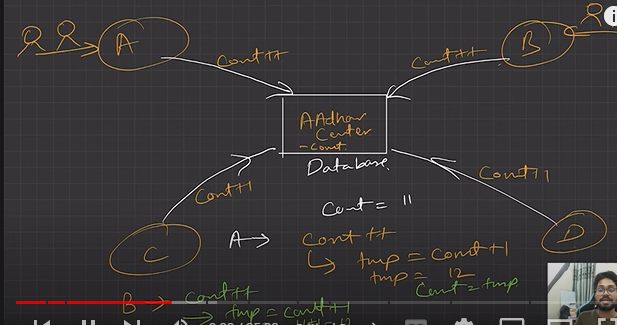
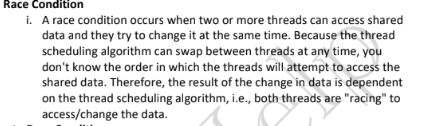
Critical Section Problem || Race condition in OS



Critical Section (C.S)

1. The critical section refers to the segment of code where processes/threads access shared resources, such as common variables and files, and perform write operations on them. Since processes/threads execute concurrently, any process can be interrupted mid-execution.
2. Here DB is critical section

Race Condition:



Results in inconsistent output, misses out concurrent operations

Solution to Race Condition:

1. Make critical section an atomic operation
2. i.e executed in single CPU
3. Mutual Exclusion
   1. T1 -> T2 or T2->T1
   2. If T1 locks in thread T2 cannot enter. And vise versa
   3. Semaphores

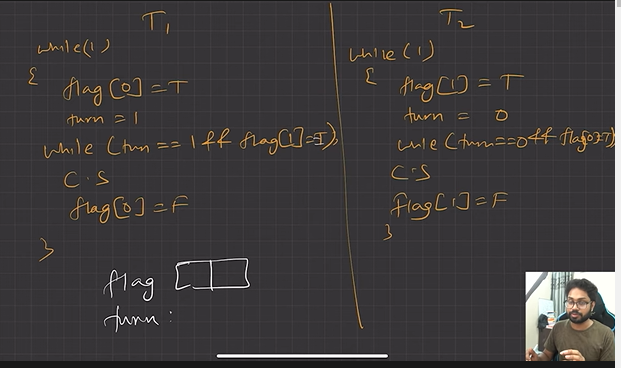
Soln to Critical section should have three conditions?

1. Mutual exclusion
2. Progress
3. Bounded Waiting

Can we use a simple flag variable to solve the problem of race condition?

-> No.

6. Peterson’s solution can be used to avoid race condition but holds good for only 2 process/threads.



Turn = 0/1

FLAG[2] = indicate a flag is ready to enter the CS, flag[i] = true implies that flag[i] is ready

But peterson is only safe for 2 threads

