ⓘ Students have either already taken or started taking this quiz, so take care when editing it. If you change any quiz questions in a significant way, you might want to consider re-grading students' quizzes who took the old version of the quiz.

Points 5 Published ⋮

Details Questions

Show question details

Question 1 pts

What is the disadvantage of implementing locks by disabling OS interrupts as follows:

```
1  void lock() {
2      DisableInterrupts();
3  }
4  void unlock() {
5      EnableInterrupts();
6  }
```

- This implementation gives control of privileged operations to the user
- This implementation fails to guarantee mutual exclusion
- This implementation fails to ensure fairness
- This implementation is inefficient

Question 1 pts

What can be said about the following implementation of a lock:

```
1  typedef struct __lock_t { int flag; } lock_t;
2
3  void init(lock_t *mutex) {
4      // 0 -> lock is available, 1 -> held
5      mutex->flag = 0;
6  }
7
8  void lock(lock_t *mutex) {
9      while (mutex->flag == 1)  // TEST the flag
10         ; // spin-wait (do nothing)
11     mutex->flag = 1;          // now SET it!
12 }
13
14 void unlock(lock_t *mutex) {
15     mutex->flag = 0;
16 }
```

- It fails to guarantee mutual exclusion
- It gives the user access to privileged operations

- It is efficient
- It ensure fairness

Question 1 pts

What can be said about a spin lock is constructed using a test-and-set instruction as follows:

```
typedef struct __lock_t {
    int flag;
} lock_t;

void init(lock_t *lock) {
    // 0: lock is available, 1: lock is held
    lock->flag = 0;
}

void lock(lock_t *lock) {
    while (TestAndSet(&lock->flag, 1) == 1)
        ; // spin-wait (do nothing)
}

void unlock(lock_t *lock) {
    lock->flag = 0;
}
```

- It guarantees that only one thread can enter the critical section at a time.
- It prevents starvation among threads.
- It avoids "busy waiting" (i.e. waiting while still being busy executing instructions)
- It is highly efficient for acquiring locks on frequently used shared resources.

Question 1 pts

Which of the following statements accurately describes a key aspect of threads in a multithreading environment?

- Threads share the same address space, allowing them to access and modify each other's variables directly.
- Critical sections in threads work fine if multiple threads simultaneously execute the same code without any synchronization
- Atomicity in threading refers to the ability of a single thread to execute multiple operations with the single hardware instruction
- Threads in a multithreading environment always operate independently, with no need for coordination or synchronization mechanisms.

Question 1 pts

Which of the following statements accurately describes the usage of `pthread_create` and `pthread_join` functions in a multithreading program?

- pthread_create is used to create a new thread, and pthread_join is used to wait for the termination of a specific thread and collect its exit status.
- pthread_create is used to create a new thread, while pthread_join is used to terminate an existing thread.
- pthread_create is responsible for joining multiple threads into a single execution flow, and pthread_join is used to create new threads.

pthread_create_t is a function used for thread creation, and pthread_join is used for managing thread priorities in a multithreading environment.

+ New question + New question group Find questions

Notify users this quiz has changed

Cancel Save