

DIEGO SÁNCHEZ MENDOZA 340401995

SISTEMAS OPERATIVOS

PRACTICA 1 PARCIAL 3

20/05/2024

CODIGO:

import threading import time

# Crear un sem√°foro con un contador inicial de 2 semaphore = threading.Semaphore(2)

def tarea(id): print(f"Tarea {id} intentando acceder al recurso") with semaphore: print(f"Tarea {id} ha adquirido el sem√°foro") time.sleep(2)

print(f"Tarea {id} ha liberado el sem√°foro")

# Crear m√∫ltiples hilos que ejecuten la funci√≥n "tarea" threads = [] for i in range(5):

thread = threading.Thread(target=tarea, args=(i,)) threads.append(thread) thread.start()

# Esperar a que todos los hilos terminen for thread in threads:

thread.join()

Preguntas de aprendizaje:

a) ¿Que sucede si cambiamos el valor inicial del semaforo?

Llega a ocurrir varios efectos como:

* Mayor valor inicial: Más hilos pueden acceder simultáneamente al recurso, aumentando la concurrencia pero potencialmente introduciendo condiciones de carrera.
* Menor valor inicial: Menos hilos pueden acceder simultáneamente, mejorando la seguridad del recurso pero aumentando el tiempo de espera.

b) ¿Cómo afecta el uso de semaforos al rendimiento del programa?

1. POSITIVOS:

* Controlan la concurrencia y previenen condiciones de carrera.
* Evitan la sobrecarga de recursos compartidos

2. NEGATIVOS:

* Los hilos pueden bloquearse esperando el semáforo, aumentando los tiempos de espera.
* Un acceso demasiado restringido puede llevar a un uso ineficiente de la CPU

c) Ejemplos de casos de usos reales de semaforos en aplicaciones

Se pueden presentar en sistemas operativos, Baes de datos, Servidores Web, aplicaciones de red, Sistenas embedidos:

* Control de acceso a archivos, memoria y dispositivos
* Gestion de transiciones y acceso simultaneas
* Control de conexión de red activas
* Sincronizacion de tarea y acceso a recursos limitados

MODIFICAR EL EJEMPLO PARA USAR UN SEMAFORO BINARIO (CONTADOR INICIAL 1)

CREA UN NUEVO SCRIPT QUE SIMULE UNA APLICACIÓN DE BANCO DONDE VARIOS

CAJEROS (HILOS) ACCEDAN A UNA BASE DE DATOS COMPARTIDA

import threading import time

# Sem√°foro binario para acceder a la base de datos db\_semaphore = threading.Semaphore(1)

# Simulaci√≥n de una base de datos class BancoDB: def \_\_init\_\_(self): self.balance = 1000 # Saldo inicial

def consultar\_balance(self):

print(f"Balance actual: {self.balance}")

def actualizar\_balance(self, monto):

print(f"Actualizando balance con: {monto}") self.balance += monto

print(f"Nuevo balance: {self.balance}")

# Funci√≥n que simula la operaci√≥n de un cajero def cajero(id, banco, monto):

print(f"Cajero {id} intentando acceder a la base de datos") with db\_semaphore:

print(f"Cajero {id} ha adquirido el sem√°foro") banco.consultar\_balance() banco.actualizar\_balance(monto)

print(f"Cajero {id} ha liberado el sem√°foro") time.sleep(1) # Simulaci√≥n del tiempo de operaci√≥n del cajero

# Crear instancia de la base de datos

banco = BancoDB()

# Crear m√∫ltiples hilos que simulan los cajeros cajeros = []

montos = [100, -50, 200, -30, 150] # Monto de transacciones de cada cajero for i in range(5): thread = threading.Thread(target=cajero, args=(i, banco, montos[i])) cajeros.append(thread) thread.start()

# Esperar a que todos los hilos terminen for cajero in cajeros:

cajero.join()