Thread Creation in PThreads

- Type: pthread_t tid; /* thread handle */
- pthread_create (&tid, thread_attr, start, arg)
  - tid returns pointer to created thread
  - thread_attr specifies attributes, e.g., stack size; use NULL for default attributes
  - start is procedure called to start execution of thread
  - arg is sole argument to proc
  - pthread_create returns 0 if thread created successfully
- pthread_join (tid, &retval);
  - Wait for thread tid to complete
  - Retval is valued returned by thread
- pthread_exit(retval)
  - Complete execution of thread, returning retval

Mutual Exclusion

- Bad things can happen when two threads "simultaneously" access shared data structures: Race condition → critical section problem
  - Data inconsistency!
  - These types of bugs are really nasty!
    - Program may not blow up, just produces wrong results
    - Bugs are not repeatable
- Associate a separate lock (mutex) variable with the shared data structure to ensure "one at a time access"

Mutual Exclusion in PThreads

- pthread_mutex_t mutex_var;
  - Declares mutex_var as a lock (mutex) variable
  - Holds one of two values: "locked" or "unlocked"
- pthread_mutex_lock (&mutex_var)
  - Waits/blocked until mutex_var in unlocked state
  - Sets mutex_var into locked state
- pthread_mutex_unlock (&mutex_var)
  - Sets mutex_var into unlocked state
  - If one or more threads are waiting on lock, will allow one thread to acquire lock

Example:

```
                                    //pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t   m;   //pthread_mutex_init(&m, NULL);
...
pthread_mutex_lock (&m);
<access shared variables>
pthread_mutex_unlock(&m);
```