Lecture 15 Summary

First part covered in lecture was pthreads. The definition of a thread is a single stream of control in the flow of a program. The basic is that threads provide software portability, inherent support for latency hiding, and ease the programming and widespread use (but debugging threaded program is a pain). The POSIX Thread API are commonly referred to as pthreads. There are two levels of threads: user-level threads and kernel-level threads, and user-level threads are bound to execute w/I a process or kernel-level thread.

Pthreads provides tow basic functions for specifying concurrency in a program: pthread\_create invokes function as a thread; pthread\_join suspends the calling thread until the thread specified in the argument terminates. A thread safe function is one that will allow for multiple instances to be active at the same instance in time and still operate correctly. Unsafe thread code has use of static variables and use of global vars. Therefore, it is important to make sure that the thread safe versions of all system call and user libs. Also, code can be thread safe by removing static vars to pure local/stack variables, letting each thread has its own local state for that function and wrapping global vars with proper mutual exclusion operations that form critical sections that serialize threads thru the execution of the piece of code (although it would hurt performance). There is usually a segment that must be executed by only one thread at any time, and this segment is called critical segment. Also, mutex locks are used with critical segments. Mutex locks have two states: locked and unlocked. Only one thread can lock a mutex lock at any point of time. Three functions are provided: pthread\_mutex\_lock, pthread\_mutex\_unlock and pthread\_mutex\_init. The producer-consumer scenario needs locks as well. There are three types of mutexes: normal, recursive and error-checking. The pthreads API allows a programmer to change the default attributes of entities using attributes objects, which would enhance modularity, readability and ease of modification. Pthreads provides: pthread\_attr\_init, pthread\_attr\_setdetachstate, pthread\_attr\_setstacksize, pthread\_attr\_setinheritsched, and pthread\_attr\_setschedpolicy. There are some tips: 1. Never rely on scheduling assumptions when exchanging data; 2. Never rely on liveness of data resulting from assumptions on scheduling; 3. Do not rely on scheduling as a means of synchronization; 4. Where possible, define and use group synchronizations and data replication.

Second part was about SGI.