Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de ciencias computacionales.

Ingeniería en computación

Seminario de Solución de Problemas de Sistemas Operativos - D01.

Violeta del Rocío Becerra Velázquez

“Programa 6. Productor-Consumidor”

2023A

10/04/2023

Flores Estrada Abraham Miguel Angel

Olguín Hernández Jair Benjamín

Índice.

Objetivo.

Recordemos que el problema de productor-consumidor es un problema de sincronización en sistemas operativos que surge cuando un conjunto de procesos productores genera datos y los depositan en un búfer, y otro conjunto de procesos consumidores retiran los datos del búfer para su uso. El objetivo es garantizar que los procesos productores y consumidores trabajen en armonía para evitar condiciones de carrera y otros problemas de concurrencia. El problema consiste en que dos procesos, productor y consumidor, ambos comparten un búfer de tamaño finito. La tarea del productor es generar un producto, almacenarlo y comenzar nuevamente; mientras que el consumidor toma (simultáneamente) productos uno a uno. El problema consiste en que el productor no añada más productos que la capacidad del buffer y que el consumidor no intente tomar un producto si el buffer está vacío.

Por lo que buscamos crear un programa que logre simular este problema, donde nosotros planeamos hacerlo de manera gráfica para que sea más fácil de entender y visualizar el cómo está presentado el problema. Donde el programa deberá contar con las validaciones necesarias para ver que el productor y consumidor puedan cumplir con sus condiciones para producir y consumir respectivamente. Tales como que el productor no puede producir más de la cantidad del contenedor que en este caso tiene una capacidad de 22, y el consumidor no pueda entrar si no hay productos, donde cumpla con los requisitos.

Lenguaje utilizado.

Haremos uso de python debido a la facilidad que nos da para crear las interfaces y así mismo poder hacer que la lista funcione de manera circular, así mismo gracias a las distintas funciones de python podemos tener más control de la validaciones necesarias.

Desarrollo del programa.

En este caso debido al programa se hizo una interfaz totalmente nueva donde en el productor será representado como el internet, el consumidor será representado como un DJ, donde el internet irá produciendo música y el DJ irá consumiendo la

música. En este caso el programa solamente termina al presionar la tecla ESC, por lo que debe continuar hasta que sea señalado, por lo que tenemos una bandera que nos ayudará detectar cuando sea presionada la tecla, asi mismo tenemos tres variables que nos servirán como apuntadores para saber en qué posición se encuentran el productor y consumidor; y quien es es el último en entrar al buffer. También tenemos un arreglo que es el que usaremos como lista circular, donde lo inicializamos en 0.

```
self.Running=True;
ConsumerPointer=0;
ProducerPointer=0;
content = [0] * 22;
last="";
```

Una vez con esto, lo que hacemos es escoger quién irá primero en entrar al buffer, donde nosotros generamos dos números y el que sea mayor será el primero en entrar.

```
while self.Running == True:
    numconsum = random.randint(1,1000);
    numprod = random.randint(1,1000);
    while numconsum == numprod:
        numconsum = random.randint(1,1000);
        numprod = random.randint(1,1000);
```

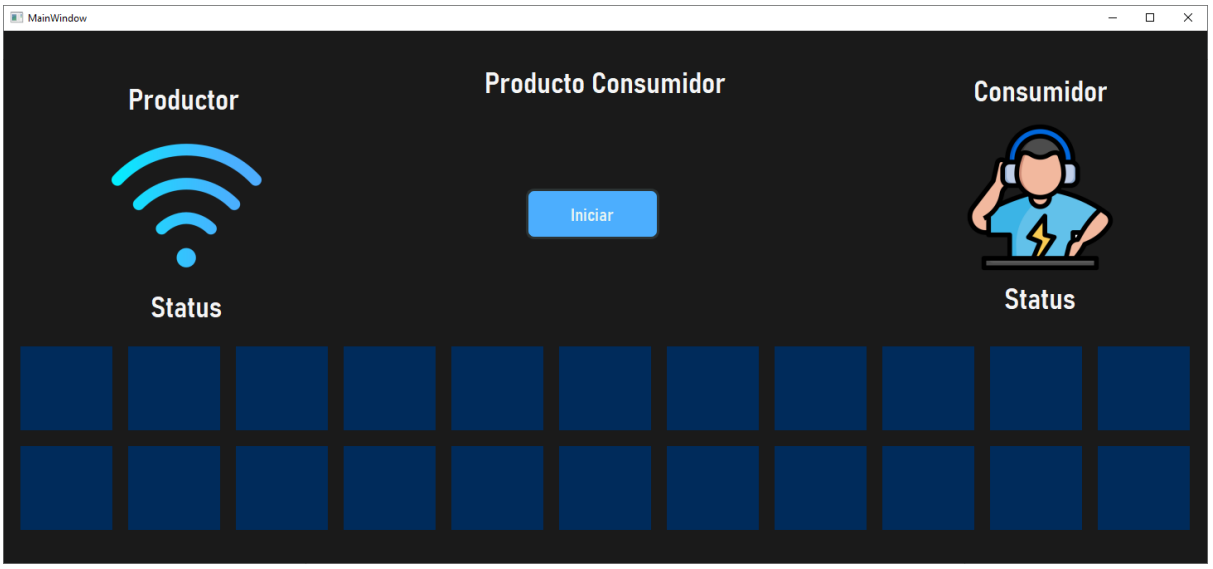
Después de esto solamente comparamos si el número de productor es mayor, en dado caso le asignamos a la variable last “Productor” (que es la encargada de saber quien fue el último en entrar) después genera un numero random entre 3 y 6, que será la cantidad de productos a generar. Después iteramos sobre el número generado, donde lo que hacemos es marcar con 1 los productos generados, ya que tenemos un arreglo con 22 espacios que han sido marcados con 0, por lo que irá marcando con 1 los que ha generado. Y así mismo validamos que si en la posición que se encuentra el productor está en 1, no genere más y si llega a ser 22 vuelva a la posición 0.

```
if numprod > numconsum:
    last="Productor"
    numprod = random.randint(3,6);
    for i in range(numprod):
        if content[ProducerPointer]==1:
            self.StatusProductor.setText("Intentando")
            break;
        else:
            self.StatusProductor.setText("Produciendo")
            content[ProducerPointer]=1;
            ProducerPointer+=1;
            if ProducerPointer==22:
                ProducerPointer=0;
```

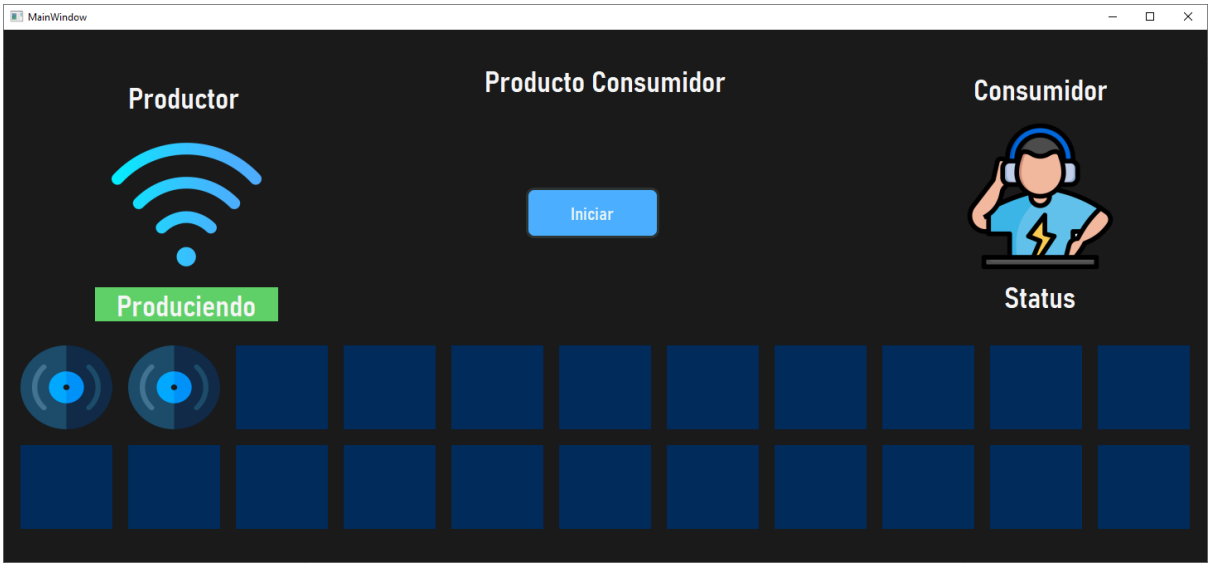
Y en dado caso que le toque al consumidor primero lo que haremos es lo contrario, marcar last con “Consumidor” y verificar que la posición donde se encuentra el consumidor sea diferente de 0, en dado caso que sea 1, marcamos la posición con 0 y checamos que no pase de 22.

```
else:
    last="Consumidor"
    numconsum = random.randint(3,6);
    for i in range(numconsum):
        if content[ConsumerPointer]==0:
            self.StatusConsumidor.setText("Intentando")
            QtTest.QTest.qWait(1000)
            break;
        else:
            self.StatusConsumidor.setText("Consumiendo")
            content[ConsumerPointer]=0;
            ConsumerPointer+=1;
        if ConsumerPointer==22:
            ConsumerPointer=0;
```

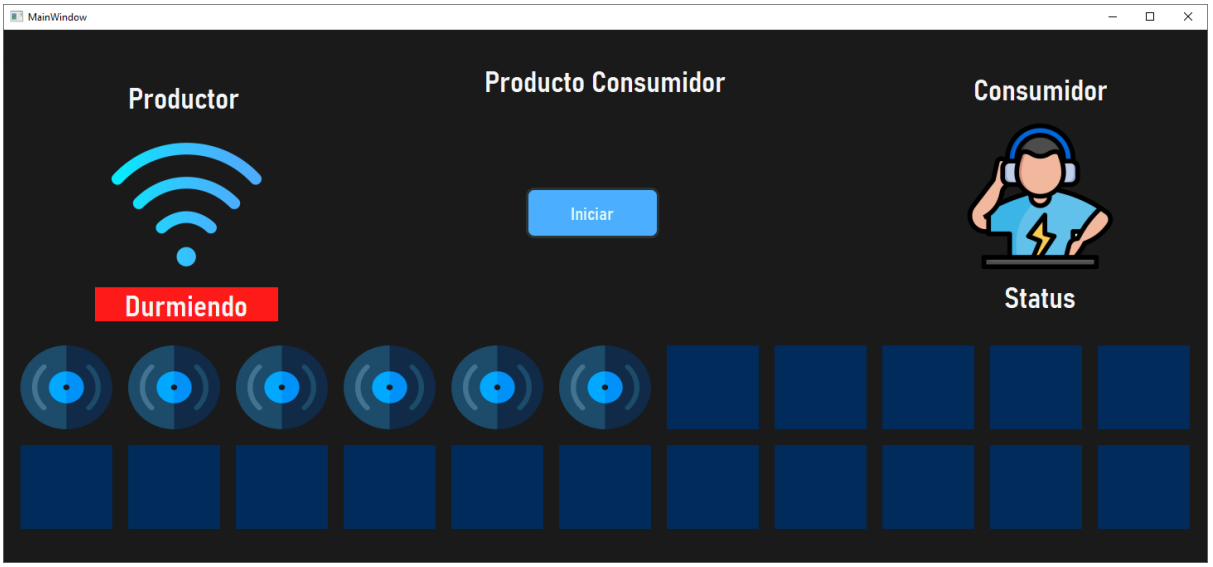
Vemos la ejecución del programa, donde lo primero que vemos son los 22 espacios, así como el respectivo producto y consumidor. Estos tienen un apartado el cual nos dirá si están activos, intentando o durmiendo.



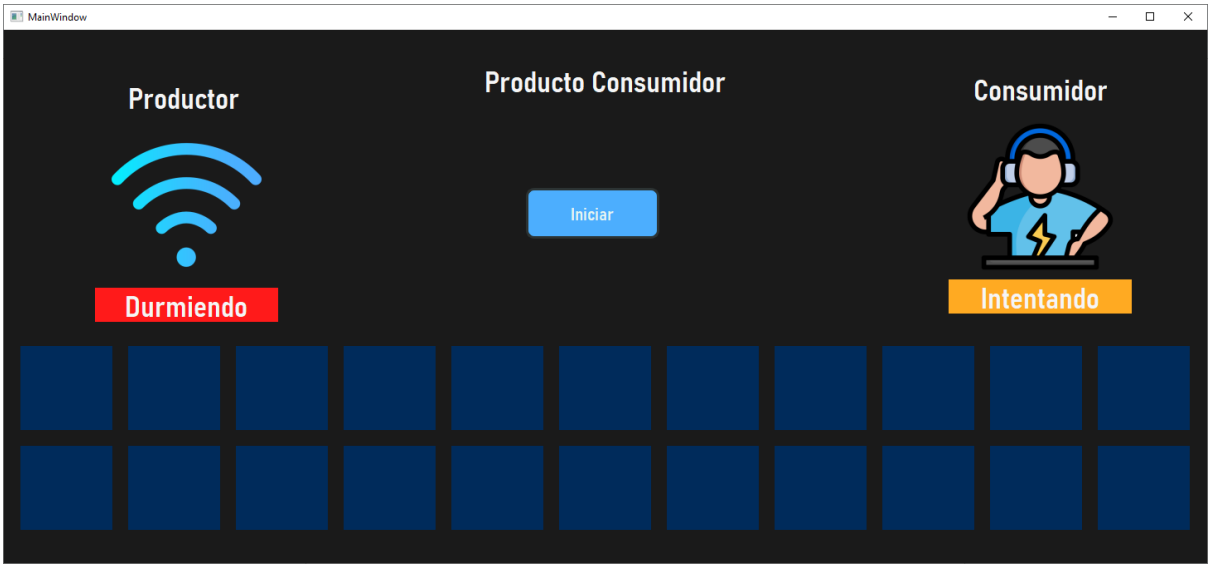
Una vez iniciamos el programa, lo que hará es escoger quién irá primero, donde en este caso le tocó al productor primero, por lo que sin problemas puede producir.



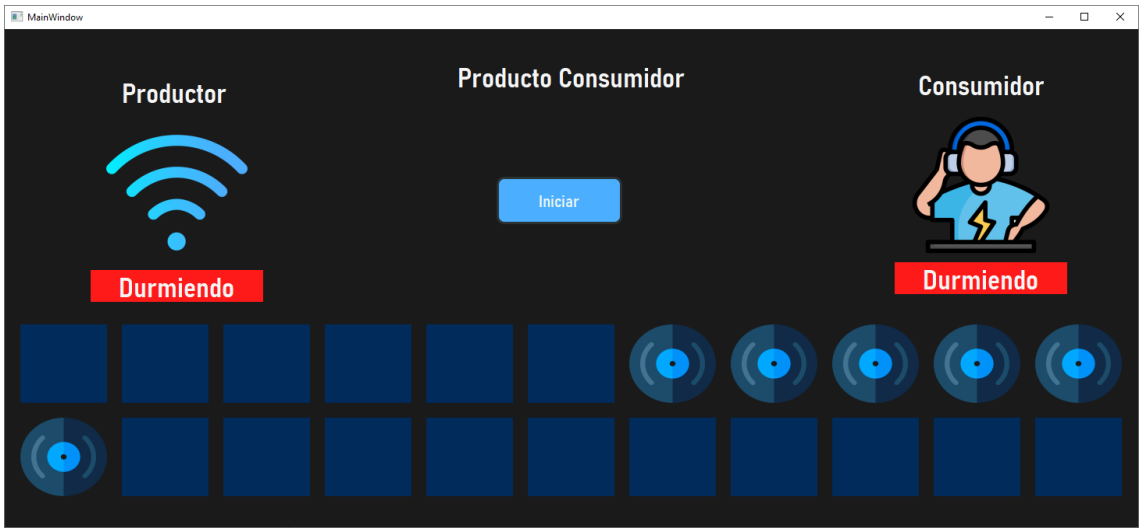
Una vez que termine, aparecerá como durmiendo y se escoge nuevamente quien le tocara entrar, donde puede entrar nuevamente el productor en caso contrario el consumidor.



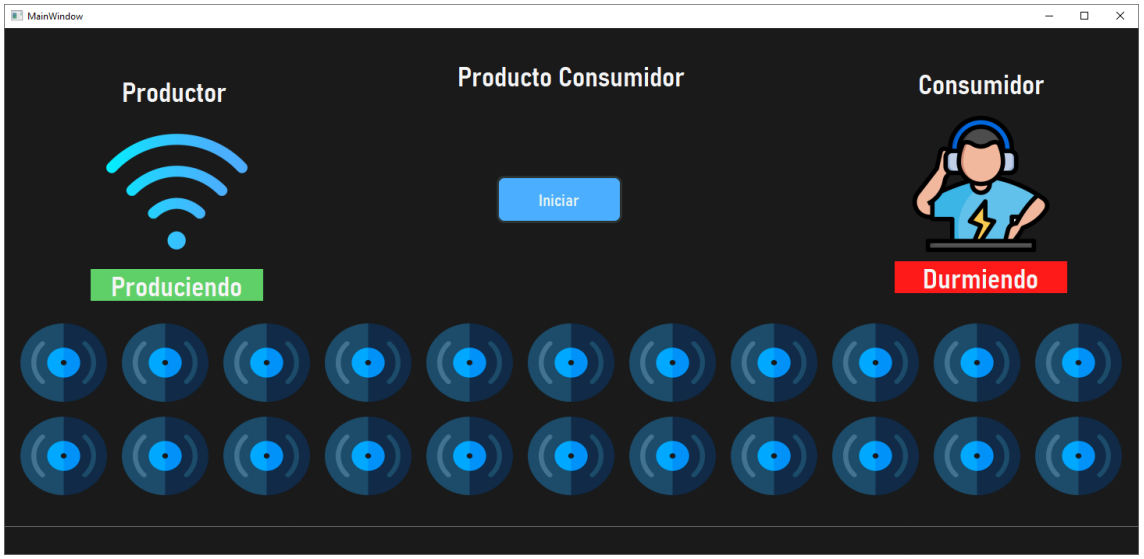
Después vuelve a elegir, y en este caso podemos ver que le tocó al consumidor y consumió todos los discos, y nuevamente le volvió a tocar pero como ahora no hay productos aparece como intentando pero no podrá entrar.



Después de esto nuevamente le tocó al productor y podemos ver que se quedó en la posición del último disco que hizo. En este caso podemos ver que es la casilla 7.



Y como podemos de igual forma una vez que agregue un producto a la casilla 22,



comenzará nuevamente en la 1. De igual forma el producto puede tener todas las casillas, pero no hará más hasta que se libere algún espacio.

Conclusiones.

El desarrollo de este programa fue más sencillo de lo que pensábamos, ya que con la investigación anterior a este programa ya conocíamos acerca del problema y de distintas soluciones planteadas, por lo que nosotros optamos por la más sencilla que es el uso de apuntadores que nos ayuden a saber la posición en que se encuentren el productor y consumidor, y en este caso nosotros solo verificamos si la posicion tenia 0 o 1, y con esto permitir el acceso a consumir o producir. Y al hacerlo con una

interfaz es más fácil de entender el cómo funciona el programa y el cómo es que debemos tener control acerca de estos problemas.