6_Quiz _L2

- Due Dec 2 at 2:15pm
- Points 10
- Questions 10
- Available until Dec 2 at 2:15pm
- Time Limit 15 Minutes

Instructions

This is a timed, closed book, closed notes quiz.
There are 10 questions.
Total time is 15 mins.
You can navigate front and back.
For code related questions, write the whole code in the area provided.
There should not be any other tab or window open on your laptop while you are attempting the quiz.
Offenders will get a 0.
You cannot use chatgpt or any other AI tool to obtain answers. Offenders will get a 0.
Good luck!!!

This quiz was locked Dec 2 at 2:15pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	9 minutes	10 out of 10

ⓘ Correct answers are hidden.

Score for this quiz: 10 out of 10
Submitted Dec 2 at 2:02pm
This attempt took 9 minutes.

Question 1
1 / 1 pts

Concurrency requires implementation of _____ and _____.

- [ ] Race Condition
- [x] Synchronization
- [x] Mutual Exclusion
- [ ] Critical Section

Question 2
1 / 1 pts

Race conditions happen when _____. (Choose all that apply)

- [x] Due to interrupts as one thread is unable to finish its update of the shared variable before a second thread accesses the same shared variable.
- [x] Two or more threads try to access a shared variable.
- [x] Two or more threads enter critical section at the same time
- [ ] CPU is idle

Question 3
1 / 1 pts

Which of the following functions in the POSIX threads (pthreads) API is used to create a new thread?

- pthread_mutex_lock()
- pthread_exit()
- pthread_join()
- pthread_create()

Question 4
1 / 1 pts

What is the primary purpose of a mutex in a multi-threaded program?

- To automatically distribute workloads between threads
- To terminate threads after execution
- To provide mutual exclusion for shared resource access
- To enable threads to communicate with each other

Question 5
1 / 1 pts

In POSIX threads, which function is used to lock a mutex?

- pthread_mutex_lock()
- pthread_lock()
- pthread_mutex_init()
- pthread_mutex_acquire()

Question 6
1 / 1 pts

What will happen if a thread attempts to lock a mutex that it has already locked in POSIX threads?

- The thread will raise a segmentation fault.
- The thread will be blocked indefinitely.
- The thread will sleep
- The thread will lock the mutex and continue execution.

Question 7
1 / 1 pts

In lock-based concurrent data structures, which strategy is often used to ensure that updates to shared data are atomic?

- Read-write locks
- Mutex Locks
- Condition variables
- Spinlocks

Question 8
1 / 1 pts

What is a significant disadvantage of using mutex locks in concurrent programming?

- They cannot be used in multi-core processors.
- They prevent race conditions but can lead to excessive CPU usage.
- They introduce the risk of deadlock and contention between threads.
- They guarantee that all threads will execute concurrently without interference.

Question 9

1 / 1 pts

What is the purpose of using the mutex `queue->lock` in the `enqueue` and `dequeue` functions in the followi code:

```c
#include <stdlib.h>
#include <pthread.h>

typedef struct Queue {
    int* data;
    int front;
    int rear;
    int capacity;
    pthread_mutex_t lock;
} Queue;

void initQueue(Queue* queue, int capacity) {
    queue->data = (int*)malloc(sizeof(int) * capacity);
    queue->front = queue->rear = 0;
    queue->capacity = capacity;
    pthread_mutex_init(&queue->lock, NULL);
}

void enqueue(Queue* queue, int value) {
    pthread_mutex_lock(&queue->lock);
    if ((queue->rear + 1) % queue->capacity == queue->front) {
        printf("Queue is full\n");
    } else {
        queue->data[queue->rear] = value;
        queue->rear = (queue->rear + 1) % queue->capacity;
    }
    pthread_mutex_unlock(&queue->lock);
}

int dequeue(Queue* queue) {
    pthread_mutex_lock(&queue->lock);
    if (queue->front == queue->rear) {
        printf("Queue is empty\n");
        pthread_mutex_unlock(&queue->lock);
        return -1;  // Empty queue
    } else {
        int value = queue->data[queue->front];
        queue->front = (queue->front + 1) % queue->capacity;
        pthread_mutex_unlock(&queue->lock);
        return value;
    }
}

int main() {
    Queue queue;
    initQueue(&queue, 5);
    enqueue(&queue, 1);
    enqueue(&queue, 2);
    enqueue(&queue, 3);
```

```
50
51      printf("Dequeued value: %d\n", dequeue(&queue));
52      printf("Dequeued value: %d\n", dequeue(&queue));
53
54      return 0;
55  }
```

- [ ] The mutex is used to prevent memory allocation errors when the queue becomes full.

Question 10

1 / 1 pts

Which of the following best describes **"critical section"** in the context of concurrency?

- [x] A section of code that only one thread should access at a time to ensure data consistency.
- [ ] A section of memory shared between all threads for faster access.
- [ ] A section of code where multiple threads are allowed to run concurrently without synchronization.
- [ ] A section of code where threads are prevented from executing at all.

Quiz Score: 10 out of