University of Pittsburgh

Introduction to Operating Systems
CS 1550

Spring 2023

Sherif Khattab

ksm73@pitt.edu

5 YEARS Pitt SCI

(Some slides are from **Silberschatz, Galvin and Gagne ©2013)**

Announcements

- Upcoming deadlines

  - Homework 3 is due **this Friday**

  - Lab 1 is due on Tuesday 2/7 at 11:59 pm

  - Project 1 is due on Friday 2/17 at 11:59 pm

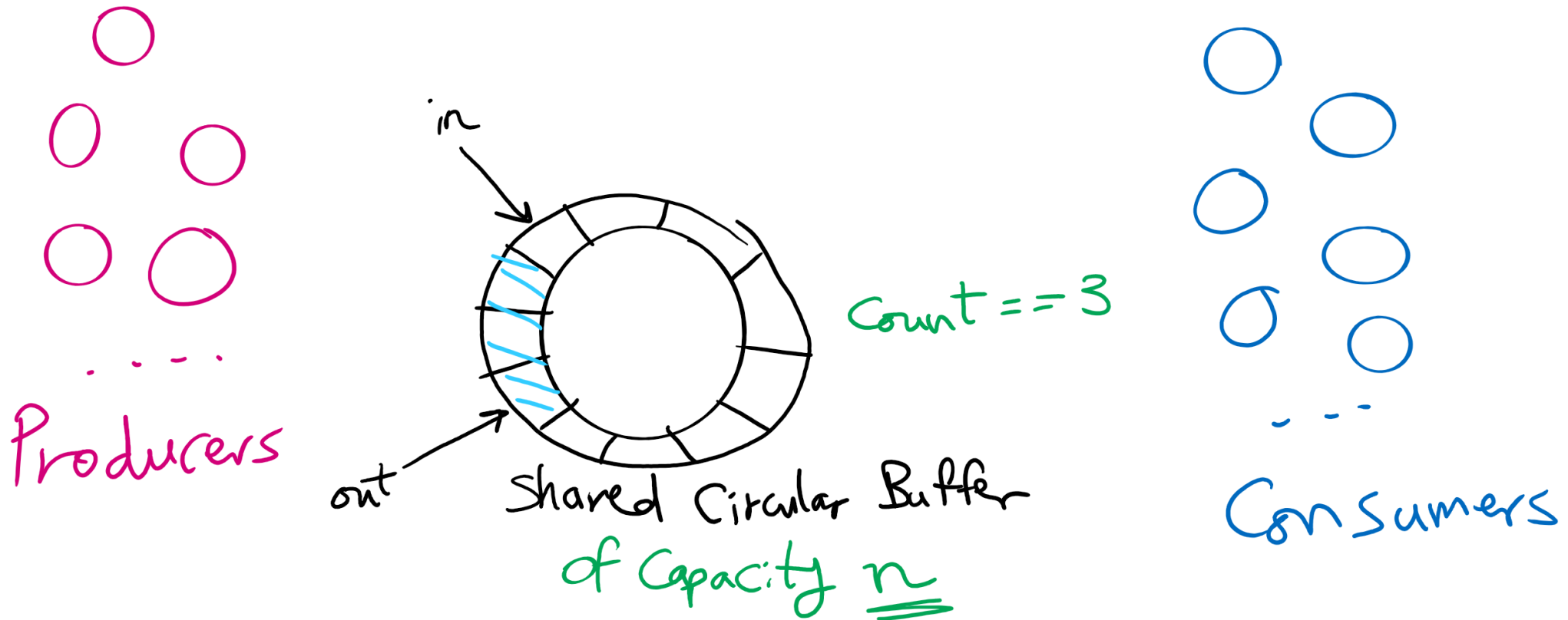   - Discussed in this week's recitations

Previous lecture …

- Bounded buffer problem

    - semaphore-based solution

It is easy to make mistakes when using semaphores

Produces Consumers Problem

Solving Producers Consumers using Semaphores

Semaphore empty($\underline{n}$), full(0)

Mutex Sem (1);

Producer

down (empty)

down (sem)

buffer[in] = new item

in += 1 % n

Count ++

up (sem)

up (full)

Consumer

down (full)

down (sem)

item = buffer[out]

out += 1 % n

Count --

up (sem)

up (empty)

Some thoughts

- Do we need the count variable?

- If we have one producer and one consumer

  - do we need the mutex?

- For multiple producers and consumers

  - Why do we need the mutex?

  - what benefit would we get if we have one mutex for producers and one for consumers?

Semaphore empty(n), full(0)
Mutex   Sem (1);

Producer

down (empty)
down (Sem)
buffer [in] = new item
in += 1 % n
Count ++
up (Sem)
up (full)

Consumer

down (full)
down (Sem)
item = buffer [out]
out += 1 % n
Count --
up (Sem)
up (empty)

Condition Variables

- Yet another construct for process/thread synchronization

  - (Add to Spinlock and Semaphore)

- Has 3 operations

  - wait()

  - signal()

  - broadcast()

- Not foreign to us at all

  - Every object variable in Java is a Condition Variable

Condition Variables Operations

- These 3 operations must be called while holding a mutex lock

- wait()
    - unlock mutex
    - block process
    - when awake, relock mutex
    - when successful, return

- signal()
    - wakeup one waiting process in the condition variable's queue if any

- broadcast()
    - wakeup all waiting processes in the condition variable's queue if any

Mutex sem;
ConditionVariable CV;

Producer

```
down (sem)
while (count == n)
        CV.wait ()
buffer[in] = new item
in = (in+1) % n
count ++
CV.broadcast ()
up(sem)
```

Consumer

```
down (sem)
while (count == 0)
        CV.wait ()
item = buffer[out]
out = (out+1) % n
count --
CV.broadcast ()
up(sem)
```

Readers & Writers Problem

- Many processes that may read and/or write

- Only one writer allowed at any time

- Many readers allowed, but not while a process is writing

- Real-world Applications

  - Database queries

  - We have this problem in Project 1

Semaphore-based Solution

Shared variables
int nreaders;
Semaphore mutex(1), writing(1);

Reader process
…
mutex.down();
nreaders += 1;
if (nreaders == 1) // wait if
  writing.down(); // 1st reader
mutex.up();
// Read some stuff
mutex.down();
nreaders -= 1;
if (nreaders == 0)   // signal if
  writing.up();   // last reader
mutex.up();
…

Writer process
…
writing.down();
// Write some stuff
writing.up();
…

- enterRead

Reader process

```
…
mutex.down();
nreaders += 1;
if (nreaders == 1) // wait if
  writing.down();  // 1st reader
mutex.up();
// Read some stuff
mutex.down();
nreaders -= 1;
if (nreaders == 0)        // signal if
  writing.up();   // last reader
mutex.up();
…
```

- read

> ## Reader process
>
> …
> mutex.down();
> nreaders += 1;
> if (nreaders == 1) // wait if
>   writing.down(); // 1st reader
> mutex.up();
> **// Read some stuff**
> mutex.down();
> nreaders -= 1;
> if (nreaders == 0)   // signal if
>   writing.up();   // last reader
> mutex.up();
> …

- doneRead

<div align="center">

Reader process

</div>

```
…
mutex.down();
nreaders += 1;
if (nreaders == 1) // wait if
  writing.down();  // 1st reader
mutex.up();
// Read some stuff
mutex.down();
nreaders -= 1;
if (nreaders == 0)        // signal if
  writing.up();   // last reader
mutex.up();
…
```

- enterWrite

Writer process

...

<mark>writing.down();</mark>

// Write some stuff

writing.up();

...

- write

Writer process

...
writing.down();
<mark>// Write some stuff</mark>
writing.up();
...

- doneWrite

Writer process

...

writing.down();

// Write some stuff

writing.up();

...