Quiz 03 Questions

Bookmark this page

Multiple Choice

1.0/1.0 point (graded)

Suppose we have three processes P1, P2, P3 and we want to execute it in order like P2,P3 and P1, then how many minimum semaphores will be needed for their orderly execution?

- (•) two
- ( ) four
- ( ) three
- ( ) one

✔

Submit You have used 1 of 1 attempt

Show Answer

Multiple Choice

1.0/1.0 point (graded)

Consider a semaphore mutex=1, and several processes are trying to enter the Critical Section (CS). If the processes call signal(mutex) in their entry section and wait(mutex) in their exit section, which of the following will occur?

- ( ) A deadlock
- ( ) Starvation
- ( ) Bounded waiting
- (•) Multiple Processes in CS

✔

Submit You have used 1 of 1 attempt

Show Answer

Multiple Choice

1.0/1.0 point (graded)

Suppose the 2 processes in peterson's solution are numbered 0 and 1. Which of the following is a valid code statement in the entry section of process 1?

- ( ) flag[1] = false
- (•) turn = 0

flag[0] = false

turn = 1

Submit You have used 1 of 1 attempt

Show Answer

Multiple Choice

1.0/1.0 point (graded)

In reader-writer problem _____ number of processes get queued in the mutex (the semaphore that works for the readers only).

n

n*n

n+1

n-1

Submit You have used 1 of 1 attempt

Show Answer

Checkboxes

2.0/2.0 points (graded)

Which of the follwings is/are false? (There might be multiple correct answers)

Peterson's solution sometimes called two processes critical section solution

Semaphore is implemented as hardware based solution.

Basic Compare and swap ensures bounded waiting

Mutually exclusiveness ensures no two processes can enter a critical section simultaneously.

Submit You have used 1 of 1 attempt

Show Answer

Checkboxes

2.0/2.0 points (graded)

Which of the follwings are true? (There might be multiple correct answers)

In race condition multiple processes cannot execute shared information concurrently.

Process synchronization ensures data consistency.

Race condition does not create data inconsistency in a shared region.

Semaphore is used to solve critical section problem.

Submit You have used 1 of 1 attempt

Show Answer

Multiple Choice

2.0/2.0 points (graded)

The enter_CS( ) and leave_CS( ) functions to implement critical section of a process are realized using test and set instruction as follows-

void enter_CS(X){

while (test-and-set(X));

}

void leave_CS(X){

X = 0;

}

In the above solution, X is a memory location associated with the CS and is initialized to 0. Now, consider the following statements-

i. The above solution to CS problem is deadlock-free

ii. It ensures mutual exclusion

iii. The processes enter CS in FIFO order

iv. More than one process can enter CS at the same time.

Which of the above statements is true?

- ○ II only
- ◉ I and II
- ○ II and III
- ○ IV only

✔

Submit You have used 1 of 1 attempt

Show Answer

Checkboxes

0.0/2.0 points (graded)

Suppose someone suggested the following code to the critical section problem for many processes (Process 0 to N where N > 2) using the compare and swap instruction. Decide if the proposed solution is correct or identify any issues with the proposal. There is no partial marking for this question. (&lock is how you access the lock variable in a code low level code, so don't worry about the & sign in the code.)

```
do {
      waiting[i] = true;
      key = true;
      while (waiting[i] && key) {
            key = compare_and_swap(&lock, 0, i) != 0;
      }
      waiting[i] = false;
      /* critical section */
      j = (i + 1) % n;
      while ((j != i) && !waiting[j]) {
            j = (j + 1) % n;
      }
      if (j == i) {
            lock = 0;
      } else {
            waiting[j] = false;
      }
      /* remainder section */
} while (true);
```

Select the right ones from the following -

- ☑ Solution is correct.
- ☑ Solution violates mutual exclusion. ✔
- ☑ Solution violates progress requirement

- [ ] Solution violates bounded waiting time requirement

✖

Submit You have used 1 of 1 attempt

Show Answer

Answers are displayed within the problem

Checkboxes

0.0/3.0 points (graded)

Assume the following processes are scheduled in the CPU using the round-robin scheduling with time quanta **2 ms**. All the processes execute the same code that involves using a SPINLOCK mutex (spinlock means the process busy-waits until other process releases and it acquires the mutex. And this waiting happens inside the acquire statement). The code is as follows:

acquire(&lock)

/* critical section */

release(&lock)

/* remainder section */

Process No	Arrival Time	Burst Time
P0	0	6ms
P1	X	6ms
P2	Y	6ms
P3	Z	6ms

The CPU burst time for the critical section part of the code is **3ms**, and for the remainder section, it is again 3ms. Thus, all processes' CPU burst time would be 6ms if they do not wait to acquire the mutex lock.

Consider, X = 1, Y = 1, Z = 7.

Select the processes that will be waiting to acquire the mutex at time **10ms**. There is no partial marking for this question.

- [ ] P1
- [x] P2 ✔
- [x] P3
- [ ] P4

✖

Submit You have used 1 of 2 attempts

Show Answer

Answers are displayed within the problem

Previous Next