39110338
Goluguri Venkata
Sai Rama Reddy

SCSA1501 - Operating Systems

Assignment - II

Fill in the blanks:

1) Producer consumer problem is otherwise called as The bounded-Buffer problem.

2) Any code segments that access shared variables and that has to be executed as an atomic action is called as A critical section.

3) If a process is executing in critical section, then no other process can be executing in this critical section is called Mutual Exclusion

4) Two types of semaphores are Binary Semaphores and Counting Semaphores

5) Concurrency is the decomposability property of a program into order- independent or partially ordered components

Short Questions:

1) What is the use of mutex lock?

Ans → Context switch will take place again and again but no thread would be able to execute the locked region of code until the mutex lock over it is released.

→ mutex lock will only be released by the thread who locked it.

→ So this ensures that once a thread has locked a piece of code then no other thread can execute the same region until it is unlocked by the thread who locked it.

2) Define deadlock. State the necessary conditions for a deadlock.

Ans. Deadlock is a situation which involves the interaction of more than one resources and processes with each other.

When a process requests for the resource that is been held another process which needs another resource to continue, but is been held by the first process. then it is called deadlock.

Conditions necessary for a dead lock are

→ mutual Exclusion

→ Hold and wait

→ No Preemption

→ Circular wait

3) What is meant by cooperating process?

Ans. In the computer system, there are many processes which may be either independent processes or cooperating processes that run in the operating system. A process is said to be independent when it cannot affect or be affected by any other processes that are running the system. It is clear that any process which does not share any data (temporary or persistent) with any another process than the process independent. On the other hand, a cooperating process is one which can affect or affected by any another process that is running on the computer. The cooperating process is one which shares data with another process.

4) What is the deadlock avoidance?

Ans. In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.

In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

5) What is RAG in operating system?

Ans. As Bankers Algorithm using some kind of table like allocation, request, available all that thing to understand what is the state of the system. Similarly

If you want to understand the state of the system instead of using those tables, actually tables are very easy to represent and understand it, but then still you could even represent the same information in the graph. That graph is called Resource Allocation Graph (RAG).

Long Question:

1) Explain in detail about critical section and explain how to implement it.

Ans. In concurrent programming, concurrent accesses to shared resources can lead to unexpected or erroneous behaviour. So parts of the program where the shared resource is accessed need to be protected in ways that avoid the concurrent access. This protected section is the critical section or critical region. It cannot be executed by more than one process at a time. Typically, the critical section accesses a shared resource, such as data structure, a peripheral device, or a network connection, that would not operate correctly in the context of multiple concurrent accesses.

Implementation of critical Sections

The implementation of critical sections vary among different operating systems.

A critical section will usually terminate in finite time, and a thread, task or process will have to wait for a fixed time to enter it. To ensure exclusive use of critical sections some synchronization mechanism is required at the entry and exit of the program.

Critical section is a piece of program that requires mutual exclusion of access.

In case of mutual exclusion (mutex), one thread blocks a critical section by using locking techniques when it needs to access the shared resource and other threads have to wait to get their turn to enter into the section. This prevents conflicts when two or more threads share the same memory space and want to access a common resource.

The simplest method to prevent any change of processor control inside the critical section is implementing a semaphore. In uni processor systems, this can be done by disabling interrupts on entry into the critical section, avoiding system calls that can cause a context switch while inside the section, and restoring interrupts to their previous state on exit. Any thread of execution entering any critical section anywhere in the system will, with this implementation, prevent any other thread, including an interrupt, from being granted processing time on the CPU - and therefore from entering any other critical section or, indeed, any code whatsoever - until the original thread leaves its critical section.

This brute-force approach can be improved upon by using semaphores. To enter a critical section, a thread must obtain a semaphore, which it releases on leaving the section. Other threads are prevented from

entering the critical section at the same time as the original thread, but are free to gain control of the CPU and execute other code, including other critical sections that are protected by different semaphores. Semaphore locking also has a time limit to prevent a deadlock condition in which a lock is acquired by a single process for an infinite time, stalling the other processes which need to use the shared resource protected by the critical section

Uses of critical Sections

→ Kernel level critical Sections

Critical sections prevent thread and process migration between processors and the preemption of processes and threads by interrupts and other processes and threads. Critical sections often allow nesting. Nesting allows multiple critical sections to be entered and exited at little cost

→ Critical sections in Data structures

In parallel programming, the code is divided into threads. The read-write conflicting variables are split between threads and each thread has a copy of them. Data structures like linked lists, trees, hash tables etc. have data variables that are linked and cannot be split between threads and hence implementing parallelism is very difficult.

→ Critical sections in computer networking

Critical sections are also needed in computer networking. when the data arrives at network sockets, it may not arrive in ordered format. Then critical section is used to protect the data of socket.