CS 360 Internet Programming
Concurrent Programming
Shared Memory Synchronization with Semaphores

Daniel Zappala
Computer Science
Brigham Young University
1. Shared Memory Synchronization with Semaphores
   - Definitions and Types
   - POSIX Semaphores
   - Binary Semaphores
   - Counting Semaphores
Semaphores

- **semaphore** is a shared integer variable
  - initialized \( \geq 0 \)
- **wait(s)**: wait for a signal on semaphore \( s \)
  - decrements semaphore, blocks if \(< 0\)
  - process suspends until signal is sent
  - OS uses a queue to hold waiting processes
- **signal(s)**: transmit a signal to semaphore \( s \)
  - increments semaphore
  - if \( \leq 0 \) then unblock someone
- **wait()** and **signal()** are atomic operations and cannot be interrupted
OS Definitions

```c
1 struct semaphore {
2     int count;
3     queueType queue;
4 } 
5 void wait(semaphore s) {
6     s.count--;
7     if (s.count < 0) {
8         place this process in s.queue;
9         block this process;
10     }
11 }
12 void signal(semaphore s) {
13     s.count++; 
14     if (s.count <= 0) {
15         remove a process P from s.queue;
16         place process P on ready list;
17     }
18 }
```
Types of Semaphores

- **binary semaphore**
  - only one process at a time may be in the critical section

- **counting semaphore**
  - a fixed number of processes $> 0$ may be in the critical section

- OS determines whether processes are released from queue in FIFO order or otherwise; usually FIFO in order to prevent starvation
Using Semaphores

1. semaphore \( s = 1; \)
2. \[
\begin{align*}
&\text{void thread(int i) } \\
&\quad \text{while (true) } \\
&\quad \quad \text{wait}(s); \\
&\quad \quad /* \text{critical section } */ \\
&\quad \quad \text{signal}(s); \\
&\quad \quad /* \text{remainder } */ \\
&\quad \}
\end{align*}
\]

- semaphore protects critical section
- can set \( s \) to \( > 1 \) to let more than one process in the critical section
  - \( s \geq 0 \) : number that can enter
  - \( s < 0 \) : number that are waiting
POSIX Semaphores

1. `#include <semaphore.h>`
2.
3. `int sem_init(sem_t *sem, int pshared, unsigned int value);`
4. `int sem_wait(sem_t *sem);`
5. `int sem_trywait(sem_t *sem);`
6. `int sem_post(sem_t *sem);`

- `sem_init()`: sets initial value of semaphore; `pshared = 0` indicates semaphore is local to the process
- `sem_wait()`: suspends process until semaphore is $> 0$, then decrements semaphore
- `sem_trywait()`: returns EAGAIN if semaphore count is $= 0$
- `sem_post()`: increments semaphore, may cause another thread to wake from `sem_wait()`
- **demonstration**
Producer Consumer Problem

- one or more producers are generating data and placing them in a buffer
- one or more consumers are taking items out of the buffer
- only one producer or consumer may access the buffer at any time
Producer Consumer with Infinite Buffer

**producer:**

1 while (True) {
2   /* produce item v */
3       b[in] = v;
4       in++;
5 }

**consumer:**

1 while (True) {
2       while (in <= out)
3           /* do nothing */;
4       w = b[out];
5       out++;
6       /* consume item w */
7 }
Shared Memory Synchronization with Semaphores

Definitions and Types
POSIX Semaphores
Binary Semaphores
Counting Semaphores

Producer Consumer using Binary Sempahores

```c
int n;
sem_t s, delay;
sem_init(&s, 0, 1);
sem_init(&delay, 0, 0);

producer:
while (True) {
   produce();
   sem_wait(&s);
   append();
n++; 
if (n==1) sem_post(&delay);
sem_post(&s);
}

cconsumer:
int m;
sem_wait(&delay);
while (True) {
   sem_wait(&s);
take();
n--; 
m = n;
sem_post(&s);
consume();
if (m==0) sem_wait(&delay)
}
```

Daniel Zappala
Computer Science
Brigham Young University

CS 360 Internet Programming
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore delay?
3. Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?
4. Why does the consumer need to wait on delay first?
5. Why does the producer signal wait only when n == 1?
6. Why does the consumer need a local variable m?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore delay?
3. Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?
4. Why does the consumer need to wait on delay first?
5. Why does the producer signal wait only when n == 1?
6. Why does the consumer need a local variable m?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore delay?
3. Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?
4. Why does the consumer need to wait on delay first?
5. Why does the producer signal wait only when n == 1?
6. Why does the consumer need a local variable m?
What is the purpose of semaphore s?

What is the purpose of semaphore delay?

Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?

Why does the consumer need to wait on delay first?

Why does the producer signal wait only when n == 1?

Why does the consumer need a local variable m?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore delay?
3. Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?
4. Why does the consumer need to wait on delay first?
5. Why does the producer signal wait only when n == 1?
6. Why does the consumer need a local variable m?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore delay?
3. Why is semaphore s initialized to 1 but semaphore delay is initialized to 0?
4. Why does the consumer need to wait on delay first?
5. Why does the producer signal wait only when n == 1?
6. Why does the consumer need a local variable m?
Important Insights

- **two purposes for semaphores**
  - **mutual exclusion**: semaphore `s` controls access to critical section
  - **signalling**: semaphore `delay` coordinates when the buffer is empty: consumer waits if buffer is empty, producer signals when buffer becomes non-empty

- **avoid race conditions**
  - `m` keeps a local copy of the data protected by the semaphore so that it can be accessed later
  - reduces amount of processing inside the critical section
Shared Memory Synchronization with Semaphores

Definitions and Types
POSIX Semaphores
Binary Semaphores
Counting Semaphores

Producer Consumer using Counting Semaphores

```c
1 sem_t s, n;
2 sem_init(&s, 0, 1);
3 sem_init(&n, 0, 0);

producer:
1 while (True) {
2   produce();
3   sem_wait(&s);
4   append();
5   sem_post(&s);
6   sem_post(&n);
7 }

consumer:
1 while (True) {
2   sem_wait(&n);
3   sem_wait(&s);
4   take();
5   sem_post(&s);
6   consume();
7 }
```

Daniel Zappala
Computer Science
Brigham Young University

CS 360 Internet Programming
Looking at the Code …

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Looking at the Code ...

1. What is the purpose of semaphore s?
2. What is the purpose of semaphore n?
3. Why is semaphore s initialized to 1 but semaphore n is initialized to 0?
4. Why can the producer signal n every time an item is added to the buffer?
5. Can the producer swap the signals for n and s?
6. Can the consumer swap the waits for n and s?
Important Insights

- more elegant solution, can’t do this easily with mutexes
- \( n \): semaphore value is number of items in buffer
  - if \( n == 0 \), consumer must wait
  - can swap `sem_post(&n);` and `sem_post(&s);` in producer and be OK
  - can’t swap `sem_wait(&n);` and `sem_wait(&s);` in consumer: otherwise consumer enters and then waits and deadlocks the producer!

- ordering of semaphore operations is important
- would like programming language support to help with organizing mutual exclusion code: monitors
Shared Memory Synchronization with Semaphores

Definitions and Types
- POSIX Semaphores
- Binary Semaphores
- Counting Semaphores

Producer Consumer Circular Buffer

```c
sem_t s, n, e;
sem_init(&s, 0, 1);
sem_init(&n, 0, 0);
sem_init(&e, 0, BUFFER_SIZE);

producer:
while (True) {
    produce();
    sem_wait(&e);
    sem_wait(&s);
    append();
    sem_post(&s);
    sem_post(&n);
}

counter: 
while (True) {
    sem_wait(&n);
    sem_wait(&s);
    take();
    sem_post(&s);
    sem_post(&e);
    consume();
}
```

Daniel Zappala
Computer Science
Brigham Young University

CS 360 Internet Programming
Looking at the Code ...

1. *What is the difference between semaphore *e* and semaphore *n*?