נפתח: 8/05/2023, 00:14
מסתיים: 4/06/2023, 23:59

Threads and Synchronization

Q1. Synchronization (10 points)

a. Assume that there is an operating system which provides only Mutex as a synchronization tool. Can you implement a counting Semaphore using the Mutex? If yes, write a C# code for such a Semaphore class. In no, explain why and justify your answer.
Code template:

```
class MySemaphore

{

    public MySemaphore(int starting, int max) { }

    public bool WaitOne() { }

    public bool Release(int num=1) {}

}
```

b. Recall that you can use Semaphores to make sure that statement S1 occurs before statement S2 (look in the synchronization class slides).

Assume that you have 3 statements S1, S2, S3 (in three different threads). Can you use semaphores to make sure the statements are executed in the following order: S1, S2, S3? if yes: show how.

c. Read about Dekker's algorithm for the critical section problem (https://en.wikipedia.org/wiki/Dekker%27s_algorithm), and understand how it work. Compare between this solution to Paterson solution, list any advantages/disadvantages.

What to submit:

1. Add Q1. a-c answers to your report.pdf file (there is no need to submit C# file in this question)

Q2. Hackathon (30 points)

Q3. Implement Sharable Spreadsheet object (40 points)

In this exercise, you implement an in-memory shared spreadsheet management object. The spreadsheet object supports several elementary operations. The spreadsheet object is used by many threads concurrently and hence needs to be designed to be thread-safe.

The motivation: This is core object of a spreadsheet that can be shared between multiple concurrent users like Google Docs and Google Sheets.

The spreadsheet represent a table of n*m cells (n=rows, m=columns).
Each cell holds a string (C# string).
The spreadsheet starts at cell 0,0 (top, left).

The complete class can be download here<<<

Guidelines:

Assume the that object can be access concurrently by many threads (=users).

Assume that users can perform arbitrary operations concurrently.

Design your synchronization strategy carefully. How do you use mutex/semaphores, how many of them, what do you protect and where in your code.

In any case of errors, invalid parameters, bad/invalid index, and so on, simply throw an exception with an informative text.

Example of a bad design: a single lock for the whole spreadsheet.

?

What to submit:

1. Add to your report.pdf detailed description of your internal object design.

In the file, include a diagram that present your design. Especially, show the locks and the types. in your answer explain:

Which types of lock does your object use?

How many lock does your program use?

2. Submit SharableSpreadSheet.cs

Q4. Multiple users simulator (10 points)

In this part, you implement a console application simulator for multiple users that uses your sharable spreadsheet. In a real-life these types of simulators are used to test and debug multi-threaded classes under 'stress'

Your program should work as follow:

Simulator <rows> <cols> <nThreads> <nOperations> <mssleep>

1. Your simulator start with a creation of new spreadsheet in a size of rows*cols.
2. After the creation of empty table fill the empty spreadsheet with any random or prepared strings (e.g., testcell11, testcell12, testcell13,..., testcell21, testcell22,.. etc.). The actual content is up to you. E.g., you can insert word-by-word from an existing text file.
3. Start nThreads number of threads (users) concurrently works on the object. nOperations is the number of random operations each thread performs with a sleep of <sleep> between each operation it perform.

Guidelines:

Initially, your simulator create a rows*cols spreadsheet.

Create nThreads threads, which start works concurrently on the same spreadsheet. Each thread represent a different user.

Each thread performs a random sequence of nOperations operations, one after another with a sleep of <sleep> milliseconds between each operation.

The random operations includes all operations supported by the class, except of "load", and "save".

For example:

> Simulator 100 1000 30 100 500

Generates 100*1000 spreadsheet. It launches 30 concurrent threads. Each thread performs a sequence of 100 random operations. Each thread sleeps 500 milliseconds between every random operation.

Debugging:

Your simulator print logs of the operations during the simulation. You can choose your own debugging format. For example you may print the the user (thread ID) and the random operation performed. You may also print the whole spreadsheet at the beginning and the end of the simulation. The debugging printouts should be enabled in the version you submit, allowing the grader to look at your debugging printouts.

Example of a nice debugging printouts:

User [100]: [12:00:12] string "sherlock" inserted to cell [1,35].

User [102]: [12:00:12] string "Holmes" found in cell [400,400].

User [105]: [12:00:13] rows [1] and [200] exchanged successfully.

User [101]: [12:00:14] a new column added after column 80.

…

Make sure that you randomly choose the operation and the data you performs.

What to submit:

Console application project named ***Simulator***

Q5. Spreadsheet GUI application (10 points + up to 5 points bonus)
Write a separated project (SpreadsheetApp) that include a simple graphical user interface for your spreadsheet (display your spreadsheet). You can use controls such as DataGridView or ListView in C#.(e.g., https://www.youtube.com /watch?v=GyLlpBZGsrE)

Your application should support 3 operation - load, save, and

What to submit:

1. Add to your report.pdf one or more snapshots of the GUI screen, explaining the operations supported by your GUI app.

2. Windows form application project named **SpreadsheetApp**.

Up to 5 points bonus will be given for nicely designed and spreadsheets that support more operations with more complex GUI .elements

Submission summary:

Submit zip file with the following files -

1. report.pdf file with all relevant answers.

2. SharableSpreadSheet.cs file

3. Console application project named ***Simulator.***

*4. Windows form application project named **SpreadsheetApp.***

Good luck!

הוספת הגשה

מצב הגשה

מצב ההגשה	אין הגשה
מצב מתן הציון	לא ניתן ציון
הזמן שנותר	נותר 12 ימים 10 שעות

עדכון אחרון	-
הערות סטודנט להגשה	הערות (0)