Teaching Announcements

2010/11

< Previous   Next >

COMP204 - PRACTICAL ASSIGNMENT 1
(2011)   18/02/2011

For the first practical assignment you are asked to write a Java program to implement the following producer-consumer situation with the use of semaphores:

The manager of a company employs 3 secretaries who are of varying ability, but who all work extremely fast. Secretary A is the most experienced secretary and is capable of typing up a letter once every second. Secretary B is less experienced and is capable of typing up a letter once every 2 seconds. Secretary C is the junior secretary and is capable of typing up a letter once every 4 seconds. When a secretary has typed up a letter he leaves it in the manager's tray for him to remove and sign. The manager removes and signs a letter from the tray once every 2 seconds. The tray can hold a maximum of 5 letters at a time. The tray's limited capacity sometimes causes the various workers to be delayed. For example, if the tray is full after a letter has been typed, the secretaries must wait until the manager makes a space available before they can add another letter to the tray. Similarly, the manager must wait for at least one letter to appear in the tray before he can take it out and sign it.

In your program, **make use of threads** to represent each of the workers (the 3 secretaries and the manager), so that they can work in parallel. You will also need to declare a tray object, and ensure that all communication is properly synchronised to avoid indeterminacy and deadlock. While an office worker is busy typing a letter or signing it, you should send that thread to sleep for the appropriate time period. This can be achieved with a call to:
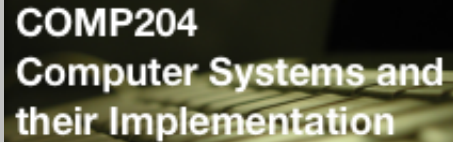
    Thread.sleep(m);

where m is the number of milliseconds for which the thread should suspend. **You should implement a simple binary semaphore** to protect the critical region - **do not use** the method type *synchronization* to manage a lock, or the methods *wait(), notify()* or *notifyAll()*.

The output from your program should take the form of a running commentary on the activity taking place in the office. An extract from it might look something like the output opposite, although it is up to you how you word this.

Run your simulation until the secretaries have typed and filed 7 letters each, and the manager has removed and signed all 21 letters.

The purpose of this exercise is to demonstrate that you know how to deal with problems that can occur in parallel computing. Therefore you do not need to hand in any design documentation but your code MUST be well commented so that it explains each step of your program. You may make

COMP204
Computer Systems and
their Implementation

Deadline:

3pm, Friday 4th Mar '11

```
1 4    0 4    1 2
Days   Hours   Mins
```

A PDF (printable) version of the assignment is also available

Sample Output

...
Secretary 1 has typed letter = 6
Secretary 3 has typed letter = 2
A letter has been removed from the tray. Tray = 4
...the Manager has taken a letter from the tray to sign; and has signed = 4
The Manager is ready to sign a letter...
A letter has successfully been added to the tray. Tray = 5
Secretary 2 has added letter 3 to the tray
Secretary 2 is ready to type a new letter
A letter has been removed from the tray. Tray = 5
A letter has successfully been added to the tray. Tray = 5
...the Manager has taken a letter from the tray to sign; and has signed = 5
The Manager is ready to sign a letter...
Secretary 2 has typed letter = 4
etc.
...

use of the code presented to you in the Comp 204 lecture notes concerning the producer-consumer problem.

SUBMISSION INSTRUCTIONS

Firstly, check that you have adhered to the following list:

1. All of your code is contained in a single file. Do NOT use more than one file. The file's name MUST be 'Office.java' (capital O; lower-case everything else). This means that the main class name must also be 'Office'.
2. Your program is written in Java, not some other language.
3. Your file is a simple text file; it must not be compressed or encoded in any way.
4. Your program compiles and runs on the computer science department's Windows system. If you have developed your code elsewhere (e.g. your home PC), port it to our system and perform a compile/check test before submission. It is your responsibility to check that you can log onto the department's system well in advance of the submission deadline.
5. Your program does not bear undue resemblance to anybody else's! Electronic checks for code similarity will be performed on all submissions and instances of plagiarism will be severely dealt with. The rules on plagiarism and collusion are explicit: do not copy anything from anyone else's code, do not let anyone else copy from your code and do not hand in 'jointly developed' solutions. The only code you may make use of is that presented to you in the lecture notes concerning the producer-consumer problem.

To submit your solution you must **PRINT IT OUT AND SUBMIT IT ELECTRONICALLY**, and adhere to the following instructions:

Printout:
- The printouts required are the source code for your Java program and the output it produces (i.e. no design documentation or test files are required: ***I will test your program to see if it works***)
- You must fill in a "Declaration on Plagiarism and Collusion Form", available from the One-Stop Student Shop, and attach this form to your printouts. (Work will NOT be marked unless accompanied by this form.)
- You must submit the above form and printouts of your work to the One-Stop Student Shop, room G09 on the ground floor of the Ashton Building before the deadline given above.

Electronic submission:
- Your code must be submitted to the departmental electronic submission system at: http://cgi.csc.liv.ac.uk/cgi-bin/submit.pl?module=comp204

- You need to login in to the above system and select 'Practical 1' from the drop-down menu. You then locate the file containing your program that you wish to submit, check the box stating that you have read and understood the university's policy on plagiarism and collusion, then click the 'Upload File' button.

Work will be accepted only if it is submitted both electronically AND in the form of a printout plus plagiarism declaration form, following the above

instructions.

Finally, please remember that it is always better to hand in an incomplete piece of work, which will result in some marks being awarded, as opposed to handing in nothing, which will guarantee a mark of 0 being awarded. Demonstrators will be on hand during the Comp 204 practical sessions to provide assistance, should you need it.