_4" contains a single program that eliminates all the deadlock conditions specified in the folder "3".
    - Folder named "Group<A/B/C/D/E/F>_5" contains a single program that eliminates all the race conditions and deadlock conditions.
  - Make 5 .pdf files (name same as the folder name, as given above) to include all the source code belonging to that folder.
  - A report in .pdf format that:
    - Scenarios about the race conditions and deadlocks that occurred in the design.
    - Discussion on how you have handled the above race conditions and deadlock situations
    - Include all the test cases, along with the output. Include screenshots.
  - readMe.txt file that includes:
    - A general description of the project
    - Project status: Either complete or still in development.
      - If it is in development stage, please use a separate file to indicate each member's commitment, contribution and achievement, reason for not completion.
    - A guide to run the code, starting from signing the XCSX machine.
    - Known bugs and any bug fixes
    - Copyright and licensing information
  - any input file(s) that is/are used to run the program(s).

- Progress report needs to be submitted by the progress due date/end date (11/10/2021, 11:59 PM CDT). The due date and the end date are same for the progress report.
  - Please refer to the progress report file for further instruction

- Project presentation should be completed by 12/03/2021, 5:00 PM CDT. Every group need to make an appointment with the TA for the final presentation and demo by 11/30/2021 5:00 pm CDT.
  - All the students should be present for the presentation and the demo.
  - Before the presentation and demo, the group should have completed the project submission on Canvas.

- Final project will be evaluated together with the progress report. So, they should have a correlation. Commitments on the progress report should be reflected on the final submissions.

- Remember, since the TA and the instructor will have no way of knowing the project activities, we will consider what is presented in the submission as the final contribution. Based on that submission, we will be grading. So, if you fail to provide sufficient information, it will be difficult for us to grade fairly. So, it is expected to provide as much as information in the final submission.

Note:

1. Every group member's work must be in a separate file. This is applicable for all folders.
    a. Each .c file should include the header information, which should include:
        i. Group Number
        ii. Group member name
        iii. Email
    b. There should be sufficient comments in the program.
        i. Each function should be clearly described along with the input arguments and return values.

2. Report needs to be very detailed and very understandable.