CSCI-GA.3033-017

Special Topics:

Multicore Programming

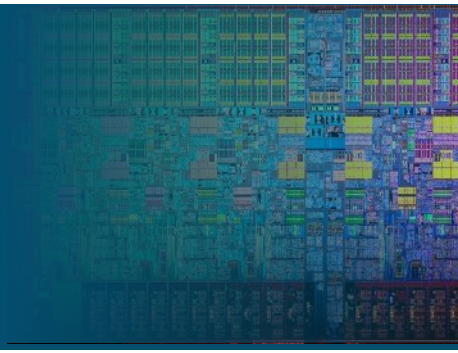
Lecture 4-6 Appendix A

C vs C++ Multicore Programming

Christopher Mitchell, Ph.D.

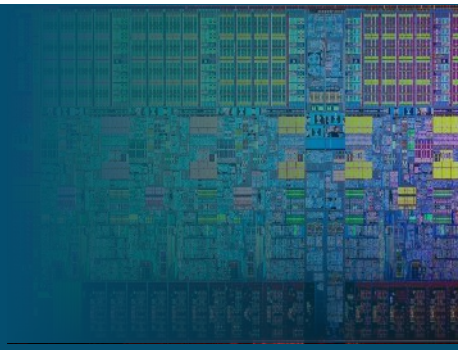
cmitchell@cs.nyu.edu || http://z80.me

Libraries



- C and C++: Pthreads
- C++: <thread> and friends (Introduced in C++11)
  - Limitations: no reader-writer lock, no semaphores, no barriers.
  - Pros: Extra primitives, like `recursive_mutex`, `shared_mutex`, `timed_mutex`, scoped `lock_guard`, etc.
- C++: Boost Thread (Discouraged)

Thread Management

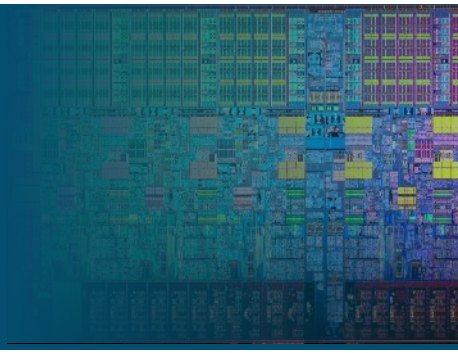


Function	Pthreads <pthread.h>	C++ <thread>
Create thread t	`pthread_create(&t)`	`std::thread t(f, args);`
Join thread t (blocking)	`pthread_join(t, &rval)`	`t.join();`
Detach thread t (nonblocking)	`pthread_detach(t)`	`t.detach();`
Exit current thread	`pthread_exit()`	*n/a*
Get current thread ID	`pthread_self()`	`this_thread::get_id()`

http://en.cppreference.com/w/cpp/thread

http://en.cppreference.com/w/cpp/thread/thread

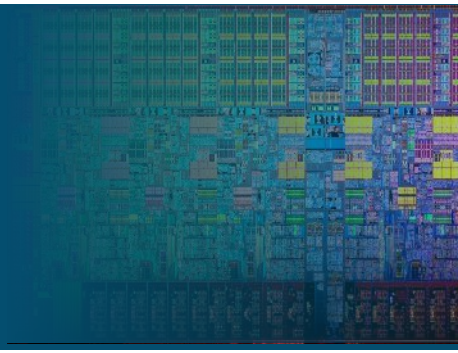
Mutex/Mutices



Function	Pthreads <pthread.h>	C++ <mutex>
Create mutex m	`pthread_mutex_init(&m)`	`std::mutex m;`
Lock mutex m	`pthread_mutex_lock(&m);`	`m.lock();`
Lock mutex m (nonblocking)	`pthread_mutex_trylock(&m);`	`m.trylock();`
Unlock mutex m	`pthread_mutex_unlock(&m);`	`m.unlock();`

http://en.cppreference.com/w/cpp/thread/mutex

Condition Variables



Function	Pthreads <pthread.h>	C++ <condition_variable>
Create condition variable c	`pthread_cond_init(&c);` `pthread_mutex_init(&m);`	`std::condition_variable cv;` `std::mutex cv_m;`

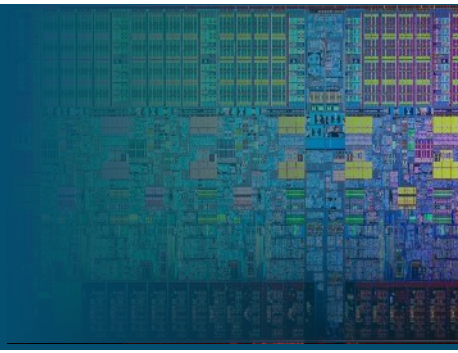
Wait for condition variable	`pthread_cond_wait(&c, &m);`	`std::unique_lock<std::mutex>`
		`  lock(cv_m);`
		`cv.wait(lk);`

Wait for condition variable with predicate	*n/a*	`std::unique_lock<std::mutex>`
		`  lock(cv_m);`
		`cv.wait(lk, []{return …});`

Signal single waiter	`pthread_cond_signal(&c);`	`cv.notify_one();`
Signal all waiters	`pthread_cond_broadcast(&c);`	`cv.notify_all();`

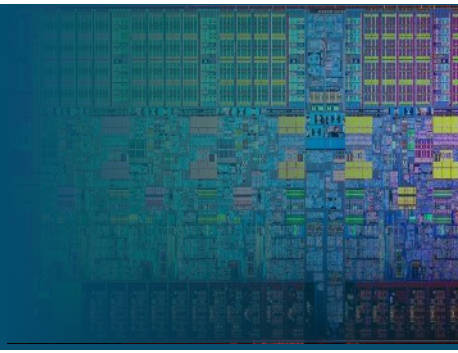
http://en.cppreference.com/w/cpp/thread/condition_variable

Semaphores



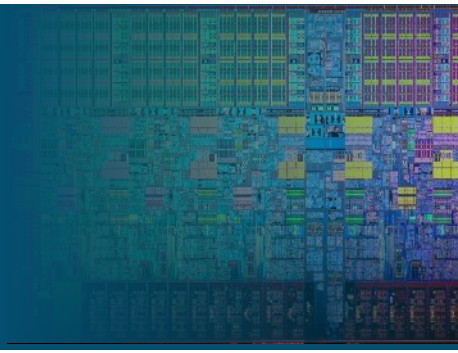
Function	Pthreads <semaphore.h>	C++
Initialize semaphore s	`sem_init(&s, shared, val)`	*n/a*
Decrement semaphore	`sem_wait(&s)`	*n/a*
Increment semaphore	`sem_post(&s)`	*n/a*

Reader-Writer Locks



Function	Pthreads <pthread.h>	C++
Initialize R-W lock rwl	`pthread_rwlock_init(&rwl);`	*n/a*
Shared (read) lock rwl	`pthread_rwlock_rdlock(&rwl);`	*n/a*
Exclusive (write) lock rwl	`pthread_rwlock_wrlock(&rwl);`	*n/a*
Unlock rwl	`pthread_rwlock_unlock(&rwl);`	*n/a*
Shared (read) lock rwl without blocking	`pthread_rwlock_tryrdlock(&rwl);`	*n/a*
Exclusive (write) lock rwl without blocking	`pthread_rwlock_trywrlock(&rwl);`	*n/a*

Barriers



Function	Pthreads <pthread.h>	C++
Create barrier b	`pthread_barrier_init(&b, attrs, count)`	*n/a*
Wait at barrier b	`Pthread_barrier_wait(&b);`	*n/a*