What are Threads?
- A thread is a single stream of control in the flow of a program
- A program, at any given time, may consist of one or more threads each executing a sequence or stream of instructions.
- All threads have access to shared address space memory.
- In shared-memory parallel programming tasks are distributed among threads
  - Threads execute concurrently on multiple processor machines achieving speedups
  - Threads 'communicate' by accessing shared memory locations

POSIX Threads API
- Standard interface for multithreaded programming
- Widely supported and available (standard in most Unix and Windows)
- POSIX threads (or pthreads) provide library routines for creating, terminating, synchronizing, and managing threads
- #include <pthread.h>

Creating and Terminating Threads
- Creating a thread
  ```c
  int pthread_create(pthread_t *thread_handle, const pthread_attr_t *attr, void* (*thread_function)(void*), void *arg);
  ```
- Terminating a thread
  ```c
  int pthread_join(pthread_t thread, void **ptr);
  ```

Mutex Locks (1)
- Mutual-exclusion (mutex) locks
  ```c
  int pthread_mutex_init(pthread_mutex_t *mutex_lock, const pthread_mutexattr_t *lock_attr);
  int pthread_mutex_lock(pthread_mutex_t *mutex_lock);
  int pthread_mutex_unlock(pthread_mutex_t *mutex_lock);
  ```

Mutex Locks (2)
- Locks represent serialization points in a program
  - Performance degrades as computations within critical sections protected by mutex locks increase
- When multiple threads contend for a mutex lock, all are blocked except one
  - Use pthread_mutex_trylock(); if lock is not available function returns allowing thread to perform other tasks
    ```c
    int pthread_mutex_trylock(pthread_mutex_t *mutex_lock);
    ```
Types of Locks

- Three kinds of locks
  - Normal (default): can be locked once only
  - Recursive: can be locked multiple times by the same thread. Each time it is locked a counter is incremented. The counter is decremented for each unlock by the thread. Another thread obtain the mutex lock until the lock counter is zero.
  - Errorcheck: Similar to normal, except that it returns an error code instead of deadlocking when it attempts to lock again
- Type of lock is set using `pthread_mutexattr_settype_np`

Condition Variables

- A condition variable is a data object used for synchronizing threads
- A condition variable has a mutex associated with it

```c
int pthread_cond_init(pthread_cond_t *cond, pthread_condattr_t *attr);
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_destroy(pthread_cond_t *cond);
```