RECITATION 11: CONCURRENCY

15-213 M12

Rick Benua

Multi-Threading

- Process can have multiple threads of execution
- Share address space
- Each thread has its own stack, local variables
  - No protection
  - Static local variables shared
- Global variables, heap shared between threads
- No hierarchy like with processes
  - Threads can create other threads, or wait for others to finish
  - No parent-child relationship

Concurrency - Critical Sections

- Portions of a program that are unsafe to execute simultaneously
  - e.g. Load from memory into register, change the value, store back to memory
  - Write a line of a log file
- Must be protected by some mutual exclusion or other concurrency primitive
  - Generally provided by the OS or thread library, with hardware help

Synchronization – Semaphores

- Important synchronization primitive
- Essentially a protected integer that cannot decrease below 0
- May be initialized to any positive value
- V increments value atomically
- P atomically waits for the value to be positive, then decrements it
  - These letters stand for something in Dutch
- Simulates a finite number of resources – claimed by P, released by V

Synchronization - Mutexes

- Mutual Exclusion – only one thread may be in a critical section at once
- Essentially a semaphore initialized to 1
  - May be implemented differently
- Lock – atomically waits for mutex to be available and acquires it
- Unlock – releases a held mutex

Synchronization – Reader/Writer Locks

- More complex lock type
  - Built on top of simpler primitives
- Useful for proxy!
- Allow multiple threads to read from the same resource simultaneously
- Ensure no readers while a (single) thread writes to the resource
- Taking the write lock ensures no other threads present
- Taking the read lock ensures no writers present

The pthreads library

- Thread library defined by POSIX, implemented by Linux
- User-level thread library possible, but slow
  - LinuxThreads is user-level, uses signals to do scheduling
  - NPTL enlists the help of the kernel – threads are scheduled the same way as processes
  - Both implement all pthreads functions
  - NPTL used by default
- To use, must pass -pthread to gcc and #include <pthread.h>

The pthreads library

- Thread lifetime functions
  - pthread_create() makes a new thread, calls a specified function
  - pthread_join() waits for a thread to terminate
  - pthread_detach() signals the system to clean up thread resources automatically – if not detached, thread resources cleaned up by pthread_join() (**not** necessarily by the thread that created it)
- Synchronization functions
  - Semaphores (sem_init(), sem_wait(), sem_post())
  - Mutexes (pthread_mutex_init(), pthread_mutex_lock(), etc)
    - pthread_mutex_trylock() does not block if unavailable
  - Reader/Writer Locks (pthread_rwlock_init(), pthread_rwlock_rdlock(), pthread_rwlock_wrlock(), etc)