CSE231 - Operating Systems
Assignment 4
Exercise 4.1

Name: Abhik S Basu
Roll Number: 2020165
Section: A
Branch: CSE

Task: Solving the Producer-Consumer problem in the kernel space

---

I have submitted 2 files namely: p.c & c.c

Approach

In this assignment we need to do mainly 3 things in order for the problem to be solved.

We need to create two sys calls: reader & writer

Before we walk into the details of what is happening, we first tell you what all we are taking globally.

We have made a circular queue of type unsigned long long and initialise it to null. Similarly head and tail are initialised to -1 as initial values. The size is 10 for the queue which is also used for instantiating the producer semaphore.

We instantiate 3 semaphores:

1) sem1: This is essentially a counting semaphore. This is the producer semaphore, we initialise it with the size of our queue. So producer is initially not sending anything to the consumer so its initial values are that

of the queue. And as we send data to the consumer the semaphore updates itself accordingly.

2) sem2: This is essentially a counting semaphore. This is the consumer semaphore, we initialise it with the size zero. So consumer is essentially not receiving anything from the producer so its initial values are that of zero. And as we send data from the producer, the semaphore updates itself accordingly.

3) sem3: This is a binary semaphore and acts like the mutex lock in our solution.It makes sure that the process happening between producer and consumer is mutually exclusive to one of the users and helps in providing exclusive access to one of them hence helping in solving the problem.

Now we implement our reader and writer sys calls. In both of these we only take a single parameter which is an unsigned long long and indicates the number to be enqueued or dequeued.

Both the calls are similar to each other, we first check in both if the circular queue is null or not. If it is null then we initialise the semaphores and queues head and tail as well. Then we essentially just replicate the enqueue and dequeue operations in write and read respectively. We make sure to use our down (wait) and up (notify) operations for the producer and consumer and mutex lock semaphores accordingly to avoid any sort of synchronisation issues.

After this we just add these to our sys call table and click on make and then make the kernel.

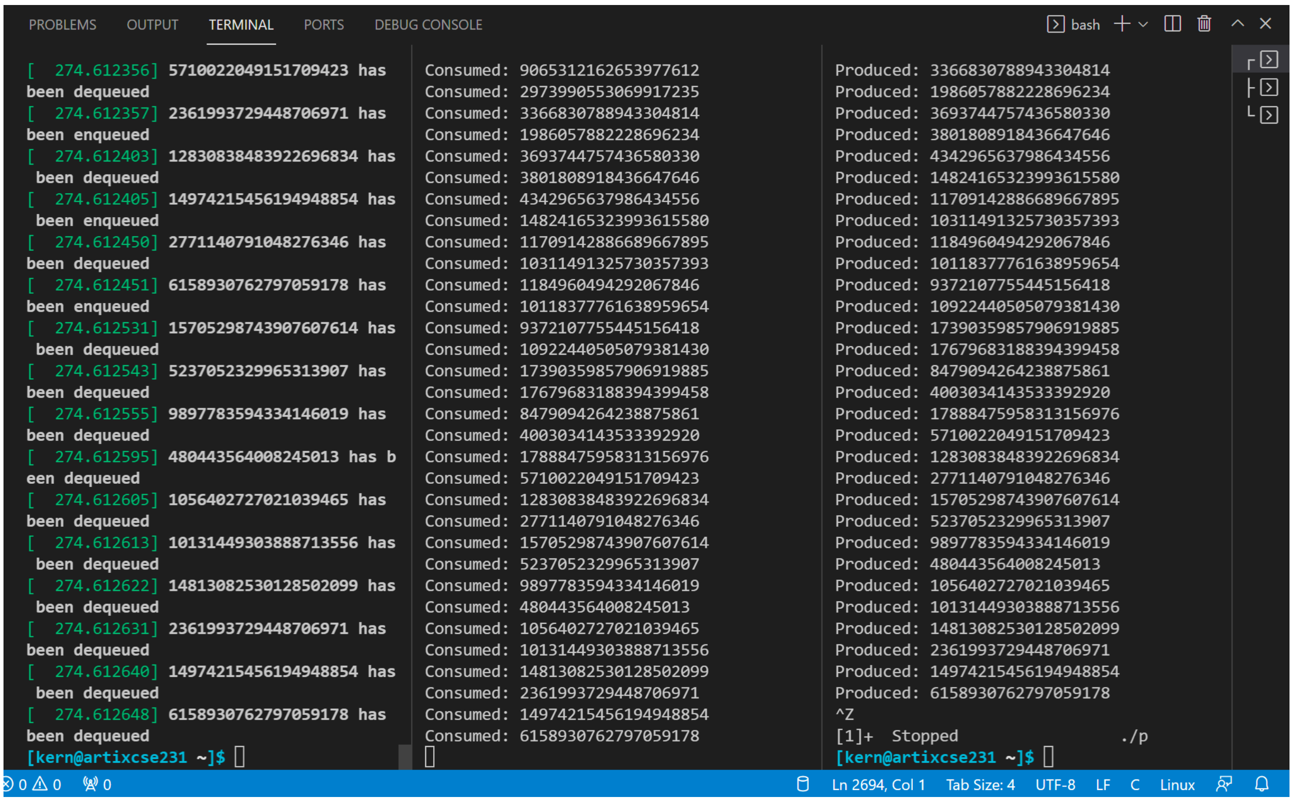
Testing

We now need to test whether our sys call is working or not. For that we create two c programmes.

One is p.c, which is basically the producer, in this we open the /dev/urandom file which is basically a pseudo random number generators. Here we generate our number and then using our syscall we put it into our queue.

In the c.c, which is basically the consumer, we just read this value from our created queue and print it as well.

In order, to make sure that both the sys calls were working, I have also printed the values enqueued and dequeued in the log to make sure that the sys call is working correctly.

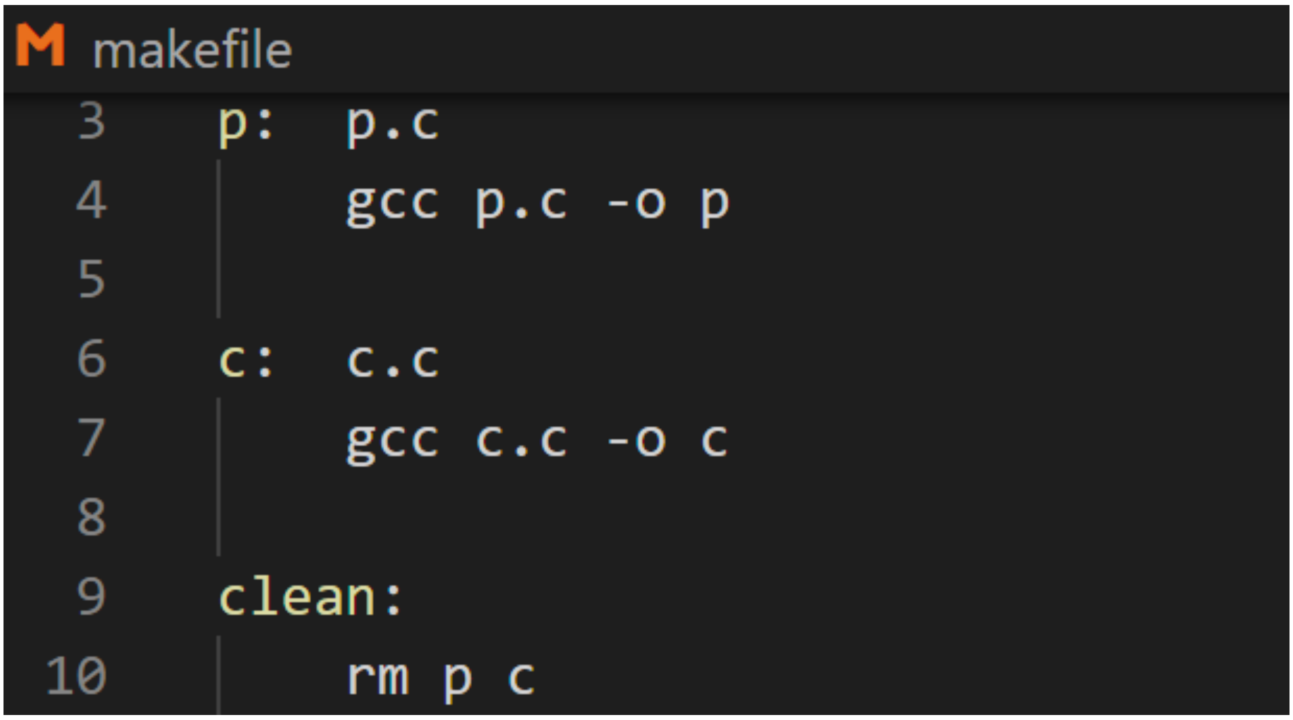


```
[  274.612356] 5710022049151709423 has
been dequeued
[  274.612357] 2361993729448706971 has
been enqueued
[  274.612403] 12830838483922696834 has
 been dequeued
[  274.612405] 14974215456194948854 has
 been enqueued
[  274.612450] 2771140791048276346 has
been dequeued
[  274.612451] 6158930762797059178 has
been enqueued
[  274.612531] 15705298743907607614 has
 been dequeued
[  274.612543] 5237052329965313907 has
been dequeued
[  274.612555] 9897783594334146019 has
been dequeued
[  274.612595] 480443564008245013 has b
een dequeued
[  274.612605] 1056402727021039465 has
been dequeued
[  274.612613] 10131449303888713556 has
 been dequeued
[  274.612622] 14813082530128502099 has
 been dequeued
[  274.612631] 2361993729448706971 has
been dequeued
[  274.612640] 14974215456194948854 has
 been dequeued
[  274.612648] 6158930762797059178 has
been dequeued
[kern@artixcse231 ~]$ 
```

```
Consumed: 9065312162653977612
Consumed: 2973990553069917235
Consumed: 3366830788943304814
Consumed: 1986057882228696234
Consumed: 3693744757436580330
Consumed: 3801808918436647646
Consumed: 4342965637986434556
Consumed: 14824165323993615580
Consumed: 11709142886689667895
Consumed: 10311491325730357393
Consumed: 1184960494292067846
Consumed: 10118377761638959654
Consumed: 9372107755445156418
Consumed: 10922440505079381430
Consumed: 17390359857906919885
Consumed: 17679683188394399458
Consumed: 8479094264238875861
Consumed: 4003034143533392920
Consumed: 17888475958313156976
Consumed: 5710022049151709423
Consumed: 12830838483922696834
Consumed: 2771140791048276346
Consumed: 15705298743907607614
Consumed: 5237052329965313907
Consumed: 9897783594334146019
Consumed: 480443564008245013
Consumed: 1056402727021039465
Consumed: 10131449303888713556
Consumed: 14813082530128502099
Consumed: 2361993729448706971
Consumed: 14974215456194948854
Consumed: 6158930762797059178
```

```
Produced: 3366830788943304814
Produced: 1986057882228696234
Produced: 3693744757436580330
Produced: 3801808918436647646
Produced: 4342965637986434556
Produced: 14824165323993615580
Produced: 11709142886689667895
Produced: 10311491325730357393
Produced: 1184960494292067846
Produced: 10118377761638959654
Produced: 9372107755445156418
Produced: 10922440505079381430
Produced: 17390359857906919885
Produced: 17679683188394399458
Produced: 8479094264238875861
Produced: 4003034143533392920
Produced: 17888475958313156976
Produced: 5710022049151709423
Produced: 12830838483922696834
Produced: 2771140791048276346
Produced: 15705298743907607614
Produced: 5237052329965313907
Produced: 9897783594334146019
Produced: 480443564008245013
Produced: 1056402727021039465
Produced: 10131449303888713556
Produced: 14813082530128502099
Produced: 2361993729448706971
Produced: 14974215456194948854
Produced: 6158930762797059178
^Z
[1]+  Stopped                 ./p
[kern@artixcse231 ~]$ 
```

To make all the files, we have to run the command make



makefile

```
p:	p.c
	gcc p.c -o p

c:	c.c
	gcc c.c -o c

clean:
	rm p c
```

To run the code, open 3 terminals and put the following commands:

```
./p
./c
sudo dmesg
```