Synchronization

1. The principle of process isolation in an operating system means that processes must not have access to the address spaces of other processes or the kernel. However, processes also need to communicate.
    a. Give an example of such communication

There are two ways intercommunication between processes can be achieved, either through the use of shared memory segments or through the use of message passaging.

    b. How does this communication work?

Shared memory segments: The processes share a common memory section where the data that all the processes need is stored and updated.

Message passaging: Allows the processes to communicate through passing messages. First a communication link is established, then the processes start sending messages using basic primitives.

    c. What problems can result from inter-process communication?

The use of inter process communication does not come without its problems, chief of which being the security vulnerability risks. The vulnerability risk arises as through the use of ICP a process may be able to access and modify data belonging to another process. Additionally inconsistencies may arise if multiple processes try to access and modify the data at the same time.

2. What is a critical region? Can a process be interrupted while in a critical region? Explain

A critical region refers to a segment of memory that must be accessed exclusively by one thread at a time. This is done in order to maintain information integrity and simultaneously avoid race conditions. Like all other processes even one in a critical region can be interrupted.

3. Explain the difference between busy waiting (polling) versus blocking (wait/signal) in the context of a process trying to get access to a critical section

The difference between busy waiting and blocking in regards to gaining access to a critical section is that, given that the critical section is currently occupied, the busy waiting will keep repeatedly asking if the critical section is available, consuming 100% of CPU cycles in the process, whilst the blocked process will be put into a sort of sleep until the operating system notifies it that the critical section is available.

4. What is a race condition? Give a real-world example

A race condition occurs when two or more operations are attempted to run simultaneously where the nature of the device or system requires that the operations be completed in the proper sequence. A real world example of a race condition could be flipping two light switches simultaneously with them both being connected to the same light bulb, only for the circuit to blow.

5. What is a spin-lock, and why and where is it used?

Spin-lock is a type of synchronization method used by the operating system to ensure that shared resources only can be accessed by one thread or process at a time. The spinlock type of synchronization uses busy waiting to check for when the lock becomes available.

6. List the issues involved with thread synchronization in multi-core architectures. Two lock algorithms are MCS and RCU (read-copy-update). Describe the problems they attempt to address. What hardware mechanism lies at the heart of each?

The issues involved with thread synchronization in multi-core architectures are as follows; Dividing activities, balance, data splitting, data dependency and testing and debugging. MCS attempts to address the problems of Mutual Exclusion, deadlocks and scalability. RCU attempts to address Blocking reads. The hardware mechanism at the heart of both MCS and RCU is the concept of memory barriers. Memory barriers are hardware constraints that enforce memory ordering and visibility constraints in a multi-core system.

Deadlocks

1. What is the difference between resource starvation and a deadlock?

Starvation relates when a process requires access to data in order to execute but keeps getting stalled by higher priority processes. A steady stream of high priority processes can result in a lower priority process never getting to execute. A deadlock occurs when each process holds onto data and awaits data held by another process. In this situation neither process gets access to the data it needs, resulting in a deadlock.

2. What are the four necessary conditions for a deadlock? Which of these are inherent properties of an operating system?

The four necessary conditions for a deadlock are Mutual Exclusion, Hold and Wait, No Preemption and Circular Wait. Of these Mutual Exclusion is inherent properties of the operating system

3. How does an operating system detect a deadlock state? What information does it have available to make this assessment?

The operating system detects a deadlock state through the use of the resource allocation graph. Detecting the presence of a cycle in this graph is a sufficient condition for a deadlock. It is however not enough to say with certainty that a deadlock has occurred. In order to make sure if a deadlock has occurred or not, the wait for graph algorithm is run. This algorithm functions by creating a wait for graph. This is a directed graph that keeps track of the dependencies between resources and processes.

Scheduling

1. Uniprocessor scheduling
    a. When is first-in-first-out (FIFO) scheduling optimal in terms of average response time? Why?

FIFO scheduling is optimal if the tasks to be executed are equal in size due to minimizing overhead by switching only when a task is done.

    b. Describe how Multilevel feedback queues (MFQ) combines first-in-first-out, shortest job first, and round robin scheduling in an attempt at a fair and efficient scheduler. What (if any) are its shortcomings?

Multilevel feedback queues (MFQ) combine first-in-first-out, shortest job first, and round robin scheduling by having three different priority queues. Each of these queues also have a different overall priority. The highest priority queue usually uses round robin scheduling, whilst the lower priority queues use first-in-first-out and shortest job first scheduling. If a process exceeds the allowed time in the highest priority queue it will be taken out and moved to a lower priority queue. Additionally if a process in a lower priority queue is not getting access it will be subject to a process called aging, where in the process moves up to higher priority queues. The shortcomings of MFQ are as follow; Complexity, starvation, overhead and difficulty in predicting process behavior

2. Multi-core scheduling
    a. Similar to thread synchronization, a uniprocessor scheduler running on a multi-core system can be very inefficient. Explain why (there are three main reasons). Use MFQ as an example

The three main reasons a uniprocessor scheduler, such as MFQ, running on a multicore system can be very inefficient are; limited cpu utilization, inefficient task distribution and limited parallelism. The reason why this leads to inefficiency is that a uniprocessor scheduler like MFQ can not efficiently leverage all the available cores, resulting in one core being overworked whilst the other cores remain underutilized. Additional unciprocessor schedulers like MFQ lack the ability to harness the parallelism offered by multi-core systems

b. Explain the concept of work-stealing.

The concept of work-stealing is a strategy for ensuring load-balancing. In the process of work-stealing idle threads seekout and steal tasks from other threads queues. This helps to ensure a more even distribution of work, whilst also providing efficient resource utilization.