BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad en Ciencias de la Computación

Programación concurrente y paralela

Reporte de la práctica

Filósofos comensales

Esmeralda Fonseca Sebastian

Para esta práctica se debe realizar un programa en python de filósofos que cumpla con las siguientes indicaciones

- Realizar un programa en python de los filósofos comensales, utilizando rlock de la librería threading.
- Evitar que se de la condición de abrazo mortal (deadlock), para ello hacer que cada filósofo tome un tenedor si es posible.
- Si no está disponible el otro tenedor lo suelte y esta **hambriento**
- Si lo intenta un número de veces (por ejemplo 10), entonces muere por **inanición**
- Si tiene los dos tenedores puede **comer** en un tiempo finito.
- Después de comer pasa a **filosofar**.

Para ello pues con los programas que hemos visto de hilos nos servirá de mucha ayuda para esta práctica

Por lo que tenemos es siguiente fuente código de los filósofos comentariado de las funciones que hace, asi como se definen varias funciones dentro del programa que son las acciones que realizan los filósofos.

Código en python:

```
import time
import random
import threading

N = 5
TIEMPO_TOTAL = 3

class filosofo(threading.Thread):
    semaforo = threading.Lock() #SEMAFORO BINARIO ASEGURA LA EXCLUSION MUTUA
    estado = [] #PARA CONOCER EL ESTADO DE CADA FILOSOFO
    tenedores = [] #ARRAY DE SEMAFOROS PARA SINCRONIZAR ENTRE FILOSOFOS,
MUESTRA QUIEN ESTA EN COLA DEL TENEDOR
    count=0

    def __init__(self):
        super().__init__()      #HERENCIA
        self.id=filosofo.count #DESIGNA EL ID AL FILOSOFO
        filosofo.count+=1 #AGREGA UNO A LA CANT DE FILOSOFOS
        filosofo.estado.append('PENSANDO') #EL FILOSOFO ENTRA A LA MESA EN
ESTADO PENSANDO
        filosofo.tenedores.append(threading.Semaphore(0)) #AGREGA EL
SEMAFORO DE SU TENEDOR( TENEDOR A LA IZQUIERDA)
```

```
        print("FILOSOFO {0} - PENSANDO".format(self.id))

    def __del__(self):
        print("FILOSOFO {0} - Se para de la mesa".format(self.id))
#NECESARIO PARA SABER CUANDO TERMINA EL THREAD

    def pensar(self):
        time.sleep(random.randint(0,5)) #CADA FILOSOFO SE TOMA DISTINTO
TIEMPO PARA PENSAR, ALEATORIO

    def derecha(self,i):
        return (i-1)%N #BUSCAMOS EL INDICE DE LA DERECHA

    def izquierda(self,i):
        return(i+1)%N #BUSCAMOS EL INDICE DE LA IZQUIERDA

    def verificar(self,i):
        if filosofo.estado[i] == 'HAMBRIENTO' and
filosofo.estado[self.izquierda(i)] != 'COMIENDO' and
filosofo.estado[self.derecha(i)] != 'COMIENDO':
            filosofo.estado[i]='COMIENDO'
            filosofo.tenedores[i].release()  #SI SUS VECINOS NO ESTAN
COMIENDO AUMENTA EL SEMAFORO DEL TENEDOR Y CAMBIA SU ESTADO A COMIENDO

    def tomar(self):
        filosofo.semaforo.acquire() #SEÑALA QUE TOMARA LOS TENEDORES
(EXCLUSION MUTUA)
        filosofo.estado[self.id] = 'HAMBRIENTO'
        self.verificar(self.id) #VERIFICA SUS VECINOS, SI NO PUEDE COMER NO
SE BLOQUEARA EN EL SIGUIENTE ACQUIRE
        filosofo.semaforo.release() #SEÑALA QUE YA DEJO DE INTENTAR TOMAR
LOS TENEDORES (CAMBIAR EL ARRAY ESTADO)
        filosofo.tenedores[self.id].acquire() #SOLO SI PODIA TOMARLOS SE
BLOQUEARA CON ESTADO COMIENDO

    def soltar(self):
        filosofo.semaforo.acquire() #SEÑALA QUE SOLTARA LOS TENEDORES
        filosofo.estado[self.id] = 'PENSANDO'
        self.verificar(self.izquierda(self.id))
        self.verificar(self.derecha(self.id))
        filosofo.semaforo.release() #YA TERMINO DE MANIPULAR TENEDORES

    def comer(self):
        print("FILOSOFO {} COMIENDO".format(self.id))
        time.sleep(2) #TIEMPO ARBITRARIO PARA COMER
        print("FILOSOFO {} TERMINO DE COMER".format(self.id))

    def run(self):
        for i in range(TIEMPO_TOTAL):
            self.pensar() #EL FILOSOFO PIENSA
            self.tomar() #AGARRA LOS TENEDORES CORRESPONDIENTES
            self.comer() #COME
```

```
            self.soltar() #SUELTA LOS TENEDORES

def main():
    lista=[]
    for i in range(N):
        lista.append(filosofo()) #AGREGA UN FILOSOFO A LA LISTA

    for f in lista:
        f.start() #ES EQUIVALENTE A RUN()

    for f in lista:
        f.join() #BLOQUEA HASTA QUE TERMINA EL THREAD

if __name__=="__main__":
    main()
```