Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales.

Ingeniería en Computación

Actividad 10.

Programa 5. Productor - Consumidor

Materia: Sistemas Operativos ***Sección:*** D04

Profesor: Becerra Velázquez Violeta del Rocío.

Equipo:

- Carriola Navarro Aldo Fernando | 221978167
- Cervantes Araujo Maria Dolores | 217782452

Fecha: 05/11/2023

Índice

Tabla de Imágenes

Programa 5. Productor – Consumidor

Objetivos

Para este quinto programa se busca crear un contenedor con un espacio de 20 elementos, de los cuales a la hora de impresión de pantalla se mostrará cada elemento con su respectivo numero de espacio dentro del buffer, en nuestro caso decidimos poner dos guiones bajos para representar un espacio 'limpio'/'consumido'.

Una vez cumplido lo anterior, se desea incorporar la funcionalidad de que exista un 'productor' y un 'consumidor'. Donde existirían tres estados posibles para cada caso: 'durmiendo', 'intentando' y 'produciendo' (en el caso del productor) o 'consumiendo' (en el caso del consumidor).

Donde al estar intentando el productor verificará inicialmente desde la primera posición del contenedor si ese espacio esta 'vacío' para poder producir; en caso de ser afirmativo lo anterior, este procedería a 'producir', representándolo con un asterisco en dicho espacio. Por otro lado, cuando el consumidor este intentando, verificará inicialmente que el primer espacio cuente con algo para 'consumirlo', y en caso de poder, modificar el asterisco representado en ese espacio por nuevamente dos guiones bajos.

Siendo así que, para ambos casos, una vez llegando a la última posición de nuestro contendor se debe incorporar la funcionalidad, que vuelva al inicio en caso de poder seguir produciendo o consumiendo respectivamente.

Además, para poder escoger quien entrará a intentarlo y quien dormirá se incorporaron números aleatorios donde además obtendremos un número del 4 al 7, siendo la cantidad de espacios por intentar producir o consumir, sea el caso que se encuentre en el momento. Finalmente, teniendo en cuenta todo lo anterior obtenemos que pueden existir 5 estados en combinación: *Productor intentando y consumidor durmiendo, Productor produciendo y consumidor durmiendo, Productor durmiendo y consumidor intentando, Productor durmiendo y consumidor consumiendo, Productor durmiendo y consumidor durmiendo.*

Finalmente, una vez incorporada la lógica anteriormente planteada, se busca que el programa sea infinito y autónomo, hasta que se presioné la tecla 'esc'.

Notas Sobre el Lenguaje

El lenguaje utilizado para llevar a cabo el programa fue *Python*, dado que nos permite experimentar con diversas funciones que apoyan en la creación del programa, permitiéndonos que sea orientado a objetos, estructural, con interfaz, a consola, con color e incluso animación.

Dado que es un lenguaje de nuestro total conocimiento y dominio para ambos, optamos por crear el programa de forma estructural, es decir, mediante funciones. Buscando siempre mediante acuerdos, que ambos nos sintiéramos familiarizados con el lenguaje y lógica del programa, de esta manera optimizamos tiempos y nos fue más sencillo modelar y visualizar el programar conforme lo esperado.

TDA o Estructuras Utilizadas en el Programa

Para esta práctica dadas las instrucciones, decidimos conformar el programa con 3 funciones y todo hacerlo recursivo para que siga el ciclo hasta que el usuario presione la tecla "*esc*".

```python
import os
from tabulate import tabulate
import random as rd
import msvcrt as mv
import threading, time
import keyboard as kb

idProductor = 0
idConsumidor = 0
rangoBuffer=20
Buffer = ['__'] * rangoBuffer

def imprimir(productor,consumidor):
    os.system("cls")
    print("\nBuffer")
    headersB = list(range(rangoBuffer))
    proceso_bloq = [[Buffer[i] for i in headersB]]
    print(tabulate(proceso_bloq, headers=headersB, tablefmt="double_outline"))
    # print("Buffer en forma de lista",Buffer,"\n")
    print("Productor está",productor)
    print("Consumidor está",consumidor)
```

Ilustración 1. Función para imprimir.

Declaramos las librerías que usaremos a lo largo del algoritmo y las variables globales iniciales para comenzar el proceso, así como la función de imprimir que se llamara en al momento de producir o consumir objetos.

```python
def verificar_esc():
    #while True:
    if kb.is_pressed('esc'):
        print("La tecla 'esc' fue presionada. Saliendo del programa.")
        os._exit(0)
```

Ilustración 2. Función para verificar.

Para esta función colocamos la evaluación para verificar si durante el ciclo el usuario digita *"esc"* con ayuda de la librería *keyboard* para detectar las teclas.

```
def productorConsumidor():
    global Buffer
    global idProductor
    global idConsumidor

    volado =  rd.randint(1,3)
    numeros = rd.randint(4,7)

    #Productor = 1
    if volado == 1:
    # print("\n\nProductor: ", numeros)
        while True:
            imprimir('intentando','durmiendo')
            for n in range(numeros):
                verificar_esc()
                elemento = Buffer[idProductor]
                if elemento != '✓':
                    Buffer[idProductor] = '✓'
                    idProductor = (idProductor+1)%len(Buffer)
                    imprimir('produciendo','durmiendo')
                    verificar_esc()
                    time.sleep(1)
                # time.sleep(1)
                else:
                    break

            break
```

Ilustración 3. Función productoConsumidor parte1.

```
    #Consumidor = 2
    elif volado == 2:
    # print("Consumidor: ", numeros)
        while(True):
            imprimir('durmiendo','intentando')
            for n in range(numeros):
                verificar_esc()
                elemento = Buffer[idConsumidor]
                if elemento != '__':
                    Buffer[idConsumidor] = '__'
                    idConsumidor = (idConsumidor+1)%len(Buffer)
                    imprimir('durmiendo','consumiendo')
                    verificar_esc()
                    time.sleep(1)
                else:
                    break
            break

    # Dormidos = 3
    elif volado == 3:
        verificar_esc()
        imprimir('durmiendo','durmiendo')


def main():
    while True:
        verificar_esc()
        productorConsumidor()


main()
```

Ilustración 4. Función productorConsumidor parte2.

En esta función declaramos todo el cuerpo del programa, donde se declaran las opciones para el ciclo del programa, que puede estar produciendo objetos, consumiendo objetos o que pueden estar durmiendo ambos estados (productor y consumidor) en el cual se declara el buffer del algoritmo para que solo permita hasta 20 objetos y a partir de ahí que se cicle, volviéndose recursivo.

Finalmente se declara la función principal y se manda llamar al final, de esta manera se repite hasta que el usuario decida salir mediante la tecla.

Funcionamiento

Ilustración 5. Estado durmiendo.

Ambos estados al momento del volado fue un estado de “durmiendo”.

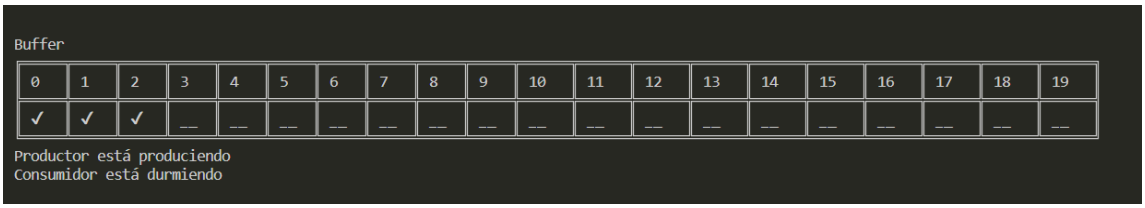

Ilustración 6. Estado produciendo.

Posteriormente en un segundo volado el buffer se llenó de objetos por el productor.

Ilustración 7. Estado consumiendo.

El ciclo continuo y en el siguiente volado fue el turno de consumir vaciando el buffer,

Ilustración 8. Saliendo del programa.

Continuo el programa de manera recursiva hasta que presionamos la tecla “esc” y sale del ciclo, mostrando el siguiente mensaje en pantalla.

Errores y Soluciones

Contador de última posición del consumidor: Este error realmente fue un descuido, ya que al programar y ejecutar por primera vez el programa (con una impresión de pantalla muy básica), como tal si asignaba correctamente números aleatorios, pero notamos que al tener que consumir (y de ser esto posible) no lo hacía así como se observa en la Ilustración 9.

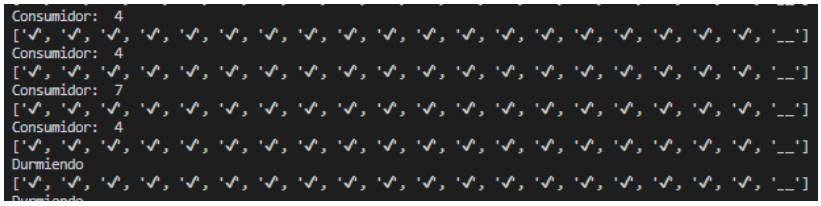

Ilustración 9.Error 1

Tras ello, analizamos las variables que teníamos dentro de nuestra función 'productorConssumidor()', obteniendo la causa de error, como se observa en la Ilustración 10.

```
29      if volado == 1:
30          print("Productor: ", numeros)
31          #if Buffer[:] != '✓':
32          while True:
33              for _ in range(numeros):
34                  elemento = Buffer[idProductor]
35                  if elemento != '✓':
36                      Buffer[idProductor] = '✓'
37                      idProductor = (idProductor+1)%len(Buffer)
38                      print(idProductor)
39                  else:
40                      break
41
42              break
43
44                  #val = True
45
46      elif volado == 2:
47          print("Consumidor: ", numeros)
48          #if Buffer[:] != '_':
49          while(True):
50              for _ in range(numeros):
51                  elemento = Buffer[idProductor]
52                  if elemento != '__':
53                      Buffer[idProductor] = '__'
54                      idProductor = (idProductor+1)%len(Buffer)
55
56                  else:
57                      break
58              break
```

Ilustración 10.Causa del Error 1

Es decir, como se puede observar anteriormente, lo que estábamos haciendo era redundancia donde siempre le mandamos a hablar al ID Productor, siendo que sería necesario solo cambiar la variable en el caso consumidor a su ID, así como se ve en la Ilustración 11.

```
46      elif volado == 2:
47          print("Consumidor: ", numeros)
48          #if Buffer[:] != '_':
49          while(True):
50              for _ in range(numeros):
51                  elemento = Buffer[idConsumidor]
52                  if elemento != '__':
53                      Buffer[idConsumidor] = '__'
54                      idConsumidor = (idConsumidor+1)%len(Buffer)
```

Ilustración 11.Solución Error 1

Salir del programa en cualquier momento: En los requerimientos del programa, claramente se encontraba el que la ejecuciónd el mismo será de forma infinita hasta que el usuario presioné la tecls ‘esc’, sin embargo, al incoporar lo siguiente en nuestra base anteriromente mostrada en la solución del error 1, como hicimos uso de time sleep, para poder ver la ejecución correcta del programa, notamos que al intentar salir del programa en ejecución no podíamos salir del mismo, ya que, entraba en estado ‘sleep’ y no detectaba la tecla al ser presionada en cualquier momento, como tal no se puede poner una Ilustración para mostrar este error, debido a que este involucra al comportamiento del teclado físico, sin embargo, en la Ilustración 12, mostramos la causa del error.

```
14  def imprimir(productor,consumidor):
15      os.system("cls")
16      print("\nBuffer")
17      headersB = list(range(rangoBuffer))
18      proceso_bloq = [[Buffer[i] for i in headersB]]
19      print(tabulate(proceso_bloq, headers=headersB, tablefmt="double_outline"))
20      print("Productor está",productor)
21      print("Consumidor está",consumidor)
22      time.sleep(2)
23
24
25  def verificar_esc():
26      if kb.is_pressed('esc'):
27          print("La tecla 'esc' fue presionada. Saliendo del programa.")
28          os._exit(0)
29
30  def productorConsumidor():
31      global volado
32      global numeros
33      global Buffer
34      global val
35      global idProductor
36      global idConsumidor
37
38      val = False
39      volado = rd.randint(1,3)
40      numeros = rd.randint(4,7)
41
42      #Productor = 1
43      if volado == 1:
44          while True:
45              imprimir('intentando','durmiendo')
46              for n in range(numeros):
47                  verificar_esc()
48                  elemento = Buffer[idProductor]
49                  if elemento != '✓':
50                      Buffer[idProductor] = '✓'
51                      idProductor = (idProductor+1)%len(Buffer)
52                      imprimir('produciendo','durmiendo')
53                  else:
54                      break
55              break
```

Ilustración 12.Causa del Error 2

Sin embargo, tras considerar el error anterior decidimos generar más llamadas a la función verificar_esc(), y considerar donde llamamos a hablar el time sleep, para así llegar a la solución mostrada en la Ilustración 13.

```
55      if volado == 1:
56      # print("\n\nProductor: ", numeros)
57          while True:
58              imprimir('intentando','durmiendo')
59              for n in range(numeros):
60                  verificar_esc()
61                  elemento = Buffer[idProductor]
62                  if elemento != '✓':
63                      Buffer[idProductor] = '✓'
64                      idProductor = (idProductor+1)%len(Buffer)
65                      imprimir('produciendo','durmiendo')
66                      verificar_esc()
67                      time.sleep(1)
```

Ilustración 13.Solución Error 2

Conclusiones

Maria Dolores.

Comprendimos el uso de los hilos y los semáforos mediante este algoritmo que pese que no se utilizan hacen el manejo de la simulación mediante el estado de espera entre cada posible volado, en un principio se implementaron hilos para la detección de la tecla y el tiempo de espera que se interponía entre cada ejecución, pero debido a que nos llegó a ocasionar varios errores con el algoritmo optamos por modificarlo a la manera tradicional.

Fue una actividad enriquecedora algo fuera de la rutina por la repetición del programa con procesos; me gusto la dinámica y fue todo un reto implementar hilos, pero al final se logró de manera satisfactoria de la mano de mi compañero que me colaboro en la creación de la lógica para este algoritmo, esperando futuras entregas.

Aldo Fernando.

Realmente, al conocer las especificaciones del programa supe junto con mi compañera que sería algo sencillo, pero que se podría prestar a múltiples formas para poder llegar al resultado deseado, siendo que, a recomendación de la maestra se podían usar semáforos, por otro lado, podíamos hacer uso de funciones o simplemente de condicionales, siendo esta última por la cual nos iríamos, ya que teníamos poco conocimiento respecto a la implementación e semáforos en Python y al intentarlo a pesar de tenerlo ordenado lógicamente, se presentaba diversos problemas.

Por otro lado, al final a mi parecer se logró sin tantas dificultades llegar al resultado deseado, asimismo, obteniendo los aprendizajes esperados respecto a hilos y semáforos.