Universidad Nacional Autónoma de México

Facultad de ingeniería

Sistemas Operativos

Presentación

Problema de sincronización: Building H_2O

Integrantes del equipo:

López Sugahara Ernesto Danjiro

Ramírez Martínez Luis Ángel

Problema de sincronización: Construyendo H_2O

Introducción general

Los problemas de sincronización están enfocados a buscar una solución a problemas relacionados al comportamiento aleatorio de hilos o procesos que deben de seguir un orden específico de eventos para funcionar de forma adecuada. Es decir, poder tener un mayor control sobre la aleatoriedad generada por el sistema operativo relacionado a hilos y procesos para que de esta forma sea posible modelar el comportamiento de eventos que requieren la sincronización de estos.

Una gran herramienta para solventar y enfrentar este tipo de problemas de sincronización son los semáforos. Un semáforo es un señalizador que funciona como mensajero entre hilos; este puede señalizar si hay paso o no (ya sea bloqueando un hilo o dándole acceso).

Para la solución del problema se utilizó el lenguaje de programación Python, donde los semáforos pueden ser visualizados como valores enteros con las siguientes características:

- Tienen un valor asignado que representa cuantos hilos pueden pasar. Este no puede ser un número menor a 0.

- Cuando el semáforo está en cero, el hilo que intenta pasa será bloqueado hasta que reciba una señal que le permita seguir adelante. Siempre que un hilo adquiera un semáforo, el contador disminuye

- Cuando un hilo incrementa el semáforo, manda una señal para dejar pasar otros hilos (ya sea que estuviesen bloqueados o vienen en un orden secuencial, funcionando como un mutex)

Planteamiento del problema

El problema fue planteado en una clase de Sistemas Operativos en la universidad de Berkeley y enuncia lo siguiente:

“Hay dos tipos de hilos, oxígeno e hidrogeno. Para poder mezclar estos hilos en moléculas de agua, se debe tener una barrera que bloquee a cada hilo hasta que una molécula completa esté lista para mezclarse.

Cada que un hilo pase la barrera, deberá de invocar la función “bond” (vincular). Se tiene que garantizar que los tres hilos de la molécula invoquen la función antes de que cualquier hilo de la siguiente molécula lo haga. En otras palabras:

- Si un hilo de oxígeno llega a la barrera cuando no hay hilos de hidrogeno presentes, deberá de esperar para a que dos hilos de hidrogeno se presenten.

- Si un hilo de hidrógeno llega a la barrera cuando no hay otros hilos presentes, deberá de esperar por un hilo de oxígeno y un hilo de hidrogeno.

No nos debemos de preocupar sobre vincular los hilos de forma explícita; es decir, los hilos no necesariamente saben con qué otros hilos están siendo vinculados. La clave es que los hilos pasen la barrera en un conjunto completo; de este modo, si examinamos la secuencia de hilos que invocaron la función "bond" y los dividimos en grupos de tres, cada grupo deberá de contener un oxígeno y dos hilos de hidrogeno"

Solución

A continuación, se muestra la solución planteada con el lenguaje de programación Python, la cual se irá desmenuzando y se mostrará la salida respectiva.

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

mutex = Semaphore(1)
mutex_bonding = Semaphore(1)
oxigeno = []
hidrogeno = []
water_bonding = ""
contador_barrera = 0
barrera = Semaphore(0)
cola_oxigeno = Semaphore(0)
cola_hidrogeno = Semaphore(0)

def bond(element):
    # Se entra a la zona de vinculación
    global water_bonding
    global contador_barrera
    mutex_bonding.acquire()
    contador_barrera += 1
    water_bonding += element
    if (contador_barrera == 3): # Es necesario volver a bloquear la barrera
        barrera.acquire()
        contador_barrera = 0
    mutex_bonding.release()
    return

def recibirOxigeno():
    # Llega un hilo de oxígeno
```

```
    global hidrogeno, oxigeno
    # Entra a la zona crítica tomando el Mutex
    mutex.acquire()
    oxigeno.append("O")
    print(oxigeno,hidrogeno)
    # Se analizan los posibles casos
    if len(hidrogeno) >= 2:
        # Como hay dos hidrogenos esperando, es posible realizar una vinculación
        # Primero se le da la señal a dos hilos de hidrogeno para que sigan su
camino hacia la barrera
        cola_hidrogeno.release(2)
        hidrogeno.pop()
        hidrogeno.pop()
        # Se da la señal a un hilo de oxígeno par que siga su camino hacia la
barrera
        cola_oxigeno.release()
        oxigeno.pop()
        # Se libera la barrera para los 3 hilos
        barrera.release()
        print("Listos para mezclar!")
    else:
        # En otro caso se libera el mutex y el oxígeno espera la señal de los
hidrogenos
        mutex.release()

    # Este es el punto de espera del oxigeno
    cola_oxigeno.acquire()

    # Cuando se libera la barrera, el oxigeno pasará a la zona de vinculación
    barrera.acquire()
    barrera.release()
    # Se vincula un hilo de oxigeno
    bond("O")
    # Libera el mutex (esto es útil cuando el oxigeno entra a la zona de
vinculación)
    mutex.release()


def recibirHidrogeno():
    # Llega un hilo de hidrogeno
    global hidrogeno, oxigeno
    # Entra a la zona crítica
    mutex.acquire()
    hidrogeno.append("H")
    print(oxigeno,hidrogeno)
    # Se analizan los psoibles casos
    if len(hidrogeno) >= 2 and len(oxigeno) >= 1:
```

```
        # Es posible realizar la vinculación, por lo que se sigue el mismo
proceso que el oxigeno
        cola_hidrogeno.release(2)
        hidrogeno.pop()
        hidrogeno.pop()
        cola_oxigeno.release()
        oxigeno.pop()
        barrera.release()
        print("Listos para mezclar!")
    else:
        # no es posible realizar la vinculación, por lo que el hidrogeno tiene
que esperar
        mutex.release()

    cola_hidrogeno.acquire()
    barrera.acquire()
    barrera.release()
    bond("H")



# Entrada:
entrada = "OOHHHHHOHHHHHHOOO"
print(f"Mezclando {entrada} en moléculas de agua:")

# Salida esperada: 6 moléculas de agua

# Se manda un hilo del tipo respectivo a la cadena de entrada:
for elemento in entrada:
    if elemento == "O":
        Thread(target=recibirOxigeno, args=[]).start()
    else:
        Thread(target=recibirHidrogeno, args=[]).start()

sleep(0.4)
print(f"Vínculos obtenidos: {len(water_bonding) / 3}:")
# Dando formato a la salida
for i,c in enumerate(water_bonding):
    print(c, end ="")
    if (i + 1) % 3 == 0 and i != len(water_bonding) - 1:
        print(" - ", end="")

print()
```

A. continuación se analiza el código anterior:

Variables utilizadas:

```
mutex = Semaphore(1)
mutex_bonding = Semaphore(1)
oxigeno = []
hidrogeno = []
water_bonding = ""
contador_barrera = 0
barrera = Semaphore(0)
cola_oxigeno = Semaphore(0)
cola_hidrogeno = Semaphore(0)
```

Para la solución de este problema se tienen las variables siguientes:

- mutex: Será de utilidad para la zona crítica del oxígeno y el hidrogeno. Este permite que únicamente el hilo activo pueda realizar la modificación al respectivo arreglo de oxígeno o hidrogeno, así como las comparaciones para determinar qué se tiene que realizar a continuación en el código.

- oxigeno: Es una lista que permitirá ir visualizando la cola de espera de los hilos relacionados al átomo de oxígeno.

- hidrogeno: Es el mismo concepto que la lista de oxígeno.

- water_bonding: Esta nos permite realizar la concatenación o vinculación de los tres hilos para obtener una molécula de agua.

- barrera: Es la que permitirá realizar la mezcla de los tres hilos.

- cola_oxigeno: Permite llevar el control de los hilos de oxígeno.

- Cola_hidrogeno: Permite llevar el control de los hilos de hidrogeno.

- mutex_bonding: Protege la zona crítica dentro de la función de vinculación.

Con base en lo anterior, se analizará el paso a paso del código con las diferentes secciones de este.
Se observa que la entrada está dada por una cadena de átomos que simularán los hilos correspondientes para realizar las vinculaciones. En este caso la entrada está dada por: “OOHHHHHOHHHHHHOOO”. Por lo tanto, se seguiría el siguiente flujo:

1. Entra un hilo de oxígeno, por lo que el hilo está en la función recibirOxígeno:

```
def recibirOxigeno():
    # Llega un hilo de oxígeno
    global hidrogeno, oxigeno
    # Entra a la zona crítica tomando el Mutex
    mutex.acquire()
    oxigeno.append("O")
    print(oxigeno,hidrogeno)
    # Se analizan los posibles casos
    if len(hidrogeno) >= 2:
        # Como hay dos hidrogenos esperando, es posible realizar una vinculación
        # Primero se le da la señal a dos hilos de hidrogeno para que sigan su
camino hacia la barrera
        cola_hidrogeno.release(2)
        hidrogeno.pop()
        hidrogeno.pop()
        # Se da la señal a un hilo de oxígeno par que siga su camino hacia la
barrera
        cola_oxigeno.release()
        oxigeno.pop()
        # Se libera la barrera para los 3 hilos
        barrera.release()
        print("Listos para mezclar!")
    else:
        # En otro caso se libera el mutex y el oxígeno espera la señal de los
hidrogenos
        mutex.release()

    # Este es el punto de espera del oxigeno
    cola_oxigeno.acquire()

    # Cuando se libera la barrera, el oxigeno pasará a la zona de vinculación
    barrera.acquire()
    barrera.release()
    # Se vincula un hilo de oxigeno
    bond("O")
    # Libera el mutex (esto es útil cuando el oxigeno entra a la zona de vinculación)
    mutex.release()
```

En esta, lo primero que realiza el hilo es obtener el mutex de zona crítica. A través de esto, se agrega un oxígeno a la lista. Posteriormente se revisan dos condiciones:

1. Hay dos hidrógenos disponibles para realizar la vinculación y liberar la barrera.

2. No hay átomos suficientes, por lo que el oxígeno libera el mutex y espera en la barrera para poder vincularse posteriormente.

Posteriormente llega otro oxígeno que, bajo la misma situación, queda en espera debido a la falta de hidrogeno para realizar la vinculación.

Luego entra el primer hidrogeno, el cual sigue el siguiente flujo de código:

```
def recibirHidrogeno():
    # Llega un hilo de hidrogeno
    global hidrogeno, oxigeno
    # Entra a la zona crítica
    mutex.acquire()
    hidrogeno.append("H")
    print(oxigeno,hidrogeno)
    # Se analizan los psoibles casos
    if len(hidrogeno) >= 2 and len(oxigeno) >= 1:
        # Es posible realizar la vinculación, por lo que se sigue el mismo proceso
que el oxigeno
        cola_hidrogeno.release(2)
        hidrogeno.pop()
        hidrogeno.pop()
        cola_oxigeno.release()
        oxigeno.pop()
        barrera.release()
        print("Listos para mezclar!")
    else:
        # no es posible realizar la vinculación, por lo que el hidrogeno tiene que
esperar
        mutex.release()

    cola_hidrogeno.acquire()
    barrera.acquire()
    barrera.release()
    bond("H")
```

En este caso, el comportamiento es prácticamente el mismo que el oxígeno. Con la llegada del primer hidrogeno sucede lo mismo que los oxígenos, aún no hay suficientes elementos para realizar una vinculación. En este punto es importante realizar un análisis de las variables que se tienen:

$$oxigeno = ['O', 'O']$$
$$hidrogeno = ['H']$$
$$cola_{oxigeno} = Hay\ dos\ hilos\ esperando$$
$$cola_{hidrogeno} = Hay\ un\ hilo\ esperando$$
$$barrera = bloqueada\ a\ la\ espera\ de\ otro\ átomo\ de\ H$$

Con esto, se espera que el siguiente átomo de hidrogeno libere la barrera para realizar la vinculación de los tres hilos respectivos. La entrada del próximo hilo de hidrogeno genera lo siguiente:

1. Se cumple la condición relacionada a la posibilidad de realizar una vinculación, ya que se cuenta con 2 hilos de hidrogeno y 1 de oxígeno.
2. Como la condición cumple, el hilo libera dos hilos de hidrogeno y no de oxígeno, los cuales se encontraban esperando antes de la barrera:

```
    cola_hidrogeno.acquire()
    barrera.acquire()
    barrera.release()
    bond("H")
```

3. Con esto, se realiza una liberación de la barrera para los tres hilos, los cuales llegarán a la función “bond” que únicamente concatena los tres elementos para obtener la molécula de agua.

```
def bond(element):
    # Se entra a la zona de vinculación
    global water_bonding
    global contador_barrera
    mutex_bonding.acquire()
    contador_barrera += 1
    water_bonding += element
    if (contador_barrera == 3): # Es necesario volver a bloquear la barrera
        barrera.acquire()
        contador_barrera = 0
    mutex_bonding.release()
    return
```

En este, se lleva el conteo hasta 3, con lo cual se visualizar que únicamente hayan pasado 3 hilos para realizar la mezcla para posterioremente volver a bloquear la barrera y asegurar que únicamente 3 hilos tengan acceso a la funciión bond. Con esto, si se ejecuta el código se observa la siguiente salida:

```
Mezclando OOHHHHHOHHHHHHHOOO en moléculas de agua:
['O'] []
['O', 'O'] []
['O', 'O'] ['H']
['O', 'O'] ['H', 'H']
Listos para mezclar!
['O'] ['H']
['O'] ['H', 'H']
Listos para mezclar!
[] ['H']
['O'] ['H']
['O'] ['H', 'H']
Listos para mezclar!
[] ['H']
[] ['H', 'H']
['O'] ['H', 'H']
Listos para mezclar!
['O'] []
['O'] ['H']
['O'] ['H', 'H']
Listos para mezclar!
[] ['H']
['O'] ['H']
['O'] ['H', 'H']
Listos para mezclar!
Vínculos obtenidos: 6.0:
OHH - HHO - HOH - HHO - HHO - HOH
```

Referencias bibliográficas

[1] Downey, A, B. (2016). *The little book of semaphores.* Green Tea Press.

[2] Python (2023). *Threading — thread-based parallelism.* Recuperado de: https://www.scribbr.es/citar/generador/folders/5o5YaVQJUH9EtlZcvOKfr0/lists/1HM6LPBO8htkZimlwzUlFg/