Critical Section Problem

- Consider a system consisting of *n* processes. All competing to use some shared data. Each process has a segment of code, called a critical section, in which shared data is accessed.
- It is important that when one process is executing its critical section, No other process is to be allowed to execute in its critical section. That is, no two processes are executing in their critical sections at the same time.
- The critical section problem is to design a protocol that the processes can use to cooperate. The general structure of a process:

```
do
{
        entry section
                critical section
        exit section
                remainder section
} while (true);
```

A solution to the critical section problem must satisfy the following three requirements:

1. **Mutual Exclusion:** If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.
2. **Progress:** If no process is executing in its critical section and some processes wish to enter their critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.
3. **Bounded Waiting:** There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

Approaches to handle critical section

Two general approaches are used to handle critical sections in operating systems

1. **Preemptive Kernels**
   - A preemptive kernel allows a process to be preempted while it is running in kernel mode.
   - A preemptive kernel is more suitable for real time programming, as it will allow a real time process to preempt a process currently running in the kernel.
   - Example- Windows XP, Windows 2000
2. **Non-preemptive kernels**
   - A non-preemptive kernel does not allow a process running in kernel mode to be preempted.
   - A non-preemptive kernel is essentially free from race conditions on kernel data structures, as only one process is active in the kernel at a time.
   - Example- Solaris, Linux 2.6

Solution of critical section problem

1. **Software Based**
    a. Dekker Solution
    b. Peterson Solution
2. **Hardware Based**
    a. Test & Set
    b. Swap
3. **Semaphores**
    a. Binary
    b. Counting
4. **Monitors**

Questions asked in semester exam:

Question: What is a Critical Section problem? Give the conditions that a solution to the critical section problem must satisfy.

[2018-2019] [7 Marks]

Question: State the Critical Section problem. Illustrate the software based solution to the Critical Section problem.

[2017-2018] [7 Marks]

Question: What is Critical Section?

[2016-2017] [2 Marks]

Question: Give the principles, mutual exclusion in critical section problem. Also discuss how well these principles are followed in Dekker's solution.

[2016-2017] [5 Marks]

Question: What do you understand by critical section?

[2015-2016] [2 Marks]

Question: What do you understand by critical section? Discuss bakery algorithm. Also show how it satisfies the requirements of a mechanism to control access to critical section.

[2014-2015] [10 Marks]

Question: List the essential requirements of Critical Section Implementation.

[2013-2014] [5 Marks]

Question: What is critical section? Discuss.

[2010-2011] [5 Marks]

Question: Provide the solution of critical section problem.

[2009-2010] [5 Marks]

Question: What is critical section? Design algorithm to solve this problem.

[2008-2009] [5 Marks]

Question: Define a critical section problem and its solution by using Semaphore. Use this approach to solve Producer/Consumer problem.

[2006-2007] [5 Marks]