Programming Project

Paras Raj Pahari
20th April, 2017

1. PROBLEM DEFINITION:

The number of threads are decided by user and supposed to be only power of 2 numbers. Threads will be like leaves in Binary tree whereas semaphores will be like internal nodes in Binary Tree. There should be a critical section at the top level. All threads should be competing to access the top level critical section.

2. Implementation:

Software Used: Eclipse

OS: Windows

Programming Language: Java

Our Projects have 4 modules:

Runner Module:

Semaphore and processes are stored in arrays, and they do have binary tree behavior allowing only one thread to make step only to the parent semaphore.

It initializes the data and processes User Input. It asks the number of threads to be input as 2, 4, 8…. (I.e. Power of 2)

It contains the main method to run the process.

My Process Module:

This module simulate the task processes.

As with the thread extension it has the number, the current semaphores and trace which is Array list data structure of semaphores in order to trace all semaphore visited from leaf to the root (critical section).

Here, the process starts several threads and every threads competes to acquire the semaphores in the upper level. If the semaphore is already acquired, other access to semaphore is blocked and displays the failure message with waiting status, else it successfully acquires the semaphores. Acquired semaphores are added to the process trace. Finally, only one thread will be able to access the semaphore [0] at the root level. Then enters into critical section and stays there for random duration of 500-1500 millisecond. Then, it releases the semaphore

After accessing the critical section, the thread releases all the semaphores in the process trace and return to the initial semaphore and sleeps for 250 to 500 milliseconds.

MySemaphore Module:

This module works with ID of Semaphore, and getting ID of current and parent semaphore.

Logger Module

This module logs the progress and status of threads while competing to access the critical section to the 'logs.txt' file and also prints to the console.

Result:

Any process thread path to critical resource conforms to the conditions described in specification. Output is as desired.