

---

# ***SYSC 3303 Real-Time Concurrent Systems***

## **Synchronizing Java Threads: Buffer Example**

- Copyright © 2003 D.L. Bailey and L.S.Marshall, Systems and Computer Engineering, Carleton University
- revised January 14, 2003

---

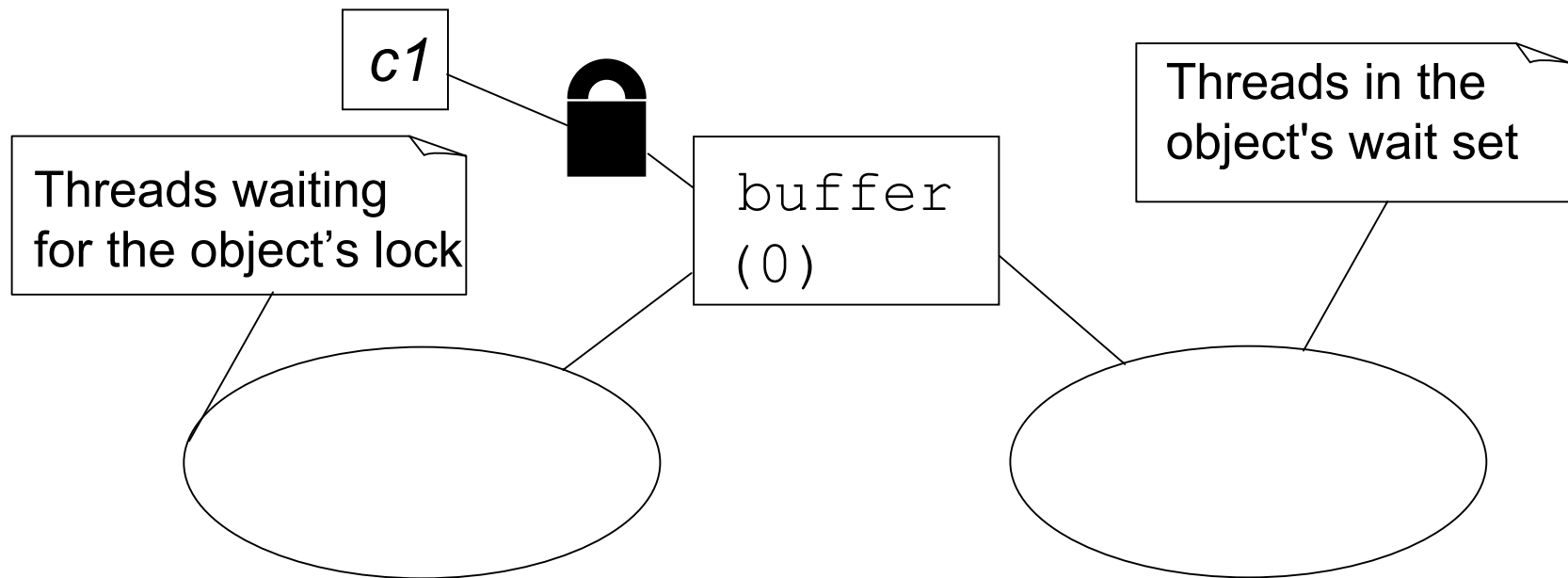
## ***Producer/Consumer Execution with Buffer***

- Suppose the `Buffer (buffer)` is empty, and consumer threads `c1` and `c2` invoke `removeFirst()`
- Then threads `p1` through `p5` invoke `addLast()`
- Finally `p6` invokes `addLast()` and `c3` invokes `removeFirst()`

---

# ***Producer/Consumer Execution with Buffer***

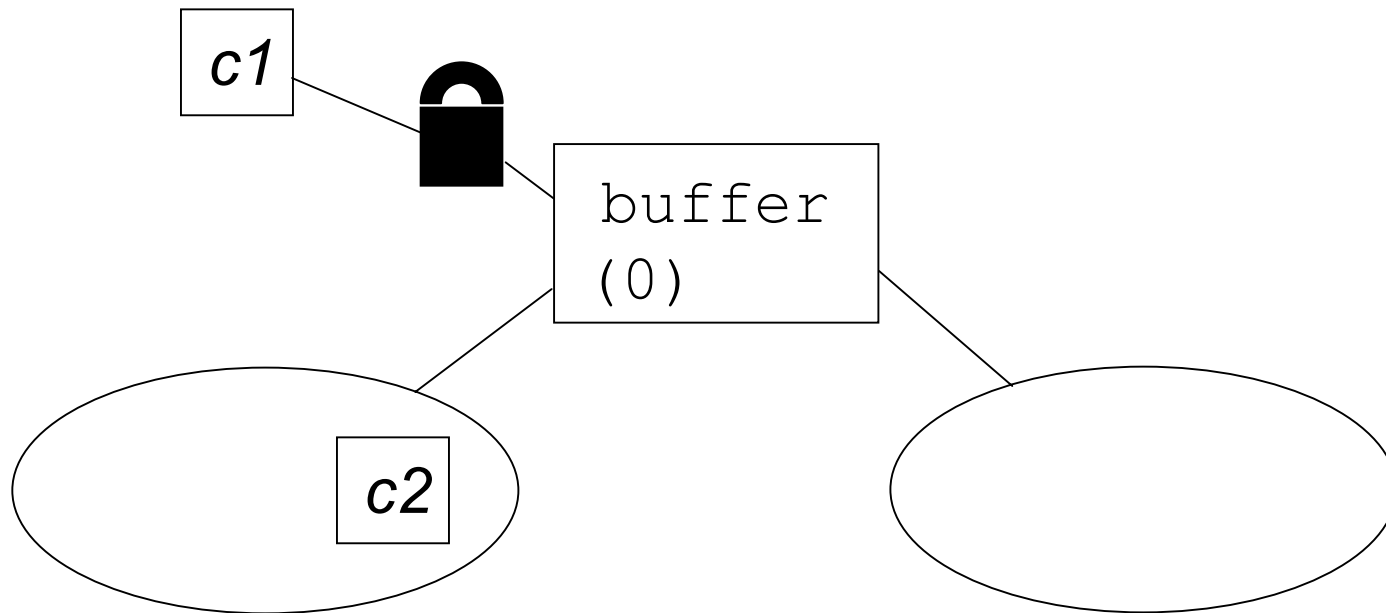
- *c1* gets the lock



---

## ***Producer/Consumer Execution with Buffer***

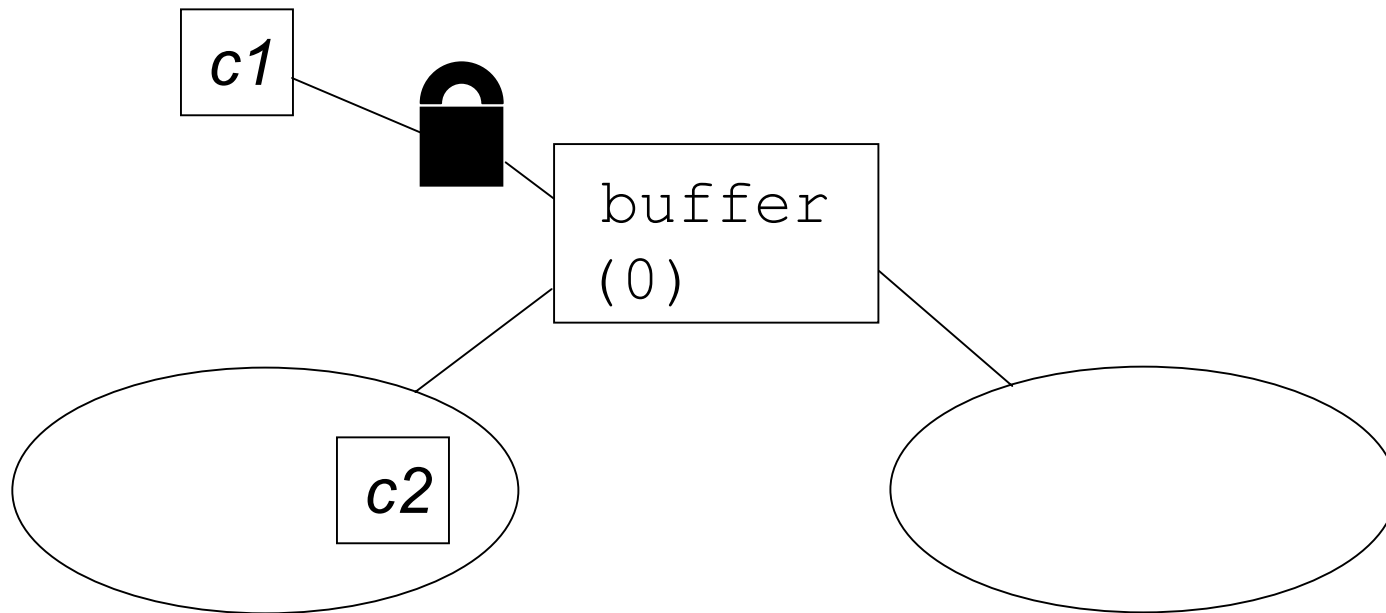
- *c2* arrives while *c1* has the lock, so *c2* blocks



---

## ***Producer/Consumer Execution with Buffer***

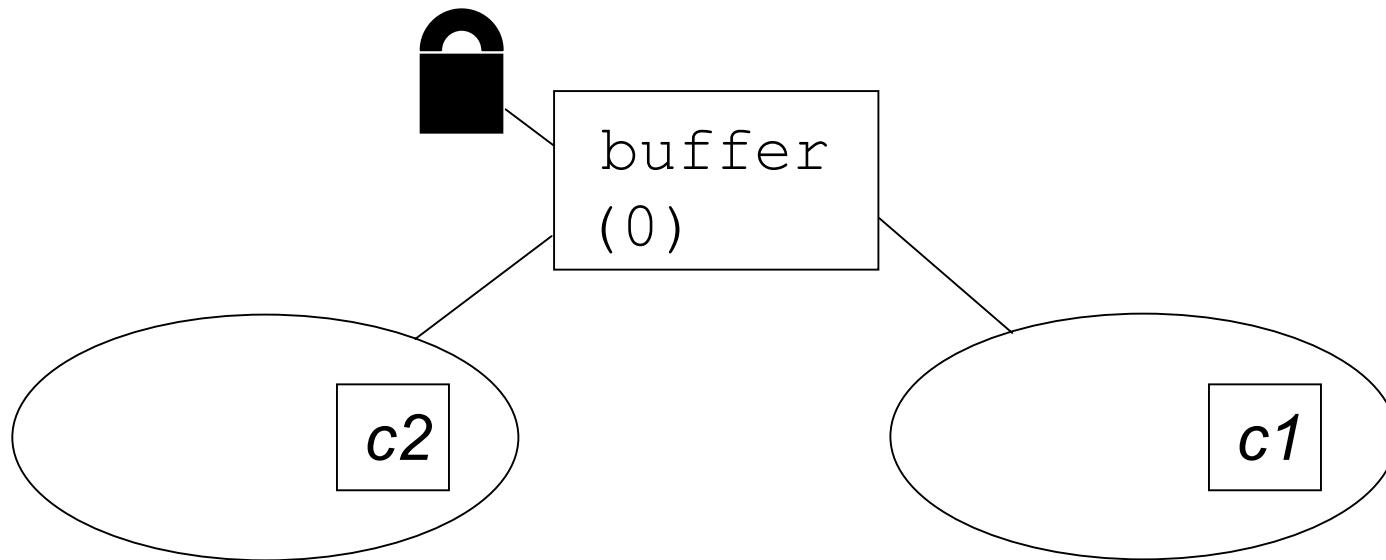
- *c1* starts executing `removeFirst()`



---

## ***Producer/Consumer Execution with Buffer***

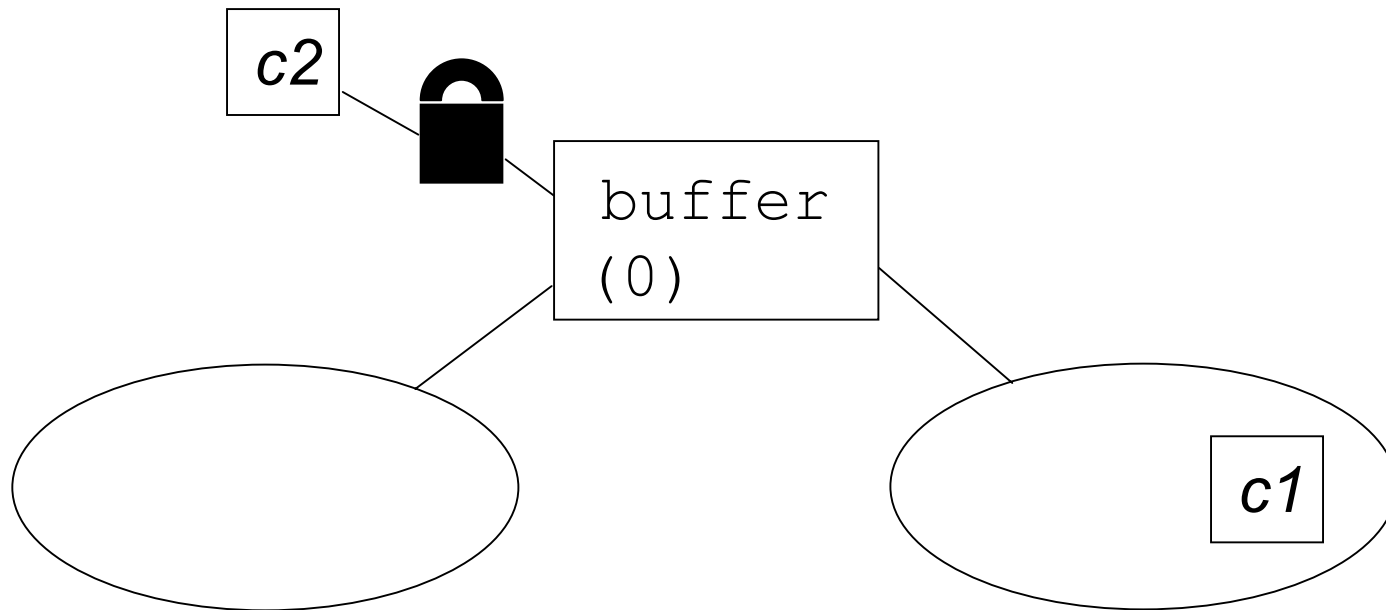
- *c1* enters the `while` loop body (`readable` is `false`)
  - it invokes `wait()`, releases the object's lock, is placed in the wait set and blocks



---

## ***Producer/Consumer Execution with Buffer***

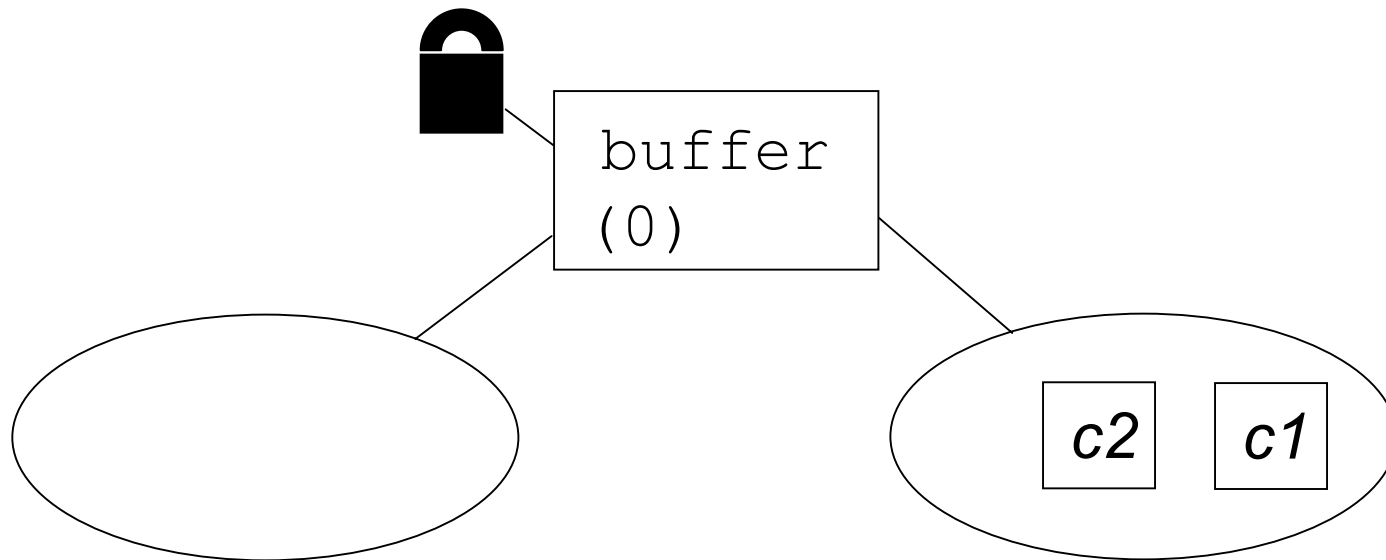
- The JVM grants the lock to *c2*, which starts executing `removeFirst()`



---

## ***Producer/Consumer Execution with Buffer***

- `c2` enters the `while` loop body (`readable` is `false`)
  - it invokes `wait()`, releases the object's lock, is placed in the wait set and blocks

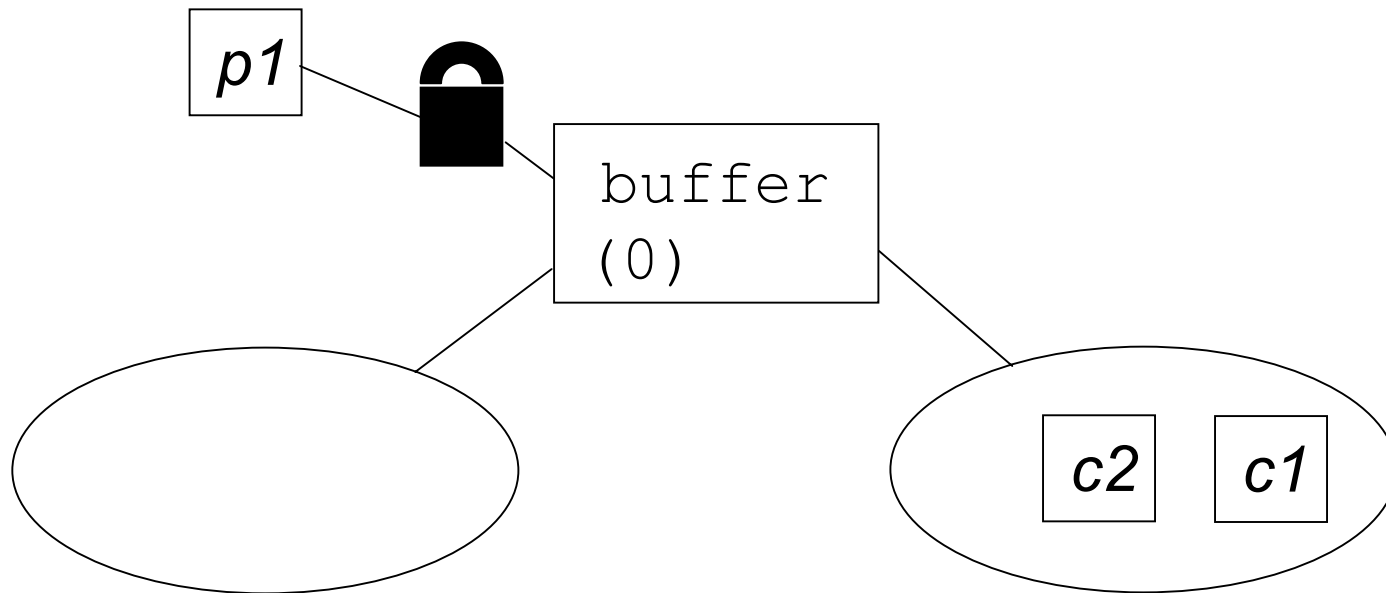




---

## ***Producer/Consumer Execution with Buffer***

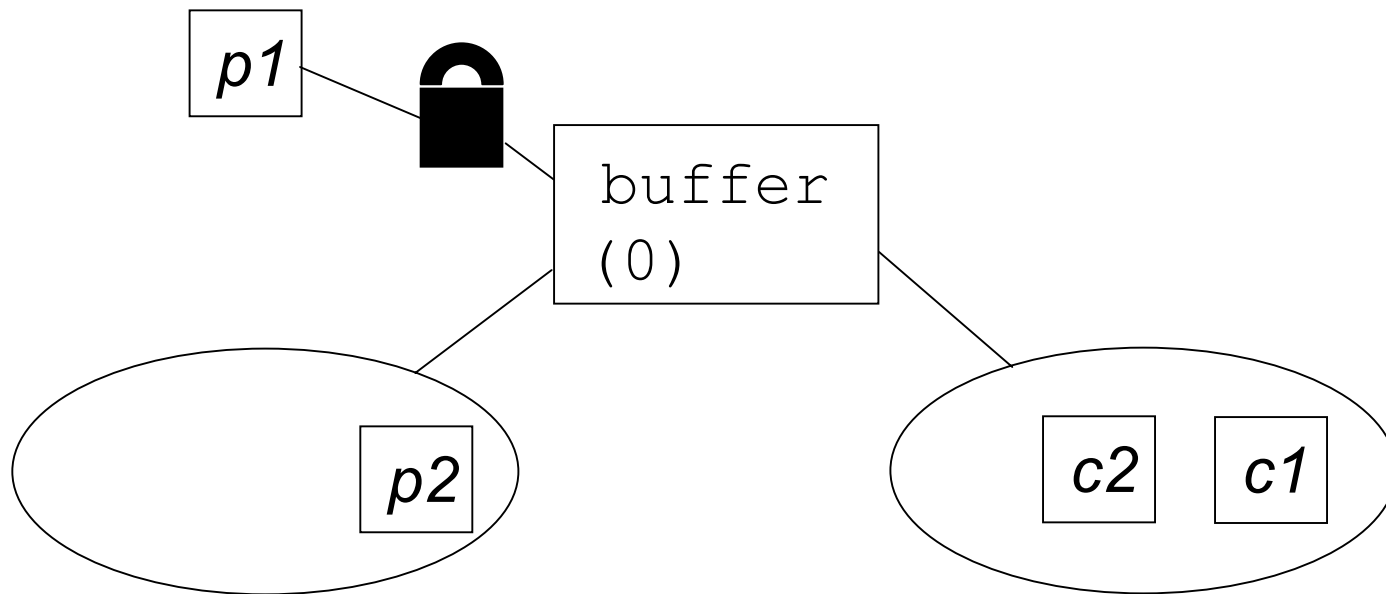
- Producer *p1* invokes `addLast()`, obtains the object's lock, and starts executing `addLast()`
- `writable` is `true`, so it doesn't invoke `wait()`



---

## ***Producer/Consumer Execution with Buffer***

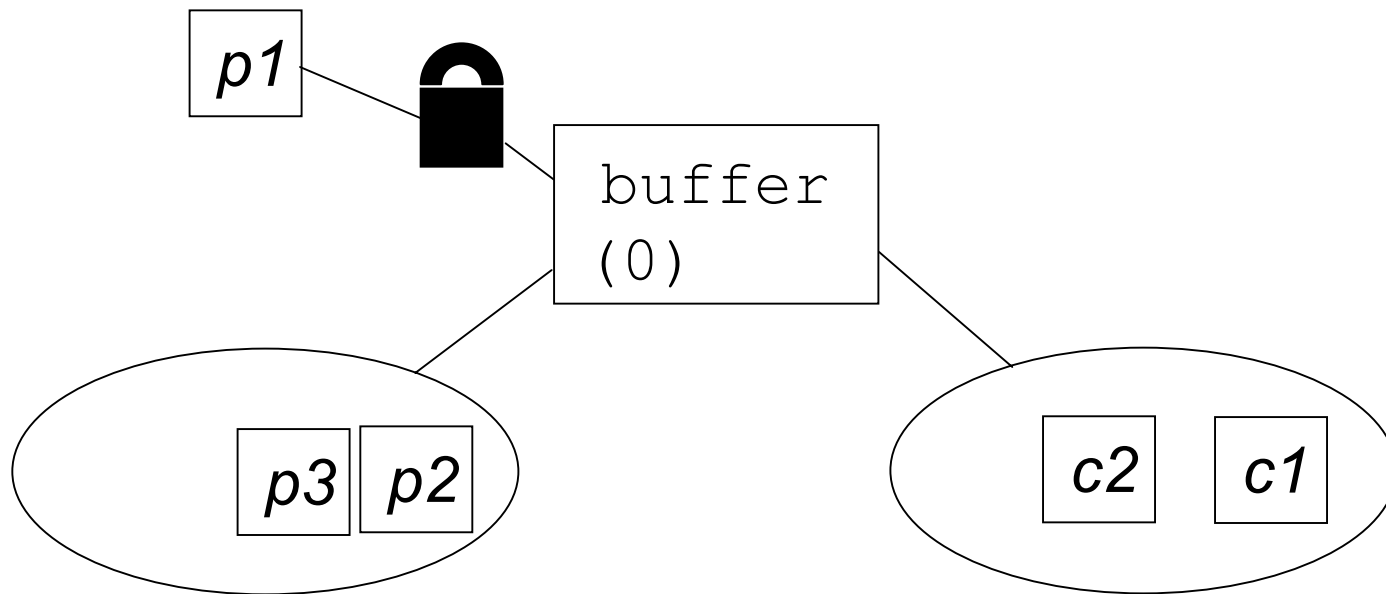
- Producer *p2* invokes `addLast()` and blocks until it can obtain the lock



---

## ***Producer/Consumer Execution with Buffer***

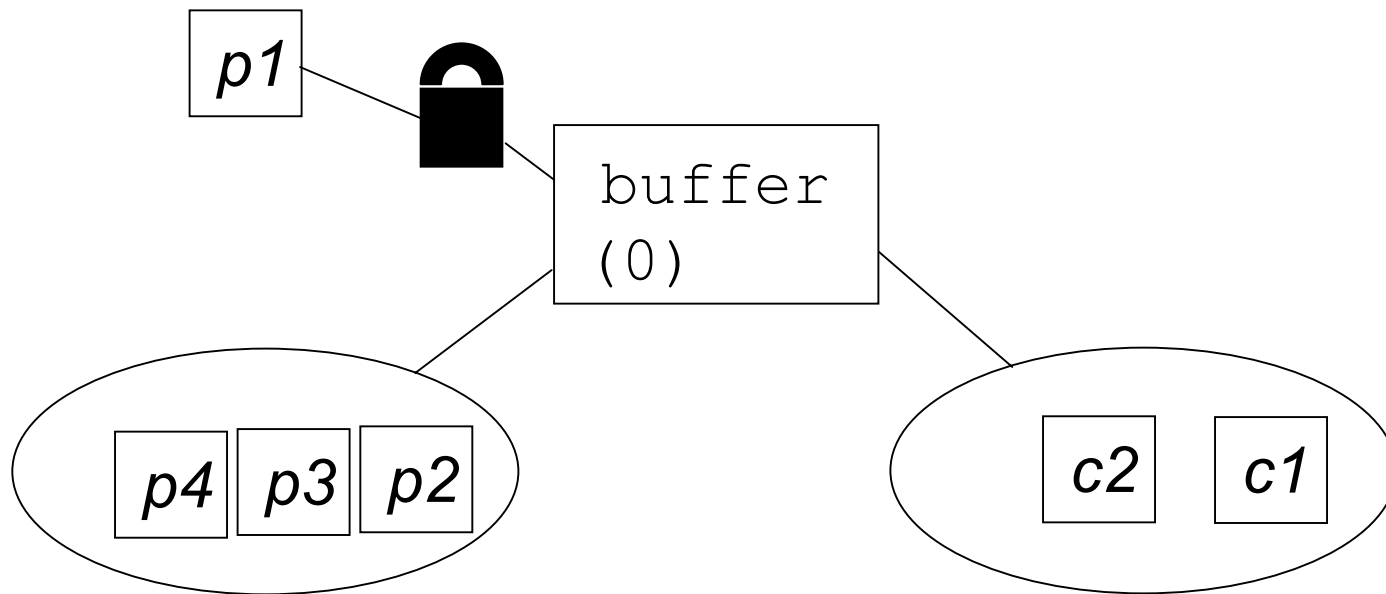
- Producer *p3* invokes `addLast()` and blocks until it can obtain the lock



---

## ***Producer/Consumer Execution with Buffer***

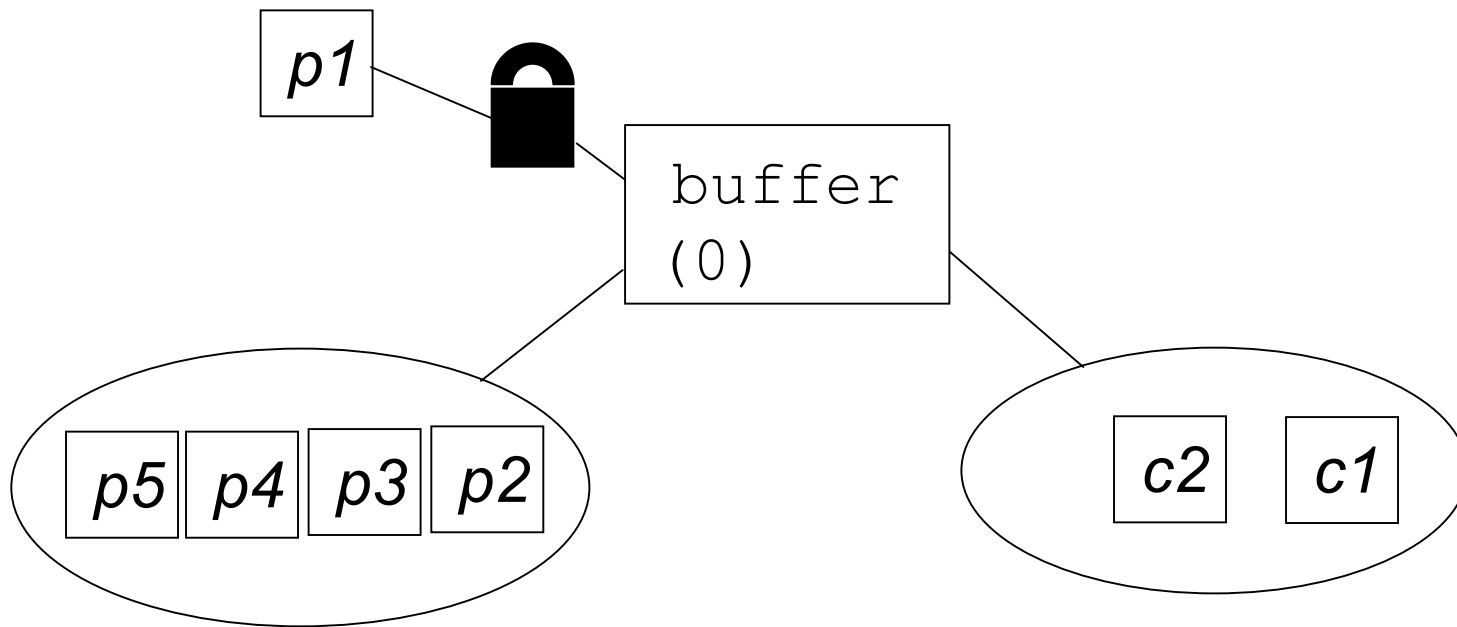
- Producer *p4* invokes `addLast()` and blocks until it can obtain the lock



---

## ***Producer/Consumer Execution with Buffer***

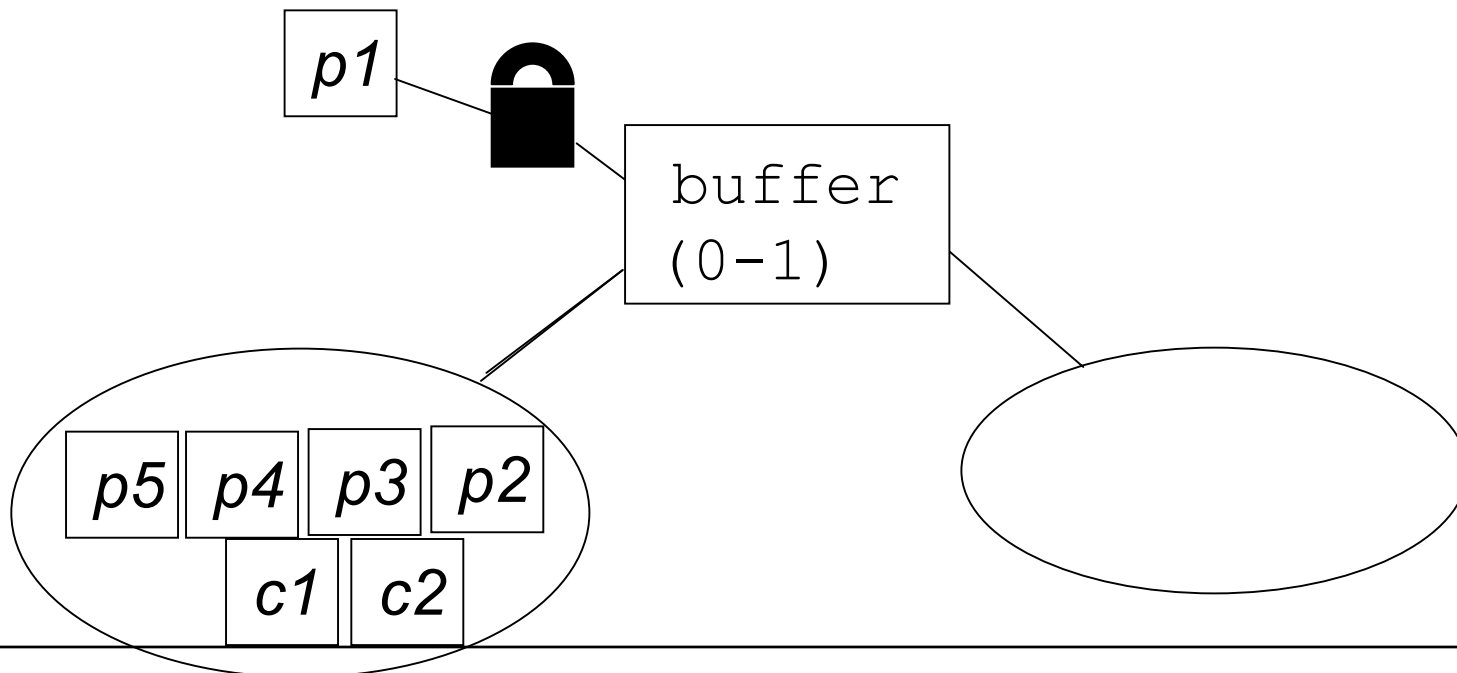
- Producer *p5* invokes `addLast()` and blocks until it can obtain the lock



---

## ***Producer/Consumer Execution with Buffer***

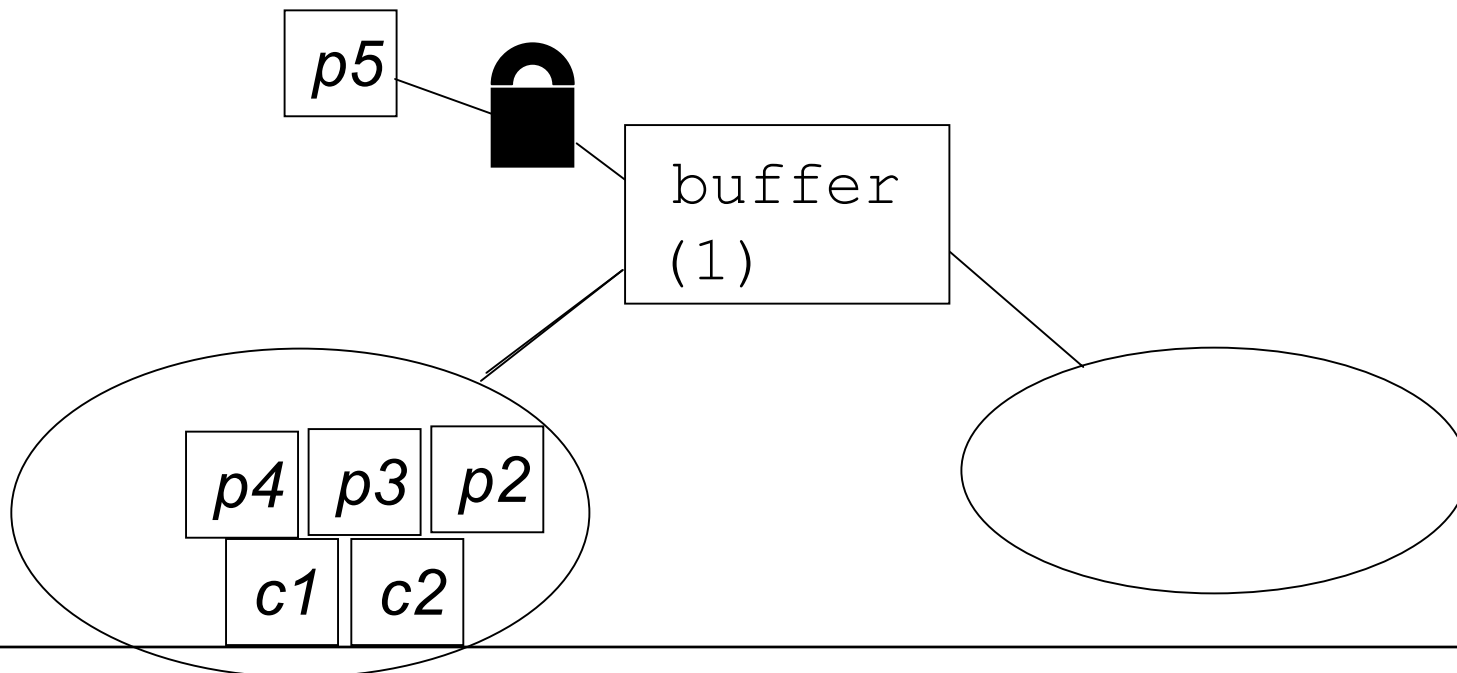
- *p1* assigns `true` to `readable`, then invokes `notifyAll()`
  - *c1* and *c2* are reenabled for thread scheduling



---

## ***Producer/Consumer Execution with Buffer***

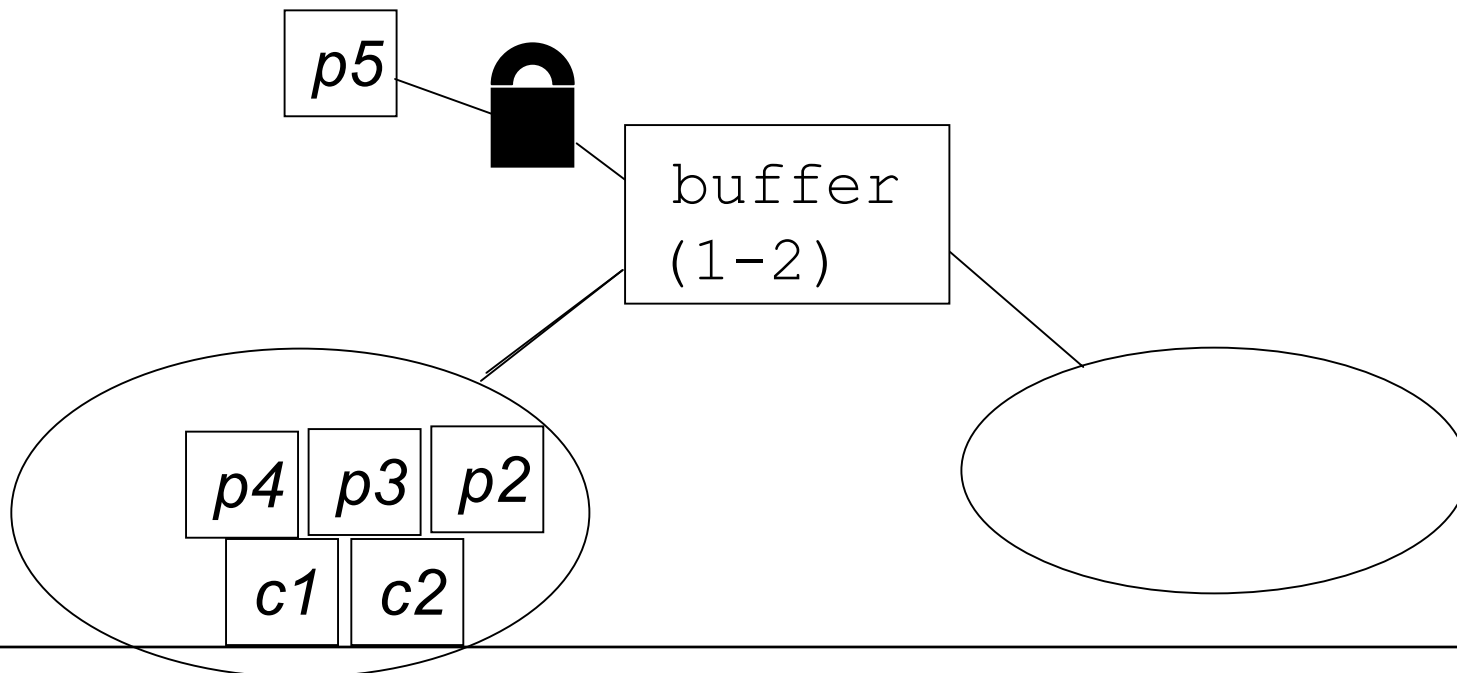
- *p1* returns from `addLast()` and relinquishes the lock
- the JVM grants the lock to *p5* (could have been any of the others), which starts executing `addLast()`



---

## ***Producer/Consumer Execution with Buffer***

- *p5* does not invoke `wait()` as `writable` is `true`
- *p5* invokes `notifyAll()`, returns from `addLast()` and relinquishes the lock

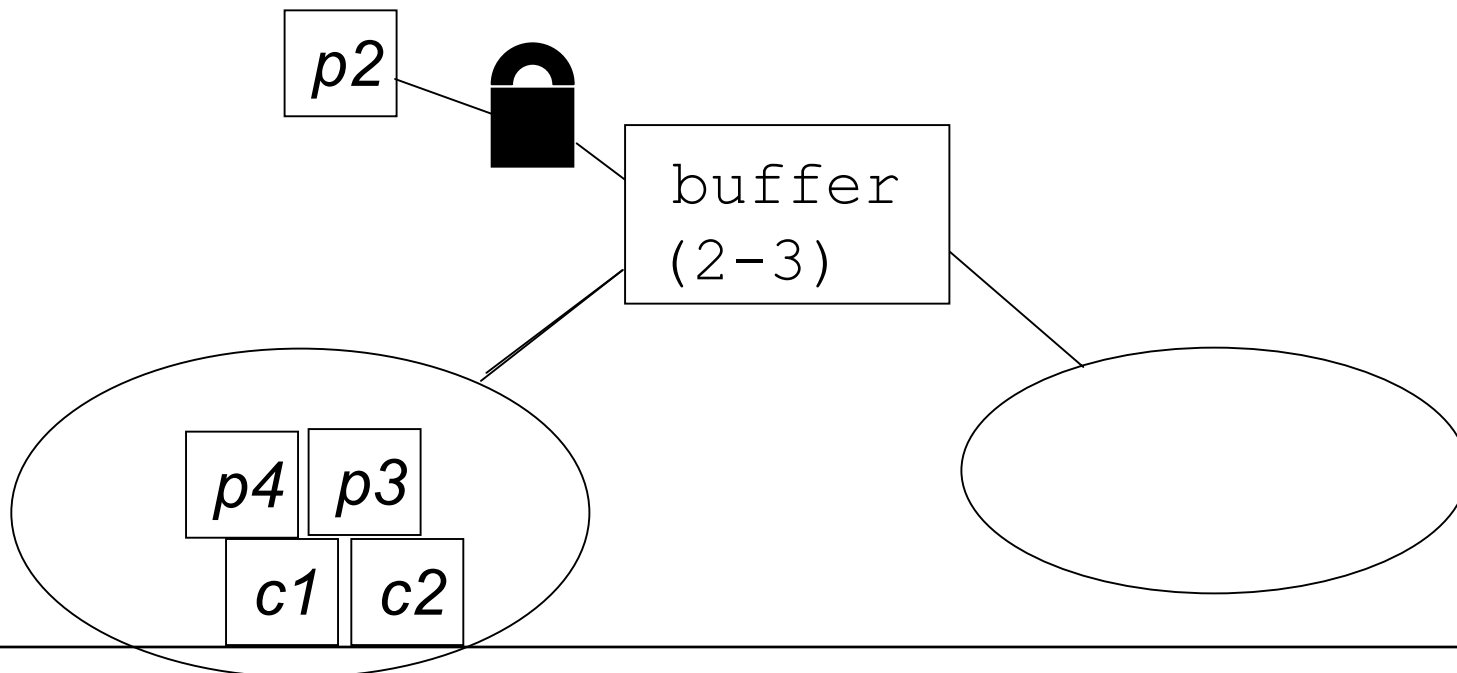




---

## ***Producer/Consumer Execution with Buffer***

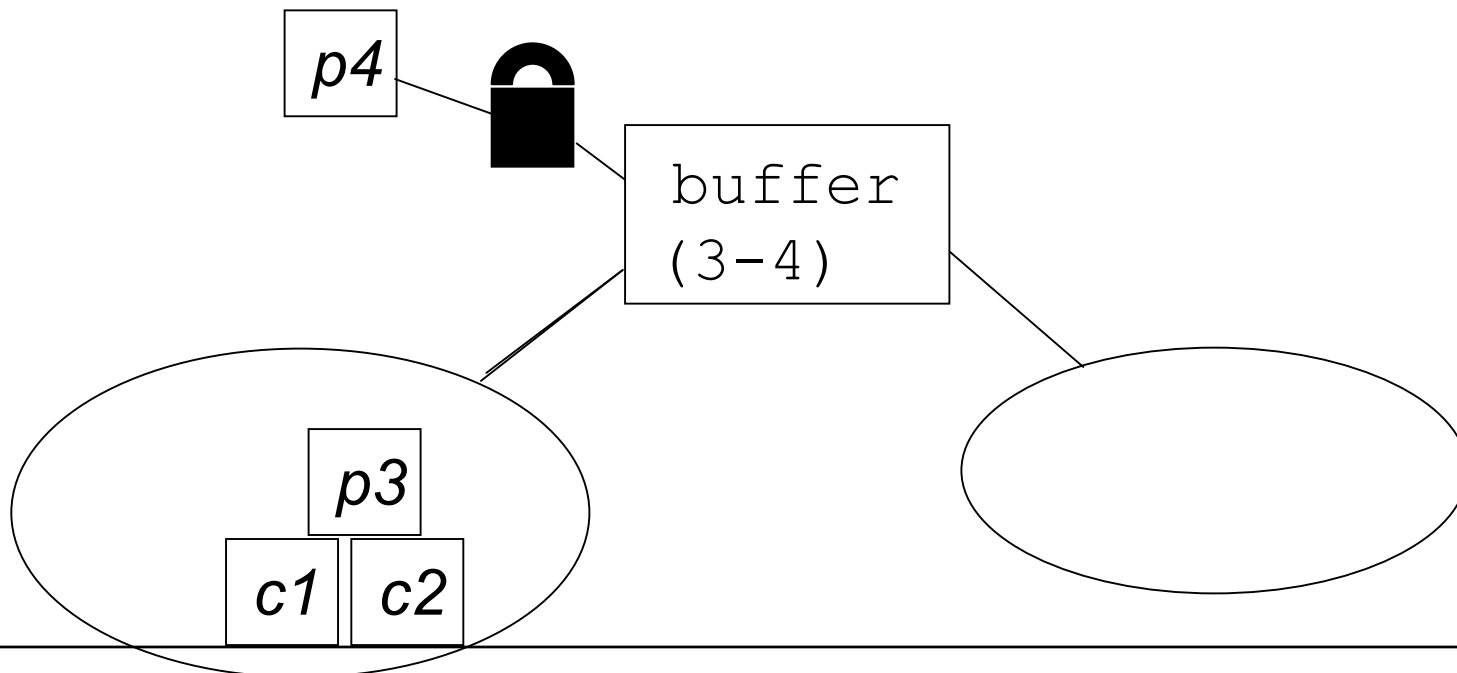
- the JVM grants the lock to *p2* (could have been any of the others), which starts executing `addLast()`
- *p2* does not invoke `wait()` as `writable` is `true`
- *p2* invokes `notifyAll()`, returns from `addLast()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

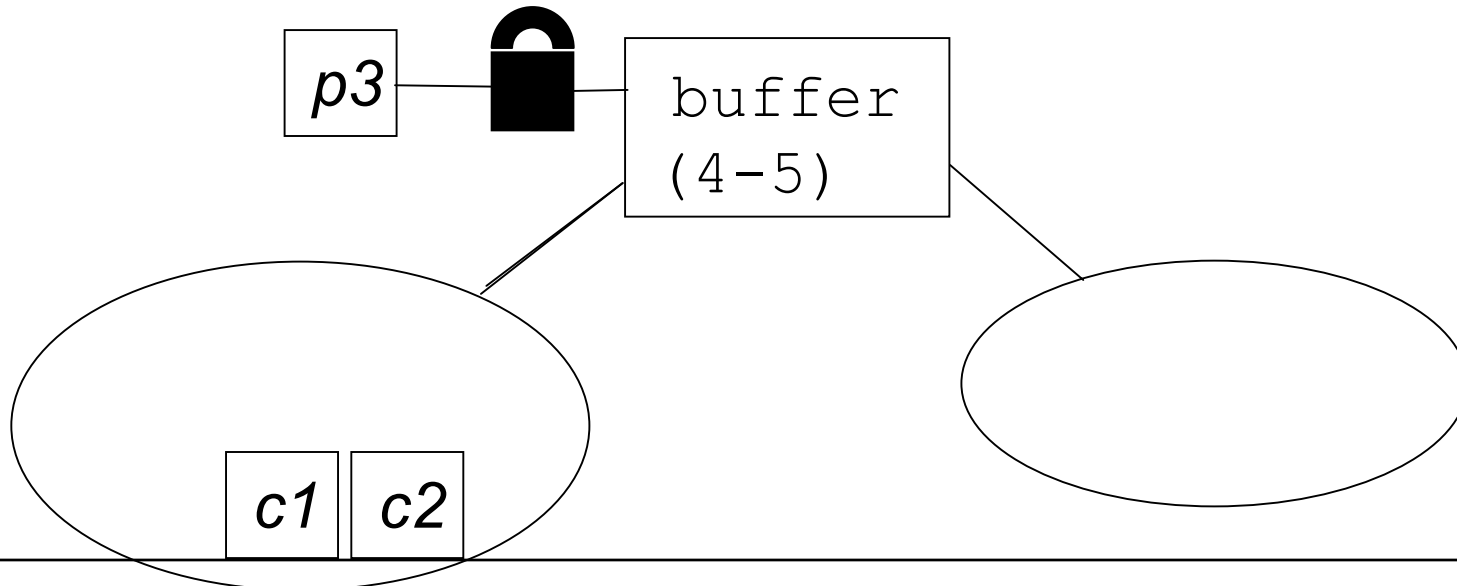
- the JVM grants the lock to *p4* (could have been any of the others), which starts executing `addLast()`
- *p4* does not invoke `wait()` as `writable` is `true`
- *p4* invokes `notifyAll()`, returns from `addLast()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

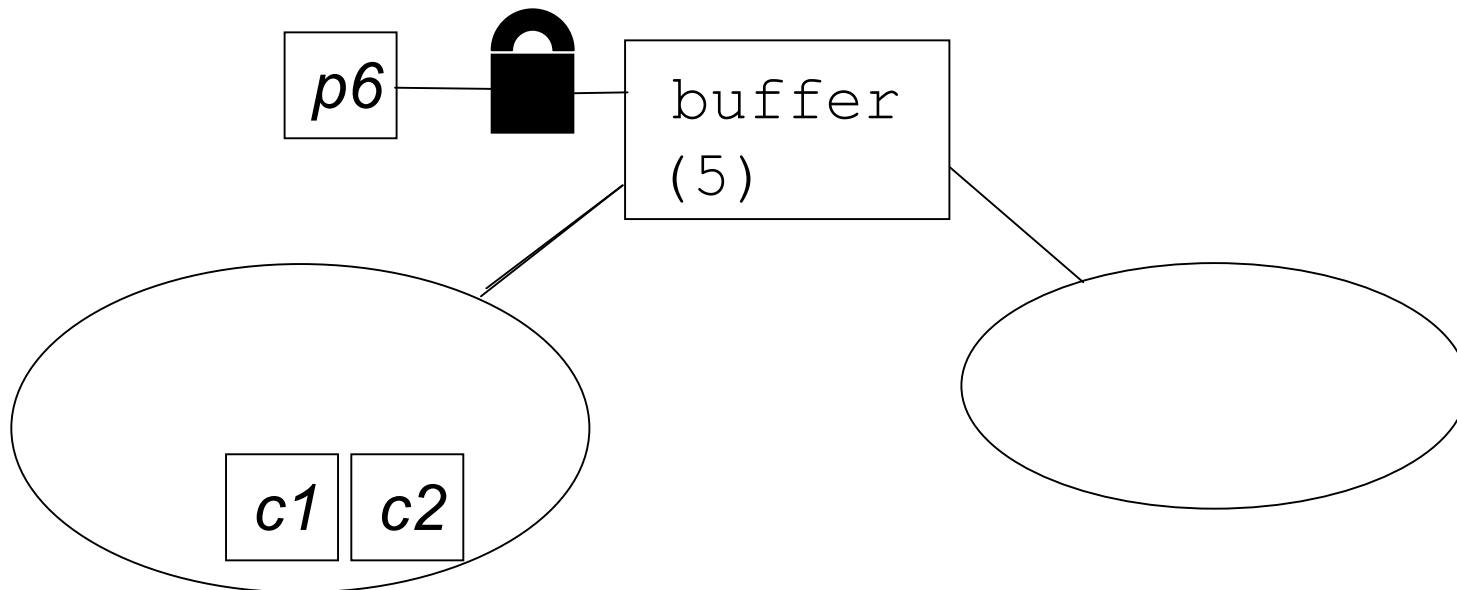
- the JVM grants the lock to *p3* (could have been any of the others), which starts executing `addLast()`
- *p3* does not invoke `wait()` as `writable` is `true`
- *p3* sets `writable` to `false` (buffer is now full), invokes `notifyAll()`, returns from `addLast()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

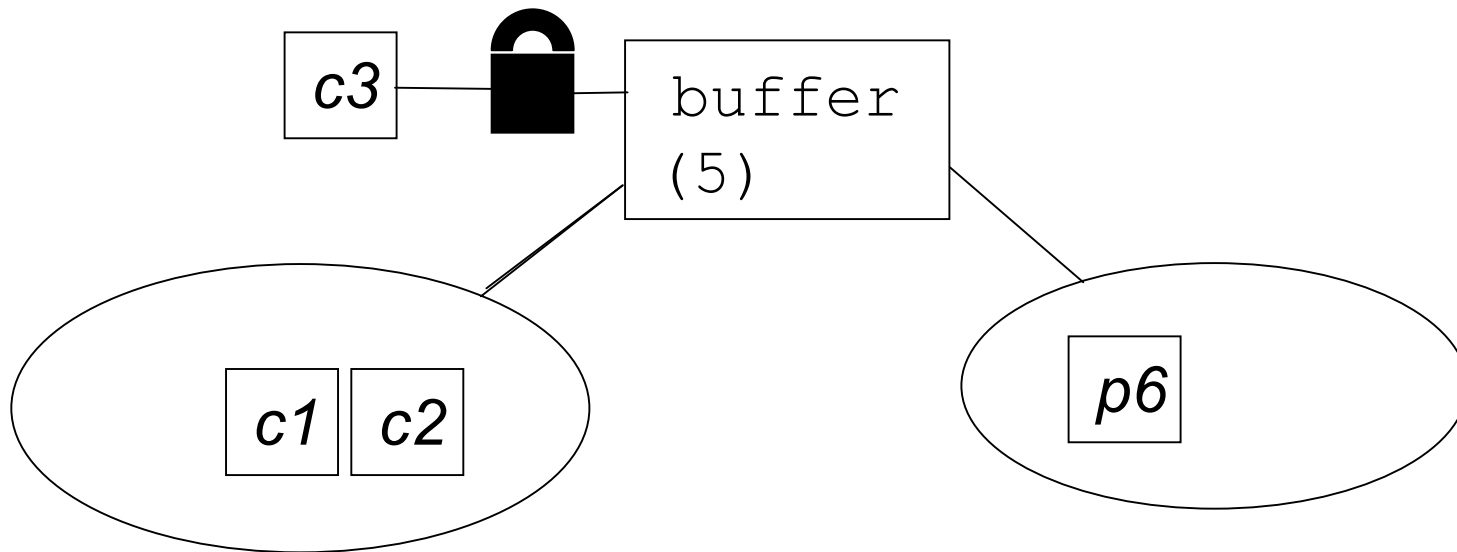
- *p6* arrives and is granted the lock, and starts executing `addLast()`
- *p6* invokes `wait()` as `writable` is `false`



---

## ***Producer/Consumer Execution with Buffer***

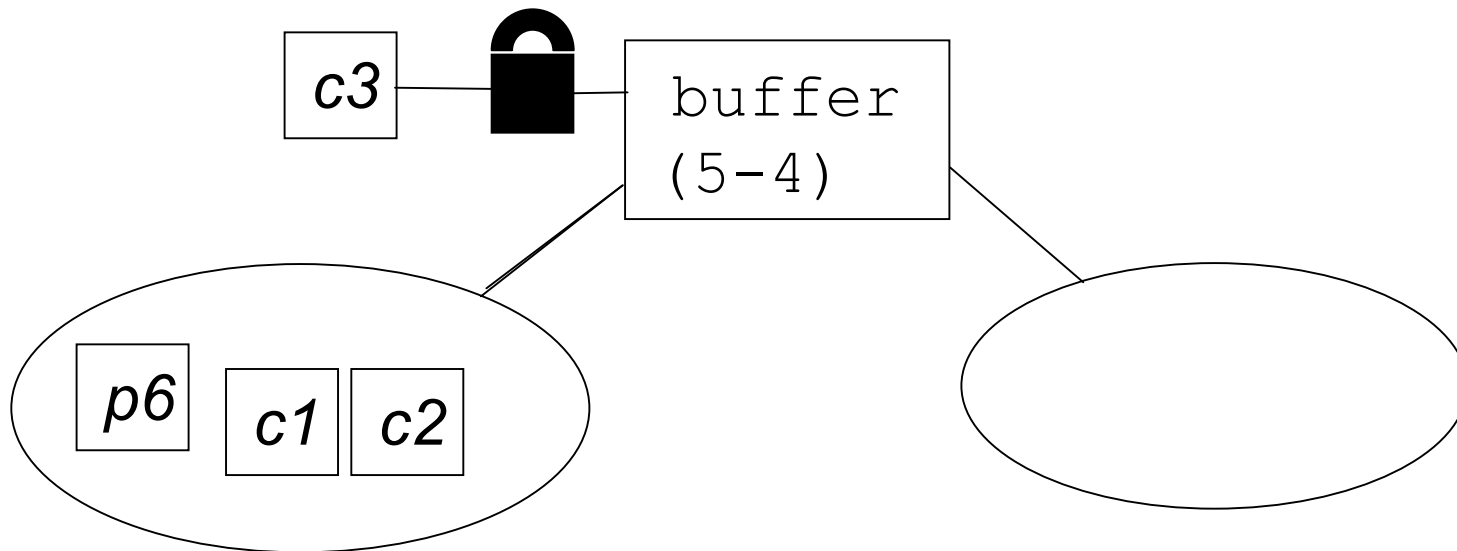
- `c3` arrives and is granted the lock, and starts executing `removeFirst()`
- `c3` does not invoke `wait()` as `readable` is `true`



---

## ***Producer/Consumer Execution with Buffer***

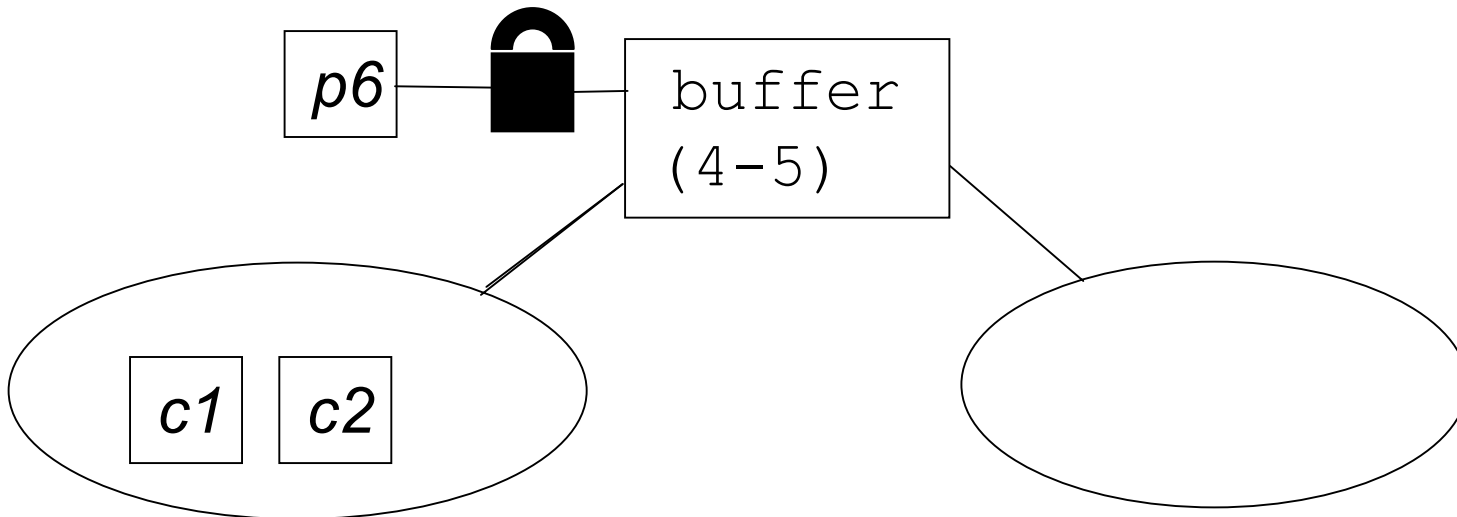
- ***c3*** sets `writable` to `true`, invokes `notifyAll()`, returns from `removeFirst()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

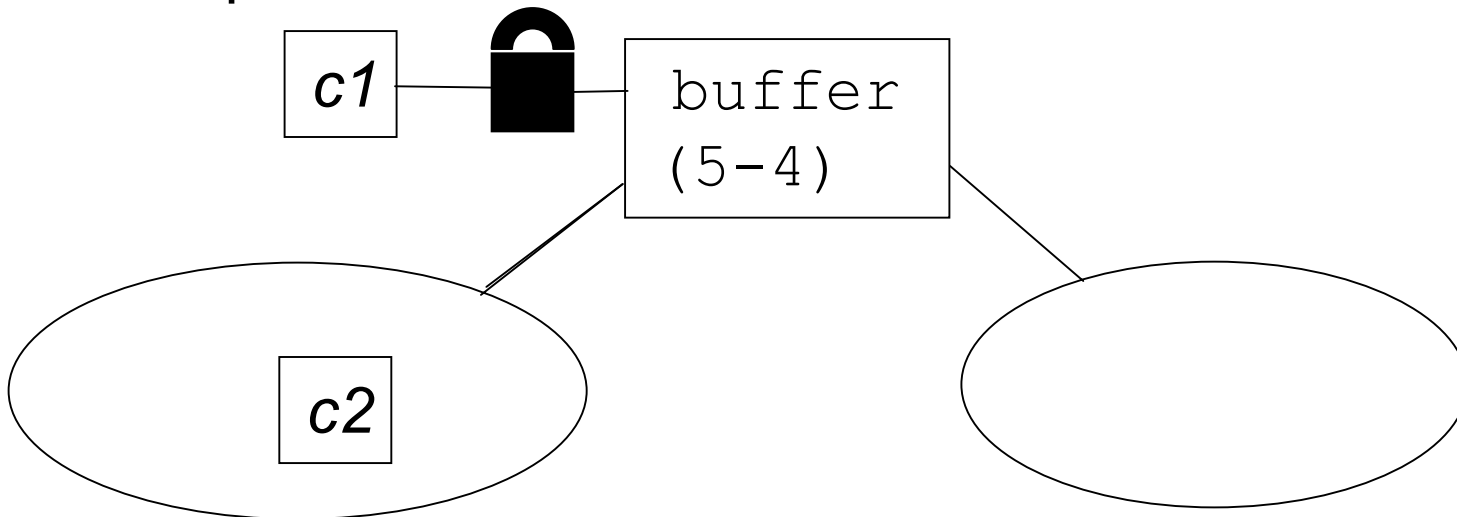
- The JVM grants the lock to *p6*, which starts executing `addLast()`
- *p6* does not invoke `wait()` as `writable` is `true`
- *p6* sets `writable` to `false`, invokes `notifyAll()`, returns from `addLast()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

- *c1* is granted the lock, and starts executing `removeFirst()`
- *c1* does not invoke `wait()` as `readable` is `true`
- *c1* sets `writable` to `true`, invokes `notifyAll()`, returns from `removeFirst()` and relinquishes the lock

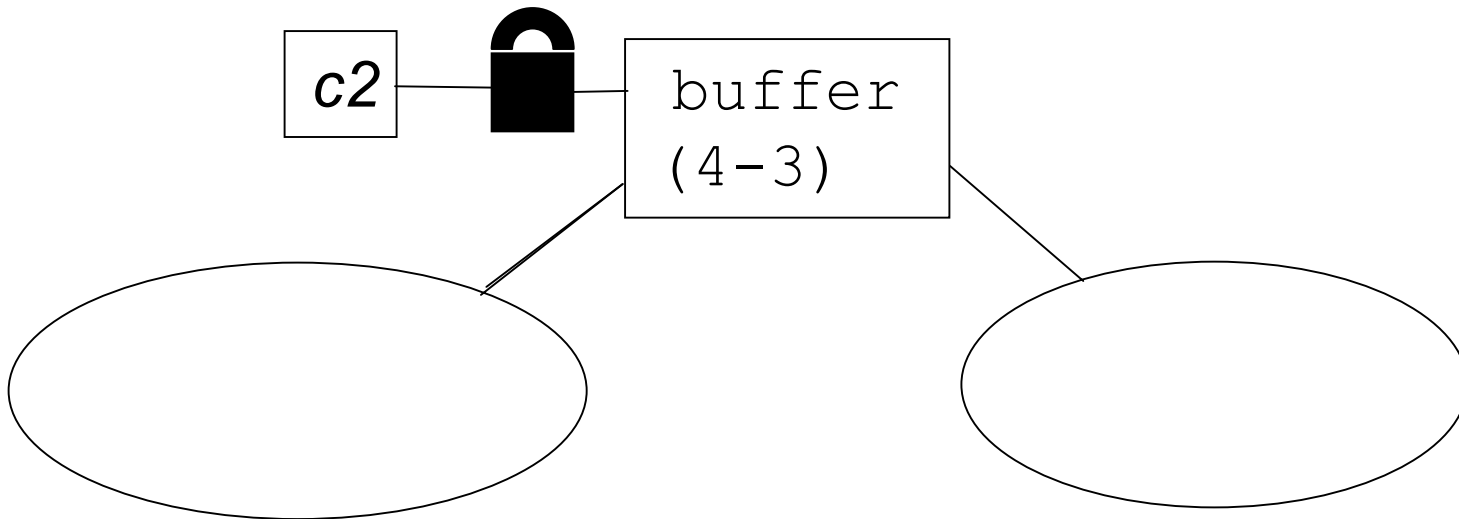




---

## ***Producer/Consumer Execution with Buffer***

- **c2** is granted the lock, and starts executing `removeFirst()`
- **c2** does not invoke `wait()` as `readable` is `true`
- **c2** invokes `notifyAll()`, returns from `removeFirst()` and relinquishes the lock



---

## ***Producer/Consumer Execution with Buffer***

- We are now awaiting more producers/consumers with 3 items remaining in the buffer

