Problems with prev solns

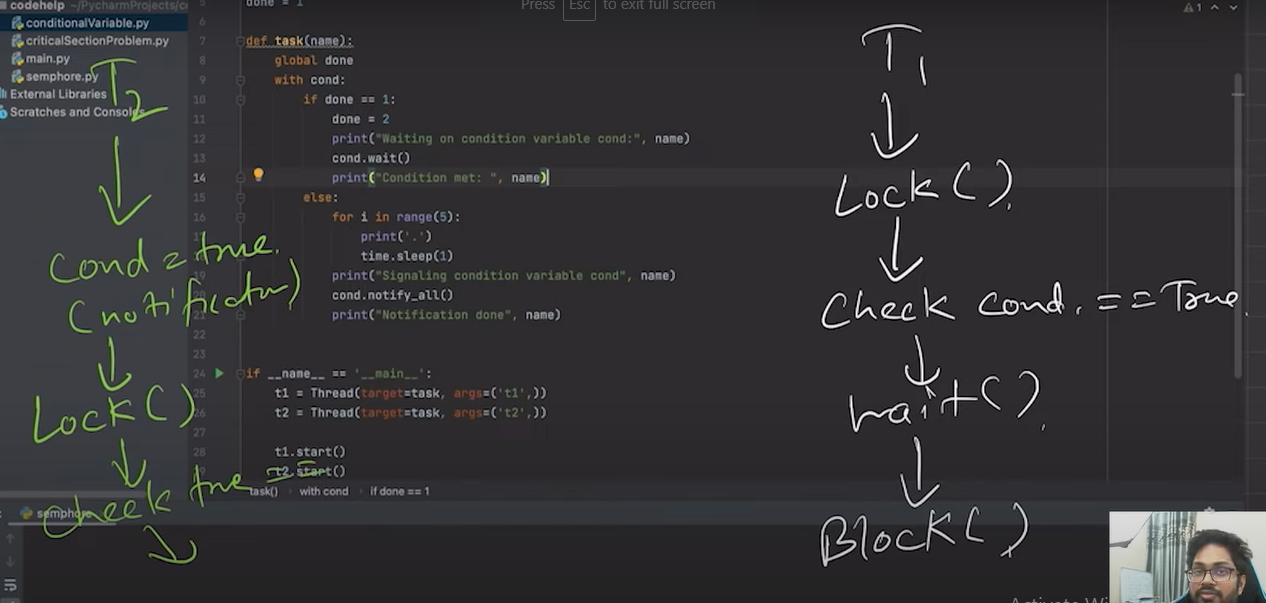
Single flag soln – No Progress

Peterson – 2 threads

Locks and mutex – Busy waiting problems

Improved soln for thread synchronisation

Conditional Variables



It avoids busy waiting. We use condition variable provided by languages sucha s python and cpp. It does not waste cpu cycles

a. The condition variable is a synchronization primitive that lets the thread wait until a certain condition occurs.

b. Works with a lock

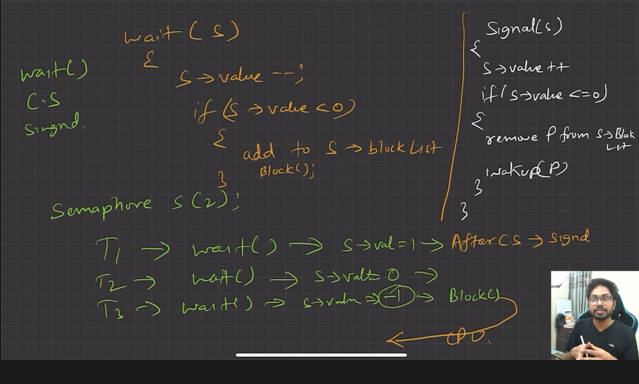
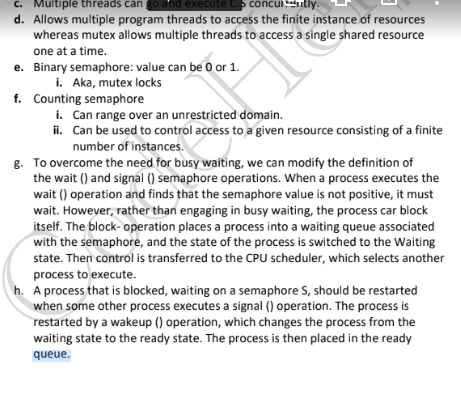
c. Thread can enter a wait state only when it has acquired a lock. When a thread enters the wait state, it will release the lock and wait until another thread notifies that the event has occurred. Once the waiting thread enters the running state, it again acquires the lock immediately and starts executing.

d. Why to use conditional variable?

i. To avoid busy waiting.

e. Contention is not here.

Semaphores:

1. Single Variable
2. Multiple instances
3. As memory has multiple instances, we maintain semaphore variable for accessing the memory by task.
4. If we have T0 – T10 task, and Semaphore is 3, T0 executes | Semphore = 2, T1 executes | Semphore = 1, T2 executes | Semphore = 0,
5. Now semaphore will go in wait state until any is free(Lock from critical section is free) .Let say T1 exits. Semaphore becomes 1.
6. Now T3 will execute making semaphore 0.
7. 
8. 
9. If semaphore = 1, binary semaphore
10. If s>1 counting semaphore