Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales

Sistemas Operativos

Profesora: Becerra Velázquez Violeta del Rocío

Alumno: Monjaraz Briseño Luis Fernando

Código: 218520958

Carrera: Ingeniería en Computación

Sección: D04

Actividad 10 Programa 5 Productor-Consumidor

Fecha: 22/10/2023

Índice

Tabla de imágenes

Datos personales

Nombres: Lomeli Navarro Jaime Adalberto y Monjaraz Briseño Luis Fernando.

Códigos: 221978094 y 218520958.

Correos: jaime.lomeli7809@alumnos.udg.mx y luis.monjaraz5209@alumnos.udg.mx

Datos de la materia

Materia: Sistemas Operativos

Sección: D04

Horario: Martes, Jueves, Sábado. 11:00 a 12:55

NRC: 204880

Clave: IL366

Número de }actividad

Programa 5. Productor-Consumidor.

Objetivo de la actividad

El objetivo de esta practica es recrear el funcionamiento de un Productor-Consumidor, en este caso el productor y el consumidor estarán generando o eliminando de 4 a 7 procesos. Esto mediante el uso de hilos. Cuando el productor ya no pueda producir mas marcara que esta durmiendo, esto igual aplica al consumidor. La única forma de cerrar el programa (aparte de darle a la x o usar alt+f4) es con la tecla “ESC”, si no el código continuara indefinidamente. Esta práctica ayudara a la comprensión del “Productor Consumidor”.

En resumen, esta actividad es sumamente útil para comprender la teoría del Productor-Consumidor y el uso de los hilos.

Nota: Esta actividad fue realizada en equipo: Lomeli Jaime y Monjaraz Luis.

Notas acerca del lenguaje

Lenguaje usado: Python.

Motivo: Es el lenguaje en común que mejor sabemos utilizar, además con Python es mucho más sencillo utilizar los hilos, pues en C++ se necesita un compilador de x64, modificar el JSON y otra serie de pasos muy rebuscados, mientras que en Python tan solo escribimos la librería “threading” y listo ya se pueden utilizar hilos. Y si utilizamos hilos para la actividad, en teoría se podría hacer solamente simulado, sin embargo, preferimos utilizar hilos para tener la experiencia “completa” de crear un productor-consumidor. Cabe aclarar que no estamos usando un semáforo para controlar el acceso al buffer, si no que es con el uso de variables globales, los cuales actúan como una especie de semáforo “sintético”.

Estructuras: Realmente en esta actividad no utilizamos una estructura, sin embargo, si utilizamos un total de 5 librerías y dos clases (POO), siendo Productor y Consumidor, los cuales se utilizan los dos hilos.

El código a diferencia de los anteriores es mucho más corto pues solo consta de 75 líneas (si quitamos comentarios se puede disminuir a 69 líneas), el motivo de que es tan corto es meramente por el uso de hilos y que no estamos empleando una estructura detalla para el apartado visual.

Clases:

- Productor: Esta clase contiene una función, la clase esta "conectada" a un hilo, se encarga del manejo del productor, creando "datos" para el buffer, esto ocurrirá en un lapso de cada 3 a 6 segundos (puede ser modificado a un rango menor o mayor). El productor generara alguno de estos caracteres aleatoriamente: 'a', 'b', 'c', '1', '2', '3', '#', '@' o '&'.
- Consumidor: Esta clase contiene una función, la clase esta "conectada" a un hilo, se encarga del manejo del consumidor, consumiendo "datos" del buffer, esto ocurrirá en un lapso de cada 4 a 8 segundos (puede ser modificado a un rango menor o mayor).

Funciones:

- Run (Productor): Esta es la función que utiliza la clase para realizar todos los procesos mencionados en la clase, esto incluye, los tiempos de espera, la impresión del buffer, el estado de durmiendo, limpieza de pantalla, etcétera.
- Run (Consumidor): Esta es la función que utiliza la clase para realizar todos los procesos mencionados en la clase, esto incluye, los tiempos de espera, la impresión del buffer, el estado de durmiendo, limpieza de pantalla, etcétera.
- check_for_escape_key: Esta clase es la que se encarga de cerrar el programa en cuando se pulsa la tecla "ESC" aunque para que se cierre se debe de esperar el tiempo del sleep utilizado, que puede variar dependiendo de cuanto le falte para producir o consumir.

```
5   @author: Jaime Lomeli y Monjaraz Luis
6   """
7
8   import threading
9   import time
10  import random
11  import keyboard
12  import os
```

Código 1 Librerías.

```python
# Buffer
buffer = [' ']*20
indice_productor = 0
indice_consumidor = 0

# Variable para controlar si el programa debe continuar ejecutándose
running = True
```

Código 2 Buffer e índices del productor-consumidor.

```python
class Productor(threading.Thread):
    def run(self):
        global buffer
        global indice_productor
        while running:
            elementos_producir = random.randint(4, 7)
            for _ in range(elementos_producir):
                if buffer[indice_productor] == ' ':
                    buffer[indice_productor] = random.choice(['a', 'b', 'c', '1', '2', '3', '#', '@', '&'])
                    print('\n')
                    print(f'Productor trabajando, produjo: {buffer[indice_productor]}')
                    indice_productor = (indice_productor + 1) % 20
                else:
                    print('Productor durmiendo')
                    break
            print(f'Buffer: {list(enumerate(buffer, start=1))}')
            time.sleep(random.randint(3, 6))
            os.system('cls' if os.name == 'nt' else 'clear')  # Limpiar pantalla
```

Código 3 Productor.

```python
class Consumidor(threading.Thread):
    def run(self):
        global buffer
        global indice_consumidor
        while running:
            elementos_consumir = random.randint(4, 7)
            for _ in range(elementos_consumir):
                if buffer[indice_consumidor] != ' ':
                    print('\n')
                    print(f'Consumidor trabajando, consumió: {buffer[indice_consumidor]}')
                    buffer[indice_consumidor] = ' '
                    indice_consumidor = (indice_consumidor + 1) % 20
                else:
                    print('Consumidor durmiendo')
                    break
            print(f'Buffer: {list(enumerate(buffer, start=1))}')
            time.sleep(random.randint(4, 8))
            os.system('cls' if os.name == 'nt' else 'clear')
```

Código 4 Consumidor.

```python
# Función para detectar la tecla ESC y detener el programa
def check_for_escape_key(e):
    global running
    running = False
```

Código 5 Función para detectar "ESC".

```python
p = Productor()
c = Consumidor()
```

Código 6 Instancias del Productor-Consumidor.

```
68
69      keyboard.on_press_key("esc", check_for_escape_key)
70
```

Código 7 Registra la pulsación de "ESC".

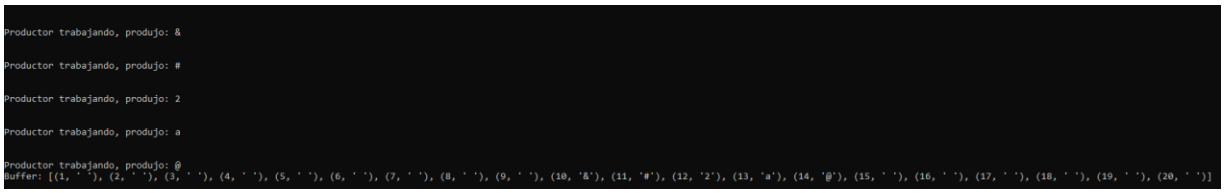
```
70
71      p.start()
72      c.start()
73
```

Código 8 Iniciadores de los hilos.

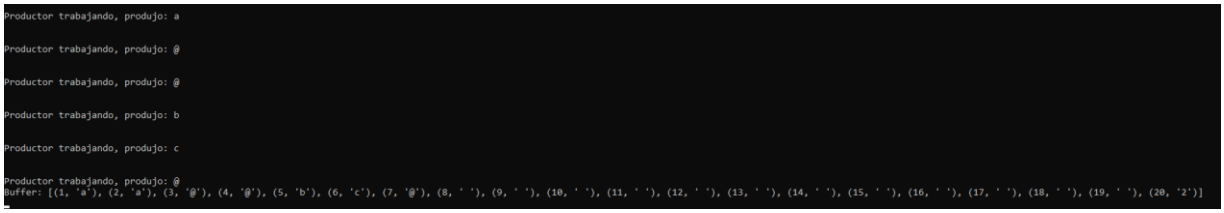
```
74      p.join()
75      c.join()
76
```

Código 9 Finalizar hilos.

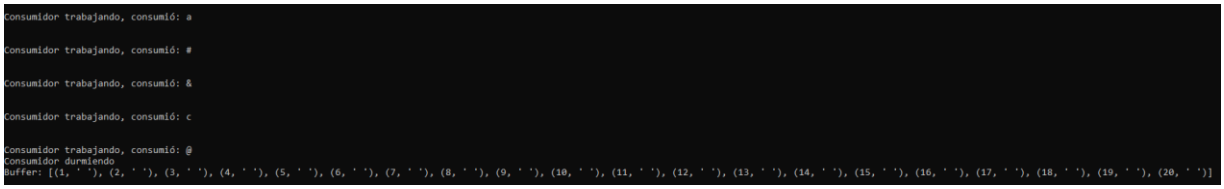
Como se debería ver



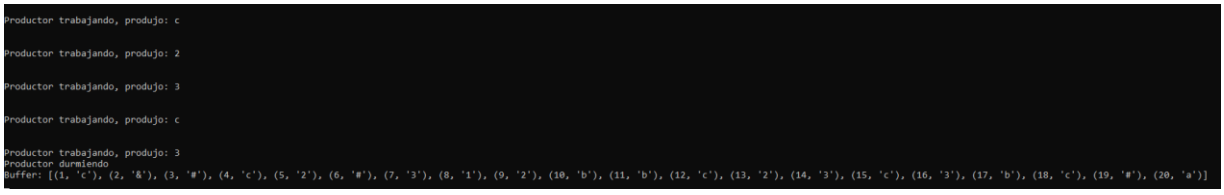
Código 10 Ejemplo Productor trabajando.



Código 11 Ejemplo Productor trabajando 2.



Código 12 Ejemplo Consumidor trabajando y durmiendo.



Código 13 Ejemplo Productor durmiendo.

Enlace de descarga (contenido):

https://drive.google.com/drive/folders/1un7npxXQTQYJ36zkRpTbMaB1zimIyIpR?usp=sharing

Conclusión

Lomeli Jaime

Como conclusión de esta actividad, me gustaría recalcar el apoyo brindado por Luis, ya que aportó trabajo y visión de las peticiones muy claras haciendo ver que fue un trabajo bastante más sencillo de lo que estamos acostumbrados.

Fue también muy interesante la nueva incorporación de hilos y la aplicación directa de exclusión mutua, en este ejemplo ha sido sumamente claro su modo de operación.

Comparando con actividades anteriores, en esta al no manejar demasiados datos, ha sido algo directamente más sencillo de comprender y realizar.

Monjaraz Luis

En conclusión, esta actividad me ayudo a comprender el funcionamiento del Productor-Consumidor, al igual que la implementación de hilos en un código. Debo de decir que esta actividad fue relativamente más sencilla que las anteriores (también puede ser por que se realizó en equipo), eso sí me gustaría decir que fue complicado el entender el uso correcto de los hilos, aunque a comparación del FCFS no fue nada. En resumen, esta actividad es sencilla pero no por ello significa que no sea útil o tenga sus dificultades, y debo de decir que fue una tomada de aire fresco después de trabajar solamente con algoritmos de planificación.