Algoritmo de la panadería

Programación concurrente y distribuida

Índice

- ¿Qué es?
- Tipología de problema
- Solución
- Código

¿Qué es?

Es un algoritmo que se llama así porque los clientes son procesos que esperan a ser atendidos. Los clientes (procesos) tienen un número único y son llamados para ser atendidos.

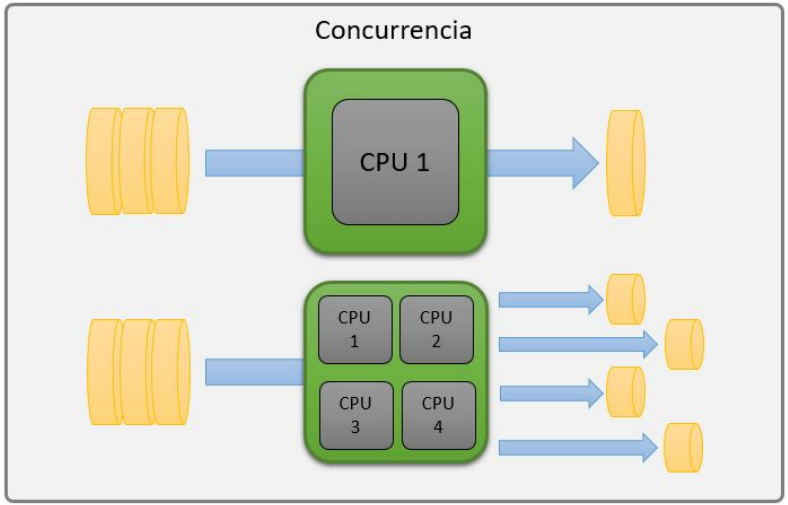
Si un cliente quiere volver a ser atendido tendrá que volver a coger un número y esperar de nuevo a ser atendido.

Tipología del problema

- La panadería solo puede recibir compras de **un cliente a la vez**.
- Se sabe que **n** clientes quieren ingresar a la panadería para comprar y organizarlos para registrar un número de check-in en la recepción en orden. El número de registro aumenta gradualmente en 1.
- Los clientes ingresan a la tienda para comprar productos en orden ascendente de números de registro.
- Los clientes que completaron la compra establecieron su número de inicio de sesión en 0 en la recepción. Si el cliente que completó la compra quiere volver a comprar en la tienda, deberá volver a hacer cola.

Solución propuesta

La solución sería contratar más empleados para que puedan atender más clientes a la vez. De esa manera conseguiremos que el número n de clientes avance más rápido evitando así aglomeraciones.



Código

```
import threading
import time

'''el cliente es un hilo'''


'''tiempo de espera en cada seccion'''
DELAY=1
'''Aqui podemos modificar el numero de hilos de nuestro programa'''
NUM_HILOS = 2
'''calcular el numero de turno'''
eligiendo=[False]*NUM_HILOS
numero= [0]*NUM_HILOS
critical=[False]*NUM_HILOS #se usa para monitorear que threads estan en la seccion critica en un momento dado

'''Cuando un hilo quiere entrar en su sección crítica, primero obtiene su número de turno, que calcula como el máximo de los turnos de los otros hilos, más uno.'''
def maximo():
    max = -1
    for i in numero:
        if (i>max):
            max=i

    return max
```

Código

```
'''hace la comparacion de tuplas'''
'''
a<c : el numero de turno del hilo j es menor que el turno del hilo i
a==c: se puede dar la casualidad de que ambos hilos tengan el mismo turno para eso la siguiente condicion
b<d : si dos o más hilos tienen el mismo número de turno, tiene más prioridad el hilo que tenga el identificador(son unicos) con un número más bajo
'''
def comparacion_tuplas(a,b,c,d):
    if(a<c):
        return True
    elif (a==c and b<d):
        return True
    return False

def bloqueado(i):

    eligiendo[i] = True
    numero[i]= 1 + maximo()
    eligiendo[i]= False
```

Código

```
    for j in range(NUM_HILOS):

        '''se espera a que haya elegido'''
        while eligiendo[j]:
            continue
        while ((numero[j] != 0) and (comparacion_tuplas(numero[j],j,numero[i],i))):
            continue
        '''
        (a, b) < (c, d)

        es equivalente a

        (a < c) o ((a == c) y (b < d))

        mirar comparacion_tuplas

        '''


def desbloqueado(i):
    '''esto dice que el hilo no quiere entrar a la seccion critica (mirar el segundo while de bloqueado donde debe ser != 0 para poder entrar)'''
    numero[i]=0
```

Código

```
def ejecutar_hilo(i):
        while(True):
                '''entra a la seccion critica'''
                bloqueado(i)
                print("Thread "+str(i)+" en seccion critica\n")
                critical[i]=True
                time.sleep(DELAY)
                critical[i]=False
                '''sale de la seccion critica'''
                desbloqueado(i)
                print("Thread "+str(i)+" fuera de seccion critica\n")
                time.sleep(DELAY)



'''****monitorea en que seccion estan los threads********
*******True: en la seccion critica************
*******False: fuera de ella*************'''
def monitorear():
        tiempo=1
        while(True):
                print ("Tiempo: "+str(tiempo))
                '''solo puede haber un True en el vector que se imprime
                porque solo hay un thread en la seccion critica a la vez'''
                print(critical)
                tiempo+=1
                time.sleep(1)
```

Código

```
'''
Asi podemos mostrar el nombre y el identificador del hilo en el que estamos
threading.current_thread().getName()
threading.current_thread().ident'''

def panaderia():
        for num_hilo in range(NUM_HILOS):
                hilo = threading.Thread(name='%s' %num_hilo, target=ejecutar_hilo, args=(num_hilo,))
                hilo.start()



panaderia()
monitorear()
```

¡MUCHAS GRACIAS POR VUESTRA ATENCIÓN!