Synchronization

What to Expect?

- Thread Synchronization Mechanisms
  - Mutex
  - Conditional Variables
  - Read/Write Locks
  - Spin Locks
  - Barriers
  - Semaphores
- Priority Inversion & its Solutions
- Understanding a Deadlock
- Inter Thread Communication

Recall

- Thread Management
  - Creation, Termination, Joining, Cleanup
  - Thread-specific Data
- What about the other data?
  - Shared: By virtue of existence
  - Communication: Inherent
    - No ITC mechanisms required
- Where is the catch?
  - Concurrency Issues & Race Conditions

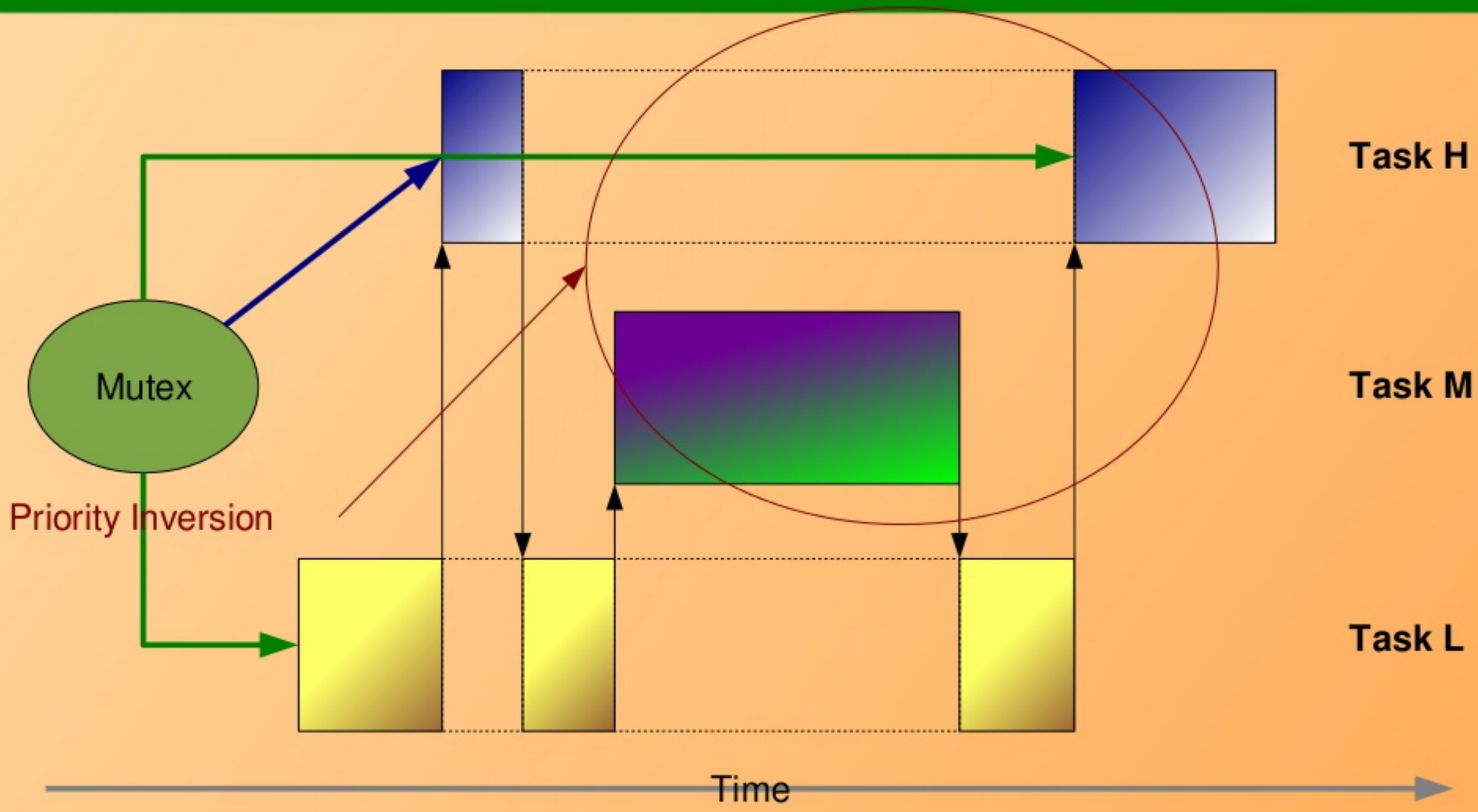
Solutions to Race Conditions

- Concurrency issues exist within a process
  - Do not need kernel resources to solve them
  - Just some internal synchronization & access protection mechanisms
- Implemented at pthreads library level
  - Mutex → Critical Sections
  - Condition Variables → Atomic Checks & Actions
  - Read/Write Locks → Readers & Writers Resources
  - Spin Locks → Critical Sections but without yielding
  - Barriers → Serializing Execution
  - Sempahores → Resource Usage

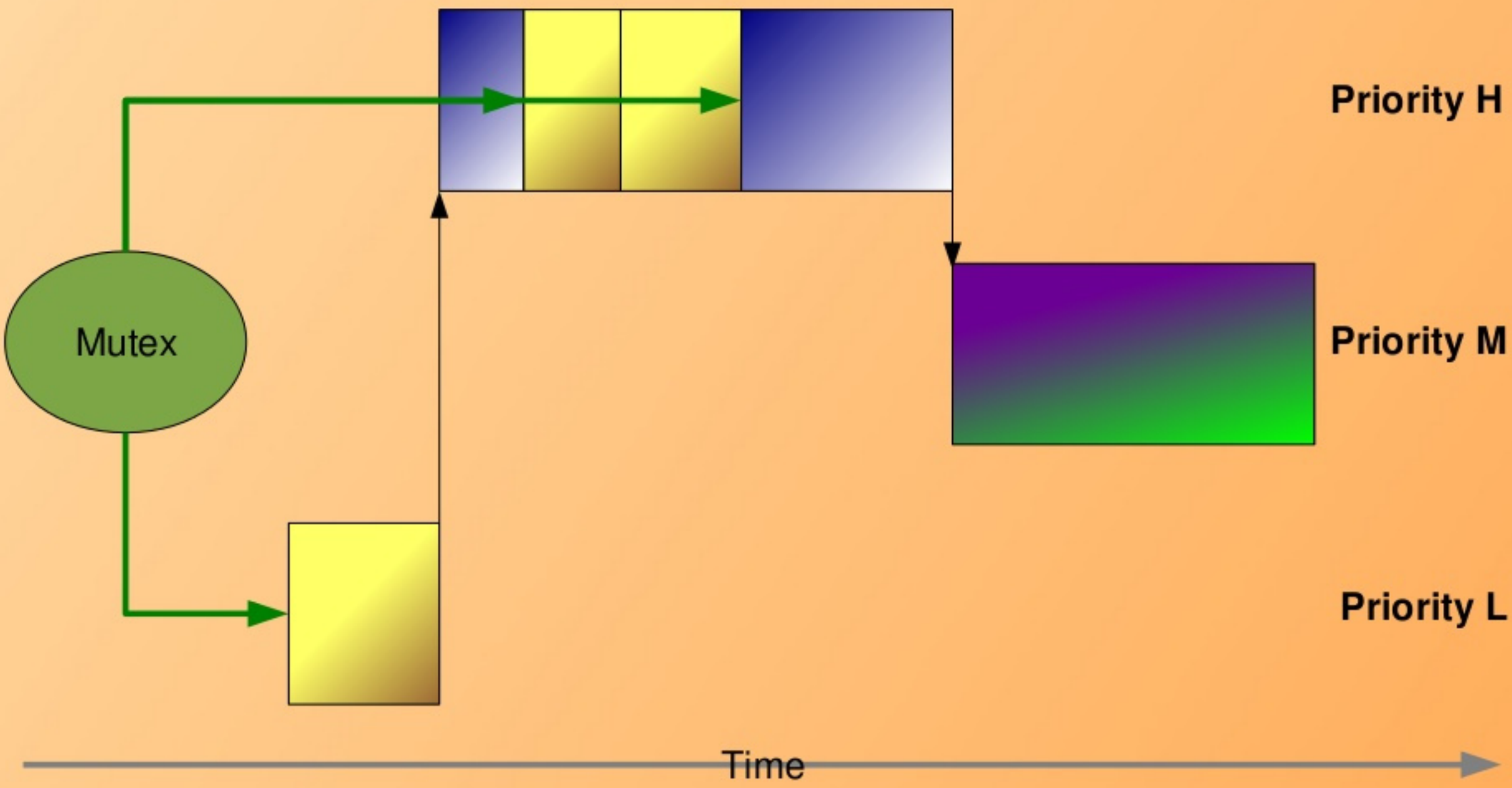
Mutual Exclusion

- Type: pthread_mutex_t
- APIs
  - int pthread_mutex_init(&mutex, &attr);
  - int pthread_mutex_destroy(&mutex);
  - int pthread_mutex_lock(&mutex);
  - int pthread_mutex_unlock(&mutex);
  - int pthread_mutex_trylock(&mutex);
- Macro
  - PTHREAD_MUTEX_INITIALIZER

Priority Inversion



Priority Inheritance



Mutual Exclusion Attributes

- Type: pthread_mutexattr_t
- APIs
  - pthread_mutexattr_init(&attr);
  - pthread_mutexattr_destroy(&attr);
  - pthread_mutexattr_settype(&attr, type);
    - PTHREAD_MUTEX_RECURSIVE / NORMAL / ERRORCHECK
  - pthread_mutexattr_setpshared(&attr, pshared);
    - PTHREAD_PROCESS_SHARED / PRIVATE
  - pthread_mutexattr_setprotocol(&attr, protocol); (Only for RT)
    - PTHREAD_PRIO_NONE, PTHREAD_PRIO_INHERIT, PTHREAD_PRIO_PROTECT
  - pthread_mutexattr_setprioceiling(&attr, proioceiling); (Only for RT)
- Try out: thread_mutex_attr.c, thread_mutex_error.c

Deadlock

- Set of entities waiting for each other
- Four necessary & sufficient conditions
  - Mutual Exclusion
  - Hold & Wait
  - Non-Preemptive
  - Circular Wait
- Have a look @ thread_deadlock.c

Dealing with Deadlock

- The Ostrich Approach
- Deadlock Detection & Recovery
  - Deadlock Algorithm: Current Resource Availability → Possible Allocation Sequences
  - Iterative Recovery Algorithm: $O(n^2)$
- Deadlock Avoidance
  - Banker's Algorithm
- Deadlock Prevention
  - Eliminate one of the four conditions

Condition Variables

- Type: pthread_cond_t
- APIs
  - int pthread_cond_init(&cond, &attr);
  - int pthread_cond_destroy(&cond);
  - int pthread_cond_wait(&cond, &mutex);
  - int pthread_cond_signal(&cond);
  - int pthread_cond_broadcast(&cond);
- Macro
  - PTHREAD_COND_INITIALIZER

Condition Variable Attributes

- Type: pthread_condattr_t
- APIs
  - pthread_condattr_init(&attr);
  - pthread_condattr_destroy(&attr);
  - pthread_condattr_setpshared(&attr, pshared);
    - PTHREAD_PROCESS_SHARED / PRIVATE
- Try out: thread_cond.c

Read / Write Locks

- Type: pthread_rwlock_t
- APIs
  - int pthread_rwlock_init(&rwlock, &attr);
  - int pthread_rwlock_destroy(&rwlock);
  - int pthread_rwlock_rdlock(&rwlock);
  - int pthread_rwlock_tryrdlock(&rwlock);
  - int pthread_rwlock_wrlock(&rwlock);
  - int pthread_rwlock_trywrlock(&rwlock);
  - int pthread_rwlock_unlock(&rwlock);
- Macro
  - PTHREAD_RWLOCK_INITIALIZER

Read / Write Lock Attributes

- Type: pthread_rwlockattr_t
- APIs
  - pthread_rwlockattr_init(&attr);
  - pthread_rwlockattr_destroy(&attr);
  - pthread_rwlockattr_setpshared(&attr, pshared);
    - PTHREAD_PROCESS_SHARED / PRIVATE
  - pthread_rwlockattr_setkind_np(&attr, kind);
    - PTHREAD_RWLOCK_PREFER_READER_NP / WRITER_NP / WRITER_NONRECURSIVE_NP
- Try out: thread_rwlock.c

Spin Locks

- Type: pthread_spinlock_t
- APIs
  - int pthread_spin_init(&spin, pshared);
  - int pthread_spin_destroy(&spin);
  - int pthread_spin_lock(&spin);
  - int pthread_spin_trylock(&spin);
  - int pthread_spin_unlock(&spin);
- Try out: thread_spinlock.c

Barriers & its Attributes

- Type: pthread_barrier_t
- APIs
  - int pthread_barrier_init(&barrier, &attr, cnt);
  - int pthread_barrier_destroy(&barrier);
  - int pthread_barrier_wait(&barrier);
- Attribute Type: pthread_barrierattr_t
- Attribute APIs
  - pthread_barrierattr_init(&attr);
  - pthread_barrierattr_destroy(&attr);
  - pthread_barrierattr_setpshared(&attr, pshared);
    - PTHREAD_PROCESS_SHARED / PRIVATE
- Try out: thread_barrier.c

Semaphores

- Type: sem_t
- APIs
  - int sem_init(sem_t *sem, int pshared, unsigned int value);
  - int sem_destroy(sem_t *sem);
  - int sem_post(sem_t *sem);
  - int sem_wait(sem_t *sem);
  - int sem_trywait(sem_t *sem);
- Try out: thread_sem.c

Exchange Offer

- Thread synchronizing mechanisms
  - Can be used for same ancestor processes
  - by setting there pshared attribute to non-zero
- Example: Semaphores for Processes
- Similarly, IPC mechanisms
  - Can be used by threads as well
- Let's see some examples

Inter Thread Communication

- Signals in Threads
- Pipe for Threads
- Memory Map Sharing for Threads
- Examples
  - thread_kill.c
  - thread_ipc.c
  - thread_ipc2.c

What all have we learnt?

- Thread Synchronization Mechanisms
  - Mutex
  - Conditional Variables
  - Read/Write Locks
  - Spin Locks
  - Barriers
  - Semaphores
- Priority Inversion & its Solutions
- Understanding a Deadlock
- Inter Thread Communication

Any Queries?