Com S 352 Exam 2

Friday, November 10, 2017

Time: 50 minutes

Last Name: _____
First Name: _____

The exam is closed-book. Please go over all the questions in the exam before you start working on it. Attempt the questions that seem easier first. If you see yourself getting stuck on one question, continue on and come back to it later if you have time. We are looking for brief answers to all questions. Good luck!

I. TRUE/FALSE questions. [2pts * 5 = 10pts]

____T_ A sequence of actions that access shared data by different processes must be done one at a time is called Critical Section.

___F__ Race conditions can be safely ignored.

___F__The Banker's Algorithm used in Deadlock Avoidance approach will not guarantee deadlock will not happen in the future.

__F____ The Deadlock Detection approach can prevent deadlock never happens.

__F____ Virtual Memory approach will not allow the size of a program to run on a computer system with a smaller size of main memory.

II. Short answer questions. [10pts * 2 = 20pts]

1. What is the main **difference** between the signal operation on **semaphore** and the signal operation **on condition variable in monitor**? [15pts]
   Answer:
   The signal() operation associated with monitors is not persistent in the following sense: if a signal is performed and if there are no waiting threads, then the signal is simply ignored and the system does not remember that the signal took place. If a subsequent wait operation is performed, then the corresponding thread simply blocks. In semaphores, on the other hand, every signal results in a corresponding increment of the semaphore value even if there are no waiting threads. A future wait operation would immediately succeed because of the earlier increment.

2. Describe the Segmentation with Paging System [15pts]
   Answer:
   Segmentation with paging does not suffer from external fragmentation but suffers from internal fragmentation. Because each segment is broken into multiple of page sizes and one process might not use all the pages. Therefore, there will be unused spaces inside the segment, which will cause internal fragmentation.

III. Banker's algorithm [30pts]

Consider a system where there are three processes (P1, P2, and P3) and three resource types (R1, R2, and R3). Initially, the available instances of R1, R2 and R3 are 7, 7 and 10, respectively. The following matrices MAX and ALLOC represent the processes' declarations of resource needs and the current resource allocation status.

MAX

	R1	R2	R3
P1	3	6	8
P2	4	3	2
P3	3	5	4

ALLOC

	R1	R2	R3
P1	2	2	5
P2	2	0	1
P3	1	2	4

(1) Is the current allocation state safe? Use the banker's algorithm to justify your
Answer:

Need

	R1	R2	R3
P1	1	4	3
P2	2	3	1
P3	2	3	0

Available resource: <R1, R2, R3 > = < 2 ,3 ,0 >
Current state is safe! Completion sequence P3,P2/P1,P1/P2.

(2) Would the following requests be granted in the current state by the banker's algorithm? Why?

a. Process P3 requests (0, 1, 0)
Answer:
This request cannot be granted because then P3 needs <2,4,0> resources to be completed which is greater than available resources. So system will be in deadlock.

b. Process P2 requests (2, 1, 0)
Answer:
This request can be granted. Because now P2 needs <4, 4, 1> to be completed. But P3 needs <2 ,3 ,0> to be completed and system has it available. When P3 is done, P2 can finish and then P1 can finish. So, system will be in safe state.

IV. Process Synchronization. [30pts]

1. **Write a monitor to solve the following readers-writers problem: If readers and writers are both waiting, then it alternates between readers and writers. Otherwise it processes them normally (i.e., readers concurrently and writers**
2. **serially).**

Solution:

```
readers-writers : monitor;
begin
readercount : integer;
busy : boolean;
OKtoread, OKtowrite : condition;

procedure startread;

    begin
    if busy OR OKtowrite.queue then OKtoread.wait;
    readercount := readercount + 1;
    if NOT OKtowrite.queue then OKtoread.signal;
    (* a reader signals the next waiting reader if there is no waiting writer *)
    end startread;

procedure endread;

    begin
    readercount := readercount - 1;
    if readercount = 0 then OKtowrite.signal;
    end endread;

procedure startwrite;

    begin
    if busy OR readercount ≠ 0 then OKtowrite.wait;
    busy := true;
    end startwrite;

procedure endwrite;

    begin
    busy := false;
    if OKtoread.queue then OKtoread.signal
                      else OKtowrite.signal;
    end endwrite;

begin (* initialization *)
readercount := 0;
busy := false;
end;
end readers-writers;
```