Semaphores and Bounded Buffer

Sarah Diesburg

Operating Systems

CS 3430

Semaphores

- ***Semaphore*** is a type of generalized lock
  - Consist of a nonnegative integer value
  - Two operations
    - ***P()***: an atomic operation that waits for semaphore to become positive, then decrement it by 1
    - ***V()***: an atomic operation that increments semaphore by 1 and wakes up a waiting thread at P(), if any.

Origin of Semaphores

- Defined by Dijkstra in the 60s
- Main synchronization primitives used in UNIX
- The P operation is an abbreviation for ***proberen*** (Dutch), meaning “to test”
- The V operation stands for ***verhogen***, meaning “to increment”

Semaphores vs. Integers

- No negative values
- Only operations are P() and V()
  - Cannot read or write semaphore values
    - (Except at the initialization times)
- Operations are atomic
  - Two P() calls cannot decrement the value below zero
  - A sleeping thread at P() cannot miss a wakeup from V()

Binary Semaphores

- A ***binary semaphore*** is initialized to 1
- P() waits until the value is 1
  - Then set it to 0
- V() *sets* the value to 1
  - Wakes up a thread waiting at P(), if any

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

litter_box = 1

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box); // purrr…
// critical section
V(litter_box);
```

litter_box = 1 → 0

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

litter_box = 0

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box); // meow…
// critical section
V(litter_box);
```

litter_box = 0

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

litter_box = 0 →1

Two Uses of Semaphores

2. Scheduling
   – Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {            Right_Paw() {
  slide_left();           P(wait_left);
  V(wait_left);           slide_left();
  P(wait_right);          slide_right();
  slide_right();          V(wait_right);
}                       }
```

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;
```

wait_left = 0
wait_right = 0

```
Left_Paw() {                Right_Paw() {
   slide_left();               P(wait_left);
   V(wait_left);               slide_left();
   P(wait_right);              slide_right();
   slide_right();              V(wait_right);
}                           }
```

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;
```

wait_left = 0
wait_right = 0

```
Left_Paw() {                 Right_Paw() {
  slide_left();                P(wait_left);
  V(wait_left);                slide_left();
  P(wait_right);               slide_right();
  slide_right();               V(wait_right);
}                            }
```

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;
```

wait_left = 0
wait_right = 0

```
Left_Paw() {                      Right_Paw() {
   slide_left();                     P(wait_left);
   V(wait_left);                     slide_left();
   P(wait_right);                    slide_right();
   slide_right();                    V(wait_right);
}                                 }
```

wait

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                  Right_Paw() {
  slide_left();                 P(wait_left);
  V(wait_left);                 slide_left();
  P(wait_right);                slide_right();
  slide_right();                V(wait_right);
}                             }
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling

- Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {           Right_Paw() {
  slide_left();          P(wait_left);
  V(wait_left);          slide_left();
  P(wait_right);         slide_right();
  slide_right();         V(wait_right);
}                      }
```

wait_left = 0 →1
wait_right = 0

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

wait_left = 1 → 0
wait_right = 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
  slide_left();               P(wait_left);
  V(wait_left);               slide_left();
  P(wait_right);              slide_right();
  slide_right();              V(wait_right);
}                           }
```

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;
```

wait_left = 0
wait_right = 0

```
Left_Paw() {                 Right_Paw() {
  slide_left();                P(wait_left);
  V(wait_left);                slide_left();
  P(wait_right);               slide_right();
  slide_right();               V(wait_right);
}                            }
```

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
   slide_left();               P(wait_left);
   V(wait_left);               slide_left();
   P(wait_right);              slide_right();
   slide_right();              V(wait_right);
}                           }
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

wait_left = 0
wait_right = 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                   Right_Paw() {
  slide_left();                  P(wait_left);
  V(wait_left);                  slide_left();
  P(wait_right);                 slide_right();
  slide_right();                 V(wait_right);
}                              }
```

Two Uses of Semaphores

2. Scheduling

- Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                 Right_Paw() {
  slide_left();                P(wait_left);
  V(wait_left);                slide_left();
  P(wait_right);               slide_right();
  slide_right();               V(wait_right);
}                            }
```

wait_left = 0
wait_right = 0 →1

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

wait_left = 0
wait_right = 1 → 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
  slide_left();               P(wait_left);
  V(wait_left);               slide_left();
  P(wait_right);              slide_right();
  slide_right();              V(wait_right);
}                           }
```

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

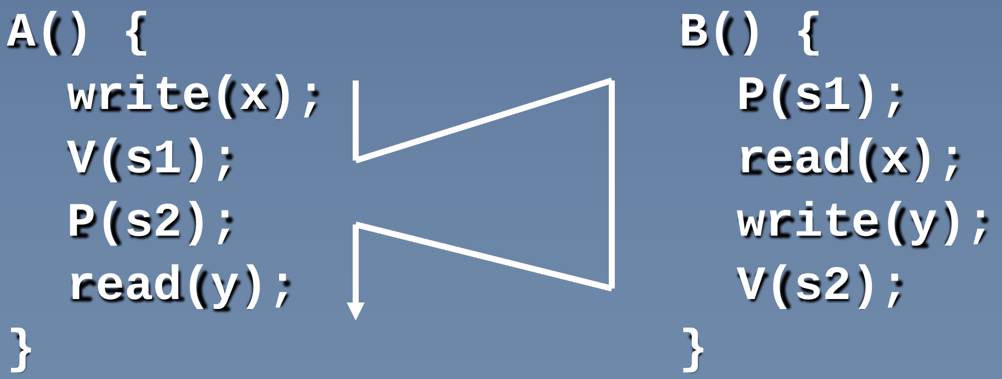
```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
  slide_left();               P(wait_left);
  V(wait_left);               slide_left();
  P(wait_right);              slide_right();
  slide_right();              V(wait_right);
}                           }
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling
   - Semaphore usually has an initial value of 0

```
semaphore s1 = 0;
semaphore s2 = 0;
```



```
A() {                        B() {
  write(x);                    P(s1);
  V(s1);                       read(x);
  P(s2);                       write(y);
  read(y);                     V(s2);
}                            }
```

Producer-Consumer with a Bounded Buffer

- A classic problem
- A producer put things into a shared buffer
- A consumer takes them out

Problem Constraints

- The solution involves both scheduling and mutual exclusion
- Constraints
  - The consumer must wait if buffers are empty (scheduling constraint)
  - The producer must wait if buffers are full (scheduling constraint)
  - Only one thread can manipulate the buffer at a time (mutual exclusion)

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = N;
semaphore nLoadedBuffers = 0;
```

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = N;
semaphore nLoadedBuffers = 0;

Producer() {

   P(mutex);
   // put 1 item in the buffer
   V(mutex);

}
```

```
Consumer() {

   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);

}
```

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = N;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);

}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);

}
```

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = N;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 2
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 2
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
→  P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 2
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
→  P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 2 → 1
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
 → P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1 → 0
nFreeBuffers = 1
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
→  // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 1
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

→ Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 1
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
 → P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 1 → 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
→  P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
 →  V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 0 → 1
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1 → 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
→  // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
 →  V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0 → 1

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 1 → 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
 →  P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
 → V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0 → 1
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 1 → 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
→  // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
→ Consumer() {
      P(nLoadedBuffers);
      P(mutex);
      // take 1 item from the
      // buffer
      V(mutex);
      V(nFreeBuffers);
  }
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
→  P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
→  V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0 → 1
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;
```

```
→ Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
  }
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
 → P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
→   V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 0 → 1
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
    P(nFreeBuffers);
    P(mutex);
    // put 1 item in the buffer
    V(mutex);
    V(nLoadedBuffers);
}
```

```
Consumer() {
    P(nLoadedBuffers);
    P(mutex);
    // take 1 item from the
    // buffer
    V(mutex);
    V(nFreeBuffers);
}
```

mutex = 1
nFreeBuffers = 1 → 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
 → P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 1 → 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
}
```

```
  Consumer() {
     P(nLoadedBuffers);
     P(mutex);
     // take 1 item from the
     // buffer
     V(mutex);
     V(nFreeBuffers);
→ }
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0

Developing the Solution

- Each constraint needs a semaphore

```
semaphore mutex = 1;
semaphore nFreeBuffers = 2;
semaphore nLoadedBuffers = 0;

Producer() {
   P(nFreeBuffers);
   P(mutex);
   // put 1 item in the buffer
   V(mutex);
   V(nLoadedBuffers);
→ }
```

```
Consumer() {
   P(nLoadedBuffers);
   P(mutex);
   // take 1 item from the
   // buffer
   V(mutex);
   V(nFreeBuffers);
}
```

mutex = 0
nFreeBuffers = 0
nLoadedBuffers = 0