CS 3502: Operating Systems

Assignment 3 — Bounded Buffer Problem

Shared Memory & Semaphore Synchronization

Name: Aspen Steele

Section: 04

Date: November 14, 2025

Git Repository: https://github.com/aspensteele/cs3502

Overview

This assignment implements the **Bounded Buffer Problem** using shared memory and semaphores. The objective is to synchronize multiple producer and consumer processes so they can safely share data without race conditions or deadlocks. Semaphores ensure mutual exclusion and coordinate access to the circular buffer.

—

Test 1 — Single Producer and Consumer

Command:

```
./producer 1 5 &
./consumer 1 5 &
wait
```

In this test, a single producer and consumer share the buffer. The producer creates five items (1000–1004), and the consumer retrieves them in order, confirming correct synchronization and shared memory setup.

Figure 1: Test 1 Output — Single producer and consumer operating correctly.

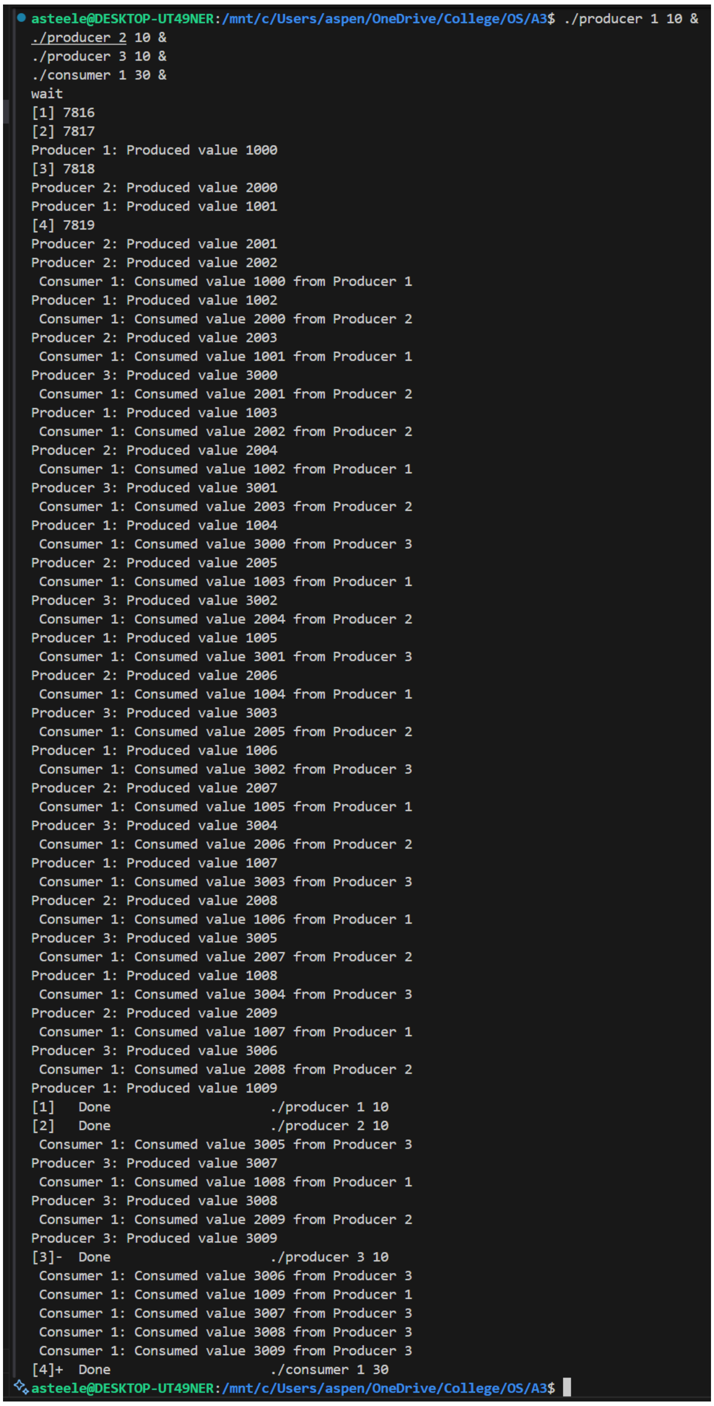
—

Test 2 — Multiple Producers

Command:

```
./producer 1 10 &
./producer 2 10 &
./producer 3 10 &
./consumer 1 30 &
wait
```

Three producers generate ten items each while one consumer retrieves all thirty. The interleaved output demonstrates proper use of semaphores — multiple producers can share the same buffer safely, without data loss or duplication.



```
asteele@DESKTOP-UT49NER:/mnt/c/Users/aspen/OneDrive/College/OS/A3$ ./producer 1 10 &
./producer 2 10 &
./producer 3 10 &
./consumer 1 30 &
wait
[1] 7816
[2] 7817
Producer 1: Produced value 1000
[3] 7818
Producer 2: Produced value 2000
Producer 1: Produced value 1001
[4] 7819
Producer 2: Produced value 2001
Producer 2: Produced value 2002
 Consumer 1: Consumed value 1000 from Producer 1
Producer 1: Produced value 1002
 Consumer 1: Consumed value 2000 from Producer 2
Producer 2: Produced value 2003
 Consumer 1: Consumed value 1001 from Producer 1
Producer 3: Produced value 3000
 Consumer 1: Consumed value 2001 from Producer 2
Producer 1: Produced value 1003
 Consumer 1: Consumed value 2002 from Producer 2
Producer 2: Produced value 2004
 Consumer 1: Consumed value 1002 from Producer 1
Producer 3: Produced value 3001
 Consumer 1: Consumed value 2003 from Producer 2
Producer 1: Produced value 1004
 Consumer 1: Consumed value 3000 from Producer 3
Producer 2: Produced value 2005
 Consumer 1: Consumed value 1003 from Producer 1
Producer 3: Produced value 3002
 Consumer 1: Consumed value 2004 from Producer 2
Producer 1: Produced value 1005
 Consumer 1: Consumed value 3001 from Producer 3
Producer 2: Produced value 2006
 Consumer 1: Consumed value 1004 from Producer 1
Producer 3: Produced value 3003
 Consumer 1: Consumed value 2005 from Producer 2
Producer 1: Produced value 1006
 Consumer 1: Consumed value 3002 from Producer 3
Producer 2: Produced value 2007
 Consumer 1: Consumed value 1005 from Producer 1
Producer 3: Produced value 3004
 Consumer 1: Consumed value 2006 from Producer 2
Producer 1: Produced value 1007
 Consumer 1: Consumed value 3003 from Producer 3
Producer 2: Produced value 2008
 Consumer 1: Consumed value 1006 from Producer 1
Producer 3: Produced value 3005
 Consumer 1: Consumed value 2007 from Producer 2
Producer 1: Produced value 1008
 Consumer 1: Consumed value 3004 from Producer 3
Producer 2: Produced value 2009
 Consumer 1: Consumed value 1007 from Producer 1
Producer 3: Produced value 3006
 Consumer 1: Consumed value 2008 from Producer 2
Producer 1: Produced value 1009
[1]   Done                    ./producer 1 10
[2]   Done                    ./producer 2 10
 Consumer 1: Consumed value 3005 from Producer 3
Producer 3: Produced value 3007
 Consumer 1: Consumed value 1008 from Producer 1
Producer 3: Produced value 3008
 Consumer 1: Consumed value 2009 from Producer 2
Producer 3: Produced value 3009
[3]-  Done                    ./producer 3 10
 Consumer 1: Consumed value 3006 from Producer 3
 Consumer 1: Consumed value 1009 from Producer 1
 Consumer 1: Consumed value 3007 from Producer 3
 Consumer 1: Consumed value 3008 from Producer 3
 Consumer 1: Consumed value 3009 from Producer 3
[4]+  Done                    ./consumer 1 30
asteele@DESKTOP-UT49NER:/mnt/c/Users/aspen/OneDrive/College/OS/A3$
```

Figure 2: Test 2 Output — Multiple producers writing to the shared buffer with correct synchronization.

—

Test 3 — Multiple Consumers

Command:

```
./producer 1 20 &
./consumer 1 10 &
./consumer 2 10 &
wait
```

In this scenario, one producer generates twenty items while two consumers process them concurrently. Each consumer receives a unique subset of items, showing that the system distributes work efficiently without overlap or corruption. The results confirm that consumer processes properly respect semaphore order and maintain data consistency.

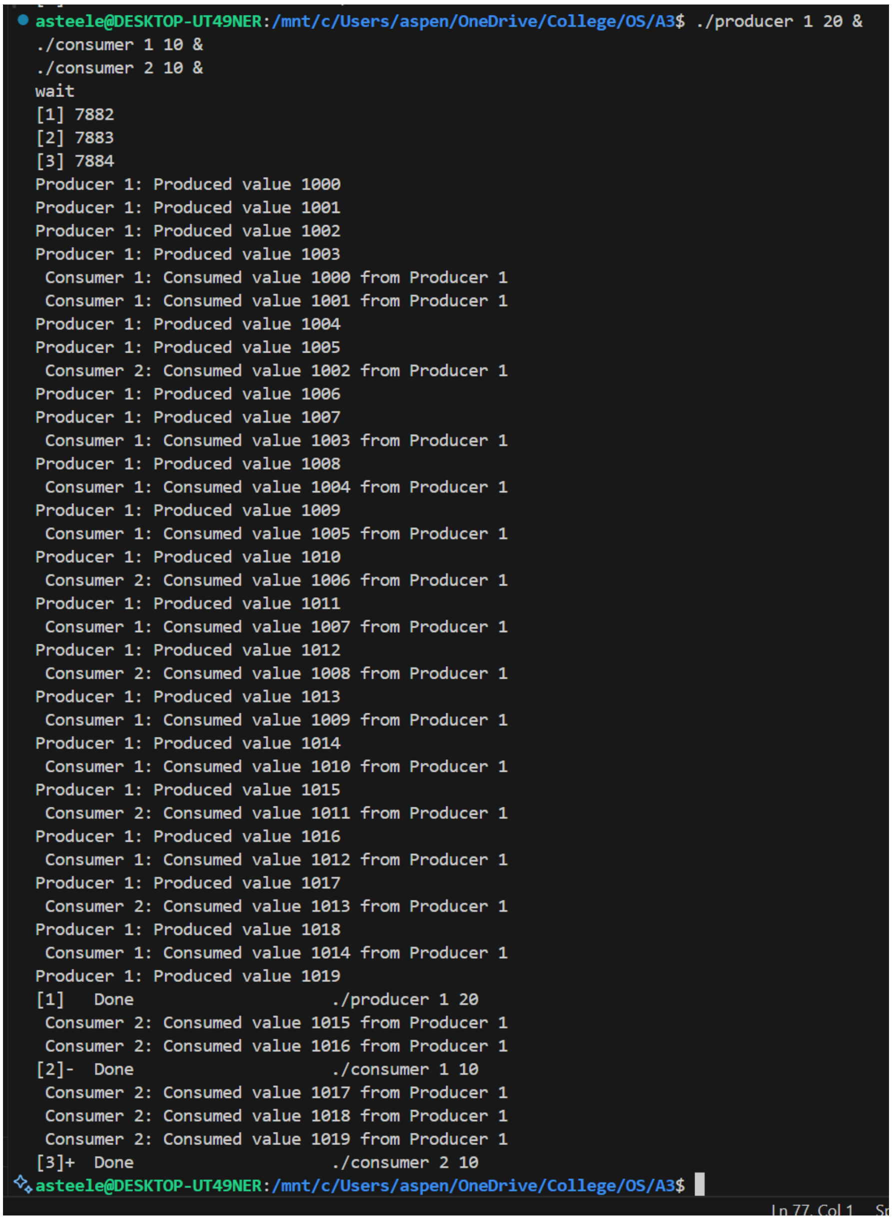


Figure 3: Test 3 Output — Two consumers evenly dividing buffer consumption while maintaining synchronization.

—

Test 4 — Shared Memory and Semaphore Check

Command:

```
ipcs -m
ls -ld /dev/shm/sem.*
```

This step verifies that the **System V shared memory** segment was created successfully and is linked under the IPC key (`0x1234`). It also verifies that the **POSIX named semaphores** (‘sem_empty‘, ‘sem_full‘, and ‘sem_mutex‘) were successfully created in the ‘/dev/shm‘ directory. The presence of all resources confirms proper IPC setup.

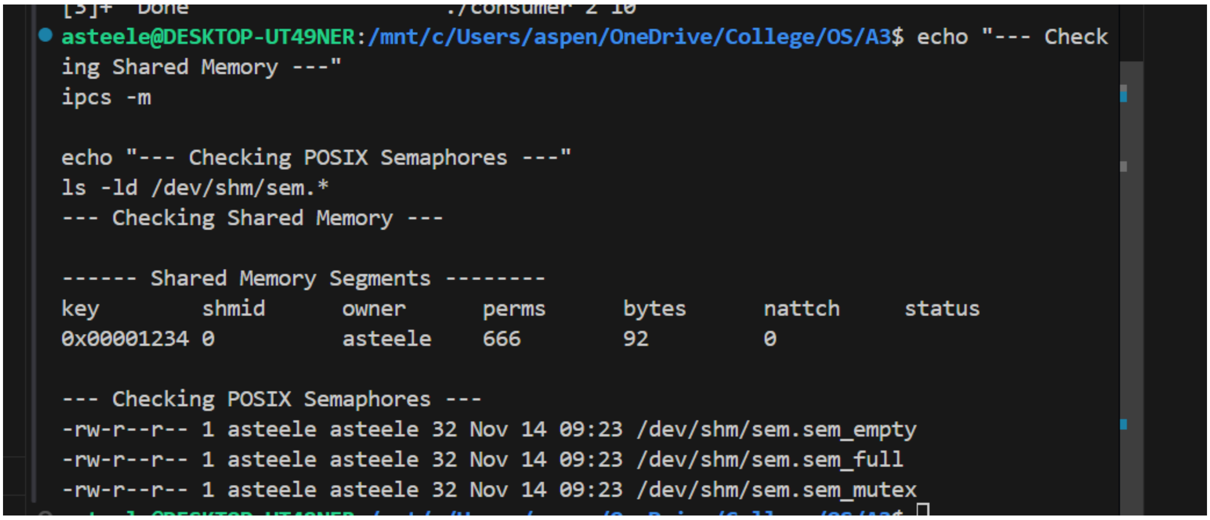


Figure 4: Test 4 Output — Shared memory and POSIX semaphore verification.

—

Test 5 — Cleanup Commands

Command:

```
ipcrm -M 0x1234
rm /dev/shm/sem.*
```

After testing, these commands remove all IPC resources. The `ipcrm` command removes the **System V shared memory** segment (`0x1234`), and the `rm` command removes any **POSIX named semaphores** located in the `/dev/shm` directory. The message `No such file or directory` simply indicates that no leftover POSIX semaphores existed, confirming a clean program shutdown.

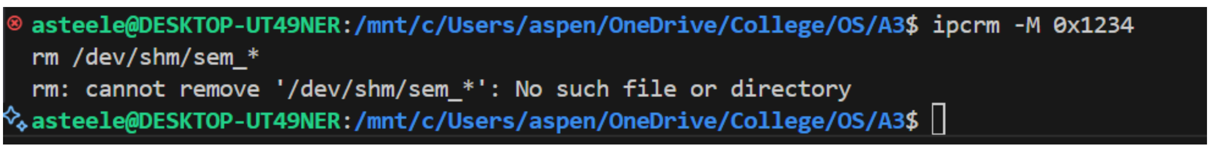


Figure 5: Test 5 Output — Successful cleanup of shared memory and semaphores.

Conclusion

All tests confirm correct synchronization of producers and consumers using shared memory and semaphores. The system performs reliably with concurrent processes, demonstrating a proper understanding of IPC mechanisms and avoiding race conditions or deadlocks.

—

Resources

1. **CS 3502 Assignment 3 — Memory Management and Synchronization Primitives (PDF)** Official assignment document containing specifications and testing requirements.

2. System V IPC documentation: `https://man7.org/linux/man-pages/man7/sysvipc.7.html`

3. Linux Semaphore and Shared Memory manual pages: `man sem_open`, `man shmget`