CSCE 313-200
Introduction to Computer Systems
Spring 2024

Synchronization V

Dmitri Loguinov
Texas A&M University

February 21, 2024

Chapter 5: Roadmap

5.1 Concurrency

5.2 Hardware mutex

5.3 Semaphores

5.4 Monitors

5.5 Messages

5.6 Reader-Writer

Mutex

- Windows kernel mutex has semantics close to a binary semaphore 2.0, with two exceptions:
  - Repeated mutex lock from the same thread does not block it
  - Mutex can only be unlocked by the thread that locked it
- Examples:

```
Semaphore semaX = {1, 1}; // (s,max)
Thread () {
        semaX.Wait();     // P
        semaX.Wait();     // P
}
```

deadlocks because it attempts to decrement s twice

```
Mutex m;          // unlocked
Thread () {
        m.Lock();
        m.Lock();
}
```

works fine as this thread already owns the mutex

Mutex

- Examples (cont'd):

thread$_1$ deadlocks if thread$_2$ runs first; how to fix this?

```
Semaphore semaX = {1, 1}; // (s,max)
Thread1 () {
        semaX.Wait();     // P
        semaX.Wait();     // P
}
```

```
Semaphore semaX = {1, 1}; // (s,max)
Thread2 () {
        // some initialization
        semaX.Release();   // V
}
```

thread$_1$ blocks temporarily, then gets unblocked by thread$_2$

```
Mutex m;
Thread1 () {
        m.Unlock(); // does nothing
}
```

```
Mutex m;      // initially unlocked
Thread2 () { // thread2 runs first
        m.Lock();
        // long critical section
}
```

thread$_1$ fails to unlock mutex owned by thread$_2$

Event

```
class Event {
        int     s;      // state
        int     mode;
        List    blocked;
        Wait (); Set (); Reset ();
}
```

- The last standard synchronization primitive is an event
  - An event can be in two states: signaled (1) and non-signaled (0) just like a binary semaphore
- However, it also has two possible modes of operation
  - AUTO = binary semaphore
  - MANUAL = event stays signaled until manually reset

```
Event::Wait() {
    if (s == NOT_SIGNALED)
        // block current thread
    else if (mode == AUTO)
        s = NOT_SIGNALED;
}
```

```
Event::Reset() {
        s = NOT_SIGNALED;
}
```

```
Event::Set() {
    if (blocked.size() > 0)
        if (mode == AUTO)
            // unblock 1 thread
        else
            // unblock all threads
            s = SIGNALED;
    else
        s = SIGNALED;
}
```

Windows APIs

```
HANDLE WINAPI CreateSemaphore(
  __in_opt  LPSECURITY_ATTRIBUTES
                lpSemaphoreAttributes,
  __in      LONG lInitialCount,
  __in      LONG lMaximumCount,
  __in_opt  LPCTSTR lpName );
```

- Semaphore
  - Security is NULL as always
  - Name can be used when multiple processes need to open the same object
- Wait (i.e., P)
  - WaitForSingleObject()
  - Returns WAIT_OBJECT_0 when ready
  - WAIT_TIMEOUT if timeout
  - Otherwise, an error
- Release (i.e., V)
  - ReleaseSemaphore(N)
- CreateMutex/CreateEvent
  - Can specify if this thread initially owns the mutex and initial state for event
- Locking done with WaitForSingleObject()
  - Unlocking with ReleaseMutex() and signaling with SetEvent()
- Resetting events
  - ResetEvent()