**TEAM-09-OS-DEMO-4**

**Q1.**

* There are 2 conditions used:

1. static pthread\_cond\_t c1 = PTHREAD\_COND\_INITIALIZER;
2. static pthread\_cond\_t c2 = PTHREAD\_COND\_INITIALIZER;

* There are 2 condition varaiables used:

1. pthread\_cond\_wait();
2. pthread\_cond\_signal();

* Solution logic:

1. First thread is called using **void \*myThread1(void \*arg)**, mutex lock guarding the condition is acquired
2. Condition is tested
3. If the condition is false, the current thread is made to wait using **pthread\_cond\_wait(&c1);** and releases the mutex before blocking the thread and re-acquires the mutex before returning.
4. Then, the function **exec\_Thread1(tid1)** is executed and condition **c2** is signaled using **pthread\_cond\_signal(&c2);** which wakes next thread i.e. **void \*myThread2(void \*arg).**
5. Now, the next thread acquires the mutex lock guarding the condition, sets the condition, executes **exec\_Thread2(tid2)** and again signals the blocked thread with **pthread\_cond\_signal(&c1);** releases the mutex.
6. After the test allows the blocked thread(which is awake now), set the condition to its new value
7. Release the mutex.

**Q2.**

* There are 3 conditions used:

1. static pthread\_cond\_t c1 = PTHREAD\_COND\_INITIALIZER;
2. static pthread\_cond\_t c2 = PTHREAD\_COND\_INITIALIZER;
3. static pthread\_cond\_t c3 = PTHREAD\_COND\_INITIALIZER;

* There are 2 condition varaiables used:

1. pthread\_cond\_wait();
2. pthread\_cond\_signal();

* Solution logic:

1. First thread is called using **void \*myThread1(void \*arg)**, mutex lock guarding the condition is acquired
2. Condition is tested
3. If the condition is false, the current thread is made to wait using **pthread\_cond\_wait(&c1);** and releases the mutex before blocking the thread and re-acquires the mutex before returning.
4. Then, the function **exec\_Thread1(tid1)** is executed and condition **c2** is signaled using **pthread\_cond\_signal(&c2);** which wakes next thread i.e. **void \*myThread2(void \*arg).**
5. If the test condition again do not allow the thread to continue, then second thread is made to wait, e**xec\_Thread2(tid2)** function is called and next thread is signaled using **pthread\_cond\_signal(&c3);**
6. Now, the next thread acquires the mutex lock guarding the condition, sets the condition, calls the function **exec\_Thread(tid3)** and again signals the blocked thread with **pthread\_cond\_signal(&c1);** releases the mutex.
7. After the test allows the blocked thread(which is awake now), set the condition to its new value .
8. Release the mutex.

Advantage of this thread synchronization mechanism is that no thread can wake up any other thread accidentally.