

# Processes Synchronization

## Definition

Problem arise when two processes enter their critical section at the same time

## Critical Section Problem

When one process enters it critical section, other processes are banned from doing it

Only the processes in the remainder section ( after they have entered the crticial section) have the rights to decide which processes get to enter critical section

Upper bound exists for the number of times other processes are allowed to enter critical section between the time a process has made its request and the time that the request is granted

## Critical Section Solution

Using preemptive or non-preemptive approach

preemptive approach would forcefully stalls the process currently running in the CPU

Non-preemptive would not wait until it finishes

Peterson's algorithms

Locks (Hardware level)

Using atomic operation like test\_and\_set and data structure

Mutex lock

cons: keeps cpu busy checking the availability of the data

pros: no context switching

Semaphore

binary semaphore

oftentimes as alternative to mutex lock

unrestricted semaphore

reduce the busy waiting to waiting queues and semaphore operation like wait and signal

still needs to cancel the interrupt in the case of multiprocessors

Monitors

Higher level abstraction of semaphores

## Other challenges

deadlocks

where either process is waiting for an event that could only be cause by each other

starvation

a process could potentially wait for ever due to priority based scheduling

priority inversion

a solution to a problem where a resources needed by high priority is accessed by process with low priority, which is interrupted when a process with medium priority comes in and preempt the low priority

Solution is priority inheritance protocol where the L would temporarily obtain a status as high as H and revert back after it is finished