Monitors
Monitor --- Class

• Monitor == data + operation + synchronization
The history of synchronization

- System level resource-access synchronization

- Application level synchronization

- THE, Nucleus, Hoare Monitors, ...
Two basic types of synchronization

• Mutual exclusion
  • What it is ...
  • Example
  • How supported by Monitor

• Ordering
  • What it is
  • How supported by Monitor

```
class Counter {
  private int value;

  public synchronized void increment() {
    value++;
    System.out.println(value);
  }

  public synchronized void decrement() {
    value--;
    System.out.println(value);
  }
}
```
Two basic types of synchronization

• Mutual exclusion
  • What it is
  • How supported by Monitor

• Ordering
  • What it is
  • How supported by Monitor

been completed. Otherwise, if two procedure bodies were in simultaneous execution, the effects on the local variables of the monitor could be chaotic. The proce
Two basic types of synchronization

- Mutual exclusion
  - What it is
  - How supported by Monitor

- Ordering
  - What it is
  - Example
  - How supported by Monitor

---

been completed. Otherwise, if two procedure bodies were in simultaneous execution, the effects on the local variables of the monitor could be chaotic. The proce-

Any dynamic resource allocator will sometimes need to delay a program wishing to acquire a resource which is not currently available, and to resume that program after some other program has released the resource.
Semantics of monitor procedures
Semantics of monitor wait & signal
Synchronization types

Be careful about Global variables & Heap variables In multi-threaded programs!

ptr = NULL;
...
ptr = malloc(sizeof(int)); *ptr = 10;

Thread 1
x++;
if (ptr){
 *ptr = 10;
 ptr = NULL;
}

Thread 2
x++;
How to enforce mutual exclusion?

Thread 1
x++;          x++;
if (ptr){
    *ptr=10;
} ptr=NULL;

Thread 2
x++;
How to enforce ordering?

Thread 1

```
ptr = NULL;
...
ptr = malloc(sizeof(int));
```

Thread 2

```
*ptr = 10;
```

- wait
- signal
- wait
- signal
How to synchronize: Example 1

Thread 1
ptr=NULL;
...
ptr=malloc(sizeof(int));
signal (&CV)

Thread 2

wait (&CV)
*ptr=10;

Is this correct?
How to synchronize: Example 1

What do we need?
1. Tracking if signal already occurred

Thread 1

ptr=NULL;
...
ptr=malloc(sizeof(int));

signal (&CV)

Thread 2

wait (&CV)

*ptr=10;
How to synchronize: Example 1

Thread 1

ptr=NULL;

Thread 2

ptr=malloc(sizeof(int));
*ptr=10;

signal (&CV)

wait (&CV)

What do we need?
1. Tracking if signal already occurred

Thread 3

ptr=NULL;

Is this correct?
How to synchronize: Example 1

Thread 1

ptr=NULL;
...
ptr=malloc(sizeof(int)); *ptr=10;
signal (&CV);

Thread 2

if (ptr==NULL)
  wait (&CV)

What do we need?
1. Tracking if signal already occurred
2. A lock!
How to synchronize: Example 1

What do we need?
1. Tracking if signal already occurred
2. A lock!

Thread 1

ptr=NULL;
Lock(&L);
ptr=malloc(sizeof(int)); *ptr=10;
signal (&CV);
Unlock (&L);

Thread 2

Lock(&L);
if (ptr==NULL) wait (&CV) ○

Unlock(&L);
Hoare monitor vs. Mesa monitor

Thread 1

ptr=NULL;
Lock(&L);
ptr=malloc(sizeof(int));

Thread 2

Lock(&L);
if (ptr==NULL)
  wait (&CV)
*ptr=10;
Unlock(&L);
Mesa monitor and its implementation

ptr=NULL;
...
ptr=malloc(sizeof(int));
*ptr=10;
signal(&CV)

while (ptr==NULL)
    wait (&CV)
Today’s ...
Example 2: producer – consumer queue

Producer

```plaintext
Item=produce();
Q[++]=Item;
```

Consumer

```plaintext
while (i<max)
    wait

Item = Q[i--];
Consume (Item);
```

What problems?
Mutual exclusion?
Ordering?
Example 2: producer – consumer queue

Producer

Item = produce();
Lock (&L);
Q[++] = Item;
signal(&CV1);
Unlock(&L);

Consumer

Lock (&L);
While(...)
wait(&CV1);
Item = Q[i--];
Unlock(&L);
Consume (Item);

What problems?
Mutual exclusion?
Ordering?
Example 2: producer – consumer queue

**Producer**

```c
Item = produce();
{
    Lock (&L);
    While(…)
        wait(&CV2);
        Q[++] = Item;
    signal(&CV1);
    Unlock(&L);
}
```

**Consumer**

```c
{
    Lock(&L);
    While(…)
        wait(&CV1);
        Item = Q[i--];
        signal(&CV2);
    Unlock(&L);
    Consume(Item);
}
```

What problems?
- Mutual exclusion?
- Ordering?
- Deadlock?
Other types of synchronization
# Synchronization primitives

<table>
<thead>
<tr>
<th></th>
<th>Mutual Exclusion</th>
<th>Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C (pthread library)</strong></td>
<td>pthread_mutex_lock</td>
<td>pthread_cond_signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pthread_cond_wait</td>
</tr>
<tr>
<td><strong>Java (monitor)</strong></td>
<td>synchronized methods</td>
<td>wait, notify</td>
</tr>
<tr>
<td><strong>semaphore</strong></td>
<td>sem_post</td>
<td>sem_wait</td>
</tr>
<tr>
<td></td>
<td>sem_wait</td>
<td></td>
</tr>
</tbody>
</table>

```
Thread 1          Thread 2
x++;      x++;      if (ptr){
    *ptr=10;      ptr=NULL;
}  
ptr=NULL;
...
ptr=malloc(sizeof(int));    *ptr=10;
free (ptr);                   *ptr=10;
```
Our 4/15 lecture ends here
Semantics & mechanisms of semaphore

- Semaphore{
  int cnt;
  Queue blocked_q;
}

- Sem_init

- Sem_post
  - Cnt ++
  - Wake a waiting queue up, if any

- Sem_wait
  - If cnt>0
    - Cnt- -
  - Else
    - Wait in blocked_q
    - Cnt--
How to enforce mutual exclusion?

Sem_init(??)

X+ +;

X+ +;

X+ +;
Public semaphore

Sem_init(??)

Sem_wait

X+ +;

Sem_post

Sem_wait

X+ +;

Sem_post
Private semaphore: how to make B waits for A

Sem_init(??)
Semaphore vs. Condition variable

**Semaphore**
- Has a counter

**Condition variable**
- No counter
Semaphore vs. Condition variable

**Semaphore**
- Has a counter
- Sem_post
  - What happens when no one is waiting?

**Condition variable**
- No counter
- Signal
  - What happens when no one is waiting?
Semaphore vs. Condition variable

Semaphore
• Has a counter
• Sem_post
  • What happens when no one is waiting?
• Sem_wait
  • Work by itself
  • No lock associated

Condition variable
• No counter
• Signal
  • What happens when no one is waiting?
• Wait
  • Need a while(flag)
  • Need a lock
  • Release lock while waiting
Example 1

Semaphore

Thread 1
sem_init(&s, 0);

ptr=NULL;
...
ptr=malloc(sizeof(int));  *ptr=10;

sem_wait(&s);

sem_post(&s);

Thread 2

Condition variable

Thread 1
Lock(&L);

ptr=NULL;
...
ptr=malloc(sizeof(int));  *ptr=10;

sem_wait(&s);

signal (&CV);
Unlock (&L);

Thread 2
Lock(&L);

if (ptr==NULL)
wait (&CV)
Unlock(&L);

Lock (&L);
Unlock (&L);
Example 2 (producer—consumer queue)

• Producer
• Consumer

What problems?
Mutual exclusion?
Ordering?
Deadlock?
Example 2 (producer—consumer queue)

• Producer

\[
\text{Item} = \text{produce()}; \\
\text{Lock} (\&\text{L}); \\
\text{While}(\ldots) \\
\quad \text{wait}(\&\text{CV2}); \\
\text{Q}[++i] = \text{Item}; \\
\text{signal}(\&\text{CV1}); \\
\text{Unlock}(\&\text{L});
\]

• Consumer

\[
\text{Lock}(\&\text{L}); \\
\text{While}(\ldots) \\
\quad \text{wait}(\&\text{CV1}); \\
\text{Item} = \text{Q}[i-]; \\
\text{signal}(\&\text{CV2}); \\
\text{Unlock}(\&\text{L}); \\
\text{Consume} (\text{Item});
\]

What problems?
Mutual exclusion?
Ordering?
Deadlock?
Example 2 (producer—consumer queue)

• Producer

```
Sem_init(&cell, MAX); sem_init(&item, 0);
Item=produce();
Sem_wait(&cell);
Sem_wait(&mutex);
Q[++i]=Item;
Sem_post(&mutex);
Sem_post(&item);
```

• Consumer

```
Sem_wait(&item);
Sem_wait(&mutex);
Item = Q[i--];
Sem_signal(&mutex);
Sem_signal(&cell);
```

What problems?
Mutual exclusion?
Ordering?
Deadlock?
What is used today?
Other issues raised by Mesa monitor paper

• Deadlocks