Universidad Nacional Autónoma De México
Facultad de Ingeniería

ASIGNATURA:
Sistemas Operativos

PROFESOR:
Ing. Gunnar Eyal Wolf Iszaevich

GRUPO: 06

FECHA DE ENTREGA: 29/11/2022

ALUMNO:
Méndez Cuenca Alan Eduardo

NÚMERO DE CUENTA:
315110757

"Una situación cotidiana paralelizable"

2023-1

Una situación cotidiana paralelizable

Los eventos en el mundo en que vivimos se producen de forma masivamente paralela, por lo cual, la concurrencia puede presentarse en muchos casos, en ámbitos no necesariamente constreñidos al cómputo.

Por tanto, es de esperarse que –si existieran las primitivas de sincronización en el mundo real– podríamos aplicar lo estudiado en esta unidad a muchas otras situaciones.

Para este proyecto, les pido que:

Identificación y descripción del problema

Describan la situación que modelarán

He decidido trabajar con un restaurante que recibe llamadas para el envío de pedidos, aquí la situación sería un número limitado de teléfonos, al mismo tiempo se deben ir preparando los alimentos y deben ser enviados mientras se van liberando las líneas para recibir otros pedidos.

¿Dónde pueden verse las consecuencias nocivas de la concurrencia? ¿Qué eventos pueden ocurrir que queramos controlar?

Al recibir demasiados pedidos al mismo tiempo, lo cual podría terminar en un colapso en la cocina, repartidores y demás personal.

¿Hay eventos concurrentes para los cuales el ordenamiento relativo no resulta importante?

Debido a las diferentes complejidades en la elaboración de alimentos según su tipo, se harán en el orden de recibido, pero se enviarán una vez realizados.

Documentación

Lenguaje empleado: Python 3.10.2

Bibliotecas: threading (Barrirer,Thread, Semaphore) , time y random.

Desarrollo en: sistema operativo Windows, y probado en la terminal (cmd) de Windows.

Descripción de los mecanismos de sincronización empleados

- Mutex y Semaphore.

Declaraciones de nuestras variables haciendo referencia el uso de mutex y semáforos. Para el control del programa se emplea y que las impresiones en consola fuesen entendibles.

El uso de los mecanismos es para llevar el control de las llamadas, así como saber si entra la llamada para posteriormente entren a la función llamada para que dichos hilos simulen las llamadas entrantes, así mismo entre a la función egreso.

Para la función de pedidos se hace uso para también tener el control de las llamadas consecuencia de la llamada un resultado, que es el alimento deseado

- Barrier.

Se emplea para el uso de una barrera, con el uso de la llamada de wait lo cual nos permite poner la barrera los hilos que se necesita para cumplir dicho requisito, es decir, tener como límite 5 teléfonos en uso durante una hora.

Lógica de operación

Las variables que se ocuparan de uso global o compartidos fueron: mut_pedido, mutLlamada, telefonos, llamadaPedido, mut_atendiendo, debido que se consideraron para uso de las funciones mostradas en el código.

La función llamada es para los clientes que quieren ordenar algún alimento. Se tiene como limite el uso de 5 teléfono, mismos que se atendrán para un pedido en cada línea. Al tener una llamada se tiene la alerta de la entrante, la toma del pedido y el conteo de los teléfonos que se están usando.

La función pedido hace el pedido en la cocina dando algún elemento de manera aleatoria para simular el pedido de un ser humano. En la misma función se manda al repartidor con el pedido y se quita un teléfono de la línea pues ya no se está usando.

Teniendo en cuenta cada una de las funciones que conlleva cada función, por lo tanto, la interacción de cada uno de los hilos está muy relacionadas, ya que uno es la base de donde sale o “llegan las llamadas” como hilos, después se verifica que cada uno de mis hilos “llamadas” son atendidas y al mismo tiempo verificar disponibilidad teniendo en mente la restricción de los 5 teléfonos disponibles y posteriormente ver el resultado de la llamada, es decir, el alimento que requiere el cliente.

Con ayuda de los mutex, fue la organización de ver la disponibilidad de dichos teléfonos, donde se tiene presente en todo momento. Con los semáforos para pretender obtener la fluidez de las llamadas, teniendo principalmente ese objetivo.

proyectoRestaurante.py

```
import random
import time
from threading import Barrier,Thread,Semaphore

#Barrera que detendra 5 hilos, 5 teléfonos
maxLlamadas=Barrier(5)

#Semaforos y mutex que se le asigna a los hilos
mut_atendiendo=Semaphore(1)
mut_pedido=Semaphore(1)
mutLlamada=Semaphore(1)

llamadaPedido=Semaphore(0)
nuevoIngt=Semaphore(0)

#Lista de servicios que se brinda
servicios=["Una Hamburguesa"," Unos Hot-Dogs"," Una Torta"]


def llamada(num):
    global mutLlamada,telefonos,llamadaPedido
    while True:
        llamadaPedido.acquire()
        mutLlamada.acquire()

        print("Llamada entrante")
        time.sleep(3)
        mutLlamada.release()
        mutLlamada.acquire()

        print("Tomando el pedido del cliente ",num)
        telefonos.append(1)
        time.sleep(2)
        mutLlamada.release()
        mutLlamada.acquire()

        print("Los telefonos ocupados\n")
        time.sleep(2)
        mutLlamada.release()
        mutLlamada.acquire()

        time.sleep(2)
        mutLlamada.release()
        llamadaPedido.release()
        maxLlamadas.wait()
```

```
#Envío del pedido
def pedido(num):
    global mut_pedido,mutLlamada,telefonos
    while True:

        mutLlamada.acquire()

        mut_pedido.acquire()
        print("Preparando: ",str(random.choice(servicios)))
        print("Va en camino el repartidor\n")
        telefonos=telefonos[:-1]
        time.sleep(3)
        mut_pedido.release()

        llamadaPedido.release()
        mutLlamada.release()
        mut_pedido.acquire()

        time.sleep(random.randint(1,5))
        mut_pedido.release()
        llamadaPedido.acquire()

def main():
    global llamadaPedido, mut_atendiendo
    while True:
        if len(telefonos) >= 0:
            telDisponible=5-len(telefonos)
        else:
            telDisponible=0

        print("Líneas disponibles: ",telDisponible)
        print("Telefonos ", telefonos, "ocupados ", len(telefonos))

        mut_atendiendo.release()
        nuevoIngt.release()
        llamadaPedido.release()
        print("Esperando llamada\n")

        time.sleep(3)

        for num_llamadas in range(1,5):
            Thread(target=llamada,args=[num_llamadas]).start()

        for num_pedido in range(1,5):
            Thread(target=pedido,args=[num_pedido]).start()

#Lista donde nos inidcara la disponibilidad y el uso de los telefonos
telefonos=[]
Thread(target=main,args=[]).start()
```

Uso en la terminal CMD de Windows:

1. Primero nos ubicamos en la ruta en la cual se encuentre nuestro programa “proyectoRestaurante.py”.

2. Al encontrar nuestro programa solo es necesario escribir su nombre del archivo ya mencionado, así como la extensión, que es py.

3. Y dar enter para su ejecución.

Funcionamiento