Chapter12: Multithreaded Programming and multiprocessors

* Why multithreading?
  + If having multiple processors, they could be executing the program components in parallel.
* Programming support for threads
  + What we want **dynamically**:
    - Create threads
    - Terminate threads
    - Communicate among threads
    - Synchronize the activities of the threads
  + Threads creation and termination
    - The operating system creates a unique and distinct memory footprint for the program, called an address space.
    - This address space contains the code, global data, stack, and heap sections specific to each program
  + Communication among threads
    - Threads may need share data
    - The global data structures become shared data structures for the threads
  + Read-write, conflict, race condition, and nondeterminism
    - Read-write conflict: multiple concurrent threads simultaneously try to write to the shared variable.
    - Race condition: when a read-write conflict exists in a program without an intervening synchronization operation separating the conflict
    - An action to be **atomic**: **indivisible**
  + Synchronization among threads
    - How could a programmer expect a deterministic behavior?
    - Synchronization:
      * Mutual exclusion
        + The producer and consumer work concurrently, except when either or both have to modify or inspect shared data structures
        + Lock: the semantics shown here
        + Only one thread can hold a particular lock at a time
        + Once a thread acquires a lock, other threads cannot get that same lock until the first thread releases the lock
        + Critical section: code that is executed in such **mutually exclusive manner** is referred to as critical section
      * Rendezvous(汇合)
    - Condition variable: the semantics of another data abstraction commonly provided by the library
      * Need to recognize that the condition that the digitizer is waiting for
* Summary of thread functions calls and threaded programming concepts
* Points to remember in programming with threads
* Using threads as software structuring abstraction
* POSIX pthreads Library calls summary
  + IEEE has standardized the application programming interface for the threads with POSIX ***pthreads*** library
* OS support for threads
  + TCB: Thread control block
* Hardware support for multithreading in a uniprocessor
* Multiprocessors
* Advanced topics