Synchronization: Three Techniques
For C-level synchronization, there are three constructs that we have available to help us synchronize access to critical sections:

Technique #1: _______________

**pthread_mutex_init**: Creates a new lock in the “unlocked” state.

**pthread_mutex_lock(pthread_mutex_t *mutex)**:
- When `mutex` is unlocked, change the lock to the “locked” state and advance to the next line of code.
- When `mutex` is locked, this function blocks execution until the lock can be acquired.

**pthread_mutex_unlock**: Moves the lock to the “unlocked” state.

**pthread_mutex_destory**: Destroys the lock; frees memory.

Technique #2: _______________

**pthread_cond_init**: Create a new conditional variable.

**pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex)**: Performs two different synchronization actions:
- 
- **pthread_cond_signal(pthread_cond_t *cond)**: Unblocks “at least one thread” that is blocked on `cond` (if any threads are blocked; otherwise an effective “NO OP”).

**pthread_cond_broadcast(pthread_cond_t *cond)**: Unblocks **ALL** threads blocked on `cond`.

**pthread_mutex_destory**: Destroys the lock; frees memory.

Q: What happens when we run this code now?...and the performance?
**Technique #3:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sem_init</td>
<td>Creates a new semaphore with a specified “value”.</td>
</tr>
<tr>
<td>sem_wait</td>
<td>When the value is greater than zero, decreases the value and continues.</td>
</tr>
<tr>
<td></td>
<td>Otherwise, <strong>blocks</strong> until the value is non-zero.</td>
</tr>
<tr>
<td>sem_post</td>
<td>Increments the value by one.</td>
</tr>
<tr>
<td>sem_destroy</td>
<td>Destroys the semaphore; frees memory.</td>
</tr>
</tbody>
</table>

**Critical Sections**

We know that critical sections require exclusive access to a resource. We also know locking a resource is computationally expensive. However, are there other concerns?

**The Dining Philosophers**

Imagine five philosophers and five chopsticks at a circular table. Each philosopher has two states: **eating** and **thinking**:

- When a philosopher is thinking, she holds no chopsticks.
- When a philosopher starts the process of eating, she must take the chopstick to her left, then her right, and then begin eating.

**Q:** Using the strategy described above (take left, take right, then eat), what happens over a long period of time?

**Deadlock:**

- **Definition:**

  - Four **necessary** conditions of deadlock:
    1) 
    2) 
    3) 
    4) 

**See Lecture Code:** 09/dinning-philosophers.c