

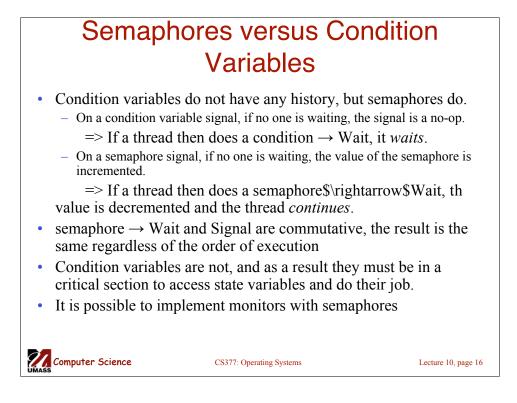
Monitors in C++: Example

<pre>class Queue { public: Add(); Remove(); private Lock lock; // queue data(); }</pre>	Queue::Add() { lock->Acquire(); // lock before usin put item on queue; // ok to access sha conditionVar->Signal(); lock->Release(); // unlock after acc } Queue::Remove() { lock->Acquire(); // lock before usin while queue is empty conditionVar->Wait(lock); // release remove item from queue; lock->Release(); // unlock after acc return item; }	ared data cess ng data e lock & sleep
Computer Science	CS377: Operating Systems	Lecture 10, page 13

Bounded Buffer using Hoare-style condition variables

<pre>class BBMonitor { public: void Append(item); void Remove(item); private: item buffer[N]; int last, count; Condition full, empty; } BBMonitor::BBMonitor { count = 0; last = 0; }</pre>	<pre>BBMonitor::Append(item) { lock->Acquire(); if (count == N) empty->Wait(lock); buffer[last] = item; last = (last + 1) mod N; count += 1; full->Signal(); lock->Release(); } BBMonitor::Remove(item) { lock->Acquire(); if (count == 0) full->Wait(lock); item = buffer[(last-count) mod N]; count = count-1; empty->Signal(); lock->Release(); } </pre>	
Computer Science	<pre>} CS377: Operating Systems</pre>	Lecture 10, page 14

Semapho	ores versus M	onitors
atomic operations and qu condition->Wait() { condition->Signal() {	{ semaphore->signal(); } only work inside a lock. If we use	?
• How about this?		
<pre>condition->Wait(Lock *lock) { lock->Release(); semaphore->wait(); lock->Acquire(); } condition->Signal() { semaphore->signal(); }</pre>		
Computer Science	CS377: Operating Systems	Lecture 10, page 15



Implementing Monitors with				
Semaphores				
class Monitor {				
public:				
void ConditionWait(); // Condition Wait				
void ConditionSignal(); // Condition Signal				
private:				
<shared data="">; // data being protected by monitor</shared>				
semaphore cvar; // suspends a thread on a wait				
int waiters; // number of threads waiting on				
// a cvar (one for every condition)				
semaphore lock; // controls entry to monitor				
semaphore next; // suspends this thread when signaling another				
int nextCount; // number of threads suspended				
} // on next				
Monitor::Monitor {				
cvar = 0; // Nobody waiting on condition variable				
lock = FREE; // Nobody in the monitor				
next = nextCount = waiters = 0;				
CS377: Operating Systems	Lecture 10, page 17			

