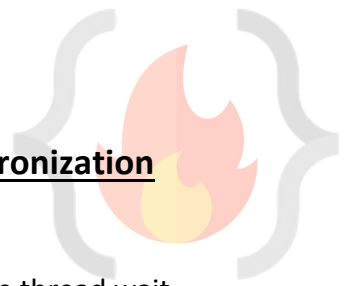


LEC-17: Conditional Variable and Semaphores for Threads synchronization



1. Conditional variable

- 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.

2. Semaphores

- a. Synchronization method.
- b. An integer that is equal to number of resources
- c. Multiple threads can go and execute C.S concurrently.
- d. Allows multiple program threads to access the finite instance of resources whereas mutex allows multiple threads to access a single shared resource one at a time.
- e. Binary semaphore: value can be 0 or 1.
 - i. Aka, mutex locks
- f. Counting semaphore
 - i. Can range over an unrestricted domain.
 - ii. Can be used to control access to a given resource consisting of a finite number of instances.
- g. To overcome the need for busy waiting, we can modify the definition of the wait () and signal () semaphore operations. When a process executes the wait () operation and finds that the semaphore value is not positive, it must wait. However, rather than engaging in busy waiting, the process can block itself. The block- operation places a process into a waiting queue associated with the semaphore, and the state of the process is switched to the Waiting state. Then control is transferred to the CPU scheduler, which selects another process to execute.
- h. A process that is blocked, waiting on a semaphore S, should be restarted when some other process executes a signal () operation. The process is restarted by a wakeup () operation, which changes the process from the waiting state to the ready state. The process is then placed in the ready queue.