Threads ,concurrency

This is a mechanism by which only one thread at a time is granted access to some data.

Concurrency is the execution of a set of multiple instruction sequences at the same time. This occurs when there are several process threads running in parallel. These threads communicate with the other threads/processes through a concept of shared memory or through message passing.

In a multithreaded process on a single processor, the processor can switch execution resources between threads, resulting in concurrent execution. Concurrency indicates that more than one thread is making progress, but the threads are not actually running simultaneously.

Mutual Exclusion in Synchronization:During concurrent execution of processes, processes need to enter the critical section (or the section of the program shared across processes) at times for execution. It might happen that because of the execution of multiple processes at once, the values stored in the critical section become inconsistent. In other words, the values depend on the sequence of execution of instructions – also known as a race condition. The primary task of process synchronization is to get rid of race conditions while executing the critical section.

This is primarily achieved through mutual exclusion.

Mutual Exclusion:

It Is a property of process synchronization that states that "no two processes can exist in the critical section at any given point of time". The term was first coined by Dijkstra. Any process synchronization technique being used must satisfy the property of mutual exclusion, without which it would not be possible to get rid of a race condition.

The need for mutual exclusion comes with concurrency. There are several kinds of concurrent execution:

1:Interrupt handlers

2:Interleaved, preemptively scheduled processes/threads

3:Multiprocessor clusters, with shared memory

4:Distributed systems

Methods are used in concurrent programming to avoid the simultaneous use of a common resource, such as a global variable, by pieces of computer code called critical sections •

The requirement of mutual exclusion is that when process P1 is accessing a shared resource R1, another process should not be able to access resource R1 until process P1 has finished its operation with resource R1.

Examples of such resources include files, I/O devices such as printers, and shared data structures.

Conditions Required for Mutual Exclusion:

According to the following four criteria, mutual exclusion is applicable:

1:When using shared resources, it is important to ensure mutual exclusion between various processes. There cannot be two processes running simultaneously in either of their critical sections.

2:It is not advisable to make assumptions about the relative speeds of the unstable processes.

3:For access to the critical section, a process that is outside of it must not obstruct another process.

4:Its critical section must be accessible by multiple processes in a finite amount of time; multiple processes should never be kept waiting in an infinite loop.

Approaches To Implementing Mutual Exclusion

1. **Software method: Leave the responsibility to the processes themselves. These methods are usually highly error-prone and carry high overheads.**

2. **Hardware method: Special-purpose machine instructions are used for accessing shared resources. This method is faster but cannot provide a complete solution. Hardware solutions cannot give guarantee the absence of deadlock and starvation.**

3. **Programming language method: Provide support through the operating system or through the programming language.**

Requirements of Mutual Exclusion

1:At any time, only one process is allowed to enter its critical section.

2:The solution is implemented purely in software on a machine.

3:A process remains inside its critical section for a bounded time only.

4:No assumption can be made about the relative speeds of asynchronous concurrent processes.

5:A process cannot prevent any other process from entering into a critical section.

6:A process must not be indefinitely postponed from entering its critical section