Bilgisayar İşletim Sistemleri Uygulama VII

Örnek Problem Çözümleri

Örnek Problemler

- Su (H_2O) oluşumu problemi
- Kola Makinesi Problemi

Su (H_2O) Oluşumu Problemi

- Su molekülü oluşumunu modellemek istersek iki iplik (thread) kullanabiliriz:
  - Hidrojen (H) (2 adet)
  - Oksijen (O)
- Molekül oluşumu için eşik değeri (enerji, sıcaklık, vs) belirlenir. Eşik değerini aşan iplik bag() fonksiyonunu çağırır.
- Bir ipliğin, (i+1). molekül için bag() fonksiyonunu çağırması, ancak ve ancak i. molekülü oluşturan için üç ipliğin de bag() fonksionunu çalıştırıp tamamlamış olması gerekir.
  - Bir O ipliği eşiğe ulaşınca
    - Hiçbir H ipliği yoksa, iki H ipliğini bekler
  - Bir H ipliği eşiğe ulaşınca
    - Eşikte kendisinden başka hiçbir iplik yoksa, bir H ipliği ve bir O ipliğini beklemek zorundadır

Su (H_2O) Oluşumu Problemi

- Soru:
  - İplikler eşiği üçlü gruplar halinde geçmektedirler. Bir grup 2 H, 1 O ipliğinden oluşmaktadır.
  - Sözkonusu kısıtları sağlayan senkronizasyon adımlarını H ve O iplikleri için oluşturunuz.

Su (H_2O) Oluşumu Problemi

- Çözüm
  - Kullanılması gereken temel bileşenler ve ilk değerleri
    - mutex = Semafor (1)
      - (Mut$_{ual}$ Ex$_{clusion}$)
    - Oksijen = 0
      - sayaç
    - Hidrojen = 0
      - sayaç
    - Esik = Esik (3)
      - bag() fonksiyonunu çağıran üç iplik de eşiğe vardığı zaman bir sonraki 3'lü grubun çalışmaya başlamasına izin verilir
    - oxyQueue = Semafor(0)
      - Oksijen ipliklerinin beklediği semafor
    - hydroQueue = Semafor(0)
      - Hidrojen ipliklerinin beklediği semafor

Su (H_2O) Oluşumu Problemi

- Kullanılacak Fonksiyonlar
  - oxyQueue.wait()
    - Oksijen kuyruğuna katıl
  - oxyQueue.signal()
    - Oksijen kuyruğundan bir oksijen ipliğini serbest bırak
  - Hidrojen iplikleri için tanımlı fonksiyonlar ise hydroQueue.signal() ve hydroQueue.wait() fonksiyonlarıdır

Su (H_2O) Oluşumu Problemi

- Başlangıçta *hydroQueue* ve o*xyQueue* semaforları kilitlenmiş durumda.
  - hydroQueue = 0
  - oxyQueue = 0
- Eşiğe ulaşan bir O ipliği, 2 H ipliğinin işleme girmesine izin vermelidir:
    - hydroQueue semaforunun değerini 2 arttırmalıdır.
    - Arttırma işleminden sonra H ipliklerinin eşiğe varmasını beklemelidir

Su (H_2O) Oluşumu Problemi

- Oksijen ipliği için Pseudocode

```
mutex.wait()
oksijen++
if ( hidrojen >= 2)
   hydroQueue.signal(2)
   hidrojen -= 2
   oxyQueue.signal()
   oksijen -=1
else
   mutex.signal()

oxyQueue.wait()
bag()
esik.wait()
mutex.signal()
```

Su (H_2O) Oluşumu Problemi

- Hidrojen ipliği için Pseudocode

```
mutex.wait()
hidrojen++
if ( hidrojen >= 2 and oksijen >=1)
   hydroQueue.signal(2)
   hidrojen -= 2
   oxyQueue.signal()
   oksijen -=1
else
   mutex.signal()

hydroQueue.wait()
bag()
esik.wait()
```

Kola Makinesi Problemi

- Problem
  - Kapasite = 10 kola
  - Durum = Yarı Dolu
  - 4 iplik
    - 2 üretici
      - produce() fonksiyonunu çağırıyor
    - 2 tüketici
      - consume() fonksiyonunu çağırıyor
    - Sonsuz döngü içinde çalışacaklar ve her döngüde μ değeri ile üssel dağılıma uyan bir ortalama süre uyuyacaklar.
    - Uzun ölçekli davranış
      - Saniyede 2 kola eklenecek, 2 kola tüketilecek
    - Kısa ölçekli davranış
      - Zaman zaman 10'dan fazla kola bulunduğu gibi, 0'ın altında bulunduğu da gözlenebilir
      - Pratikte olmaması gerekiyor.

Kola Makinesi Problemi

- İstenenler
  - Kola erişimini karşılıklı dışlamaya göre düzenle
  - Eğer makinedeki kola sayısı 0 (sıfır) ise tüketici kendini kilitlemeli ve bir kola eklenene kadar beklemeli
  - Eğer makinedeki kola sayısı 10 ise, üretici, en az bir kola tüketilene kadar yeni kola ilave etmemeli

Kola Makinesi Problemi

- **Asenkron durum için pseudocode**

```
import random
class shared:
    def __init__(self,start = 5):
        self.cokes = start
    def consume(shared):
        shared.cokes -=1
        print shared.cokes

    def produce(shared)
        shared.cokes +=1
        print shared.cokes

    def loop(shared, f, mu = 1):
      while True:
      t = random.expovariate(1/mu)
      time.sleep(t)
      f(shared)

    shared = Shared()
    fs = [consume]*2 + [produce]*2
    threads = [Thread (loop, shared, f) for f in fs]
    for thread in threads: thread.join()
```

Kola Makinesi Problemi

- Çözüm:
  - Paylaşılan değişkenler

```
Def __init__(self, start = 5, kapasite = 10)
    self.cokes = semafor(start)
    self.slots = semafor (kapasite - start)
    self.mutex = semafor(1)
```

Kola Makinesi Problemi

- Çözüm:
  - Üretim ve tüketim fonksiyonları düzenlenmeli

```
def consume(shared):
    shared.cokes.wait()
    shared.mutex.wait()
    print shared.cokes.value()
    shared.mutex.signal()
    shared.slots.signal()

def produce(shared)
    shared.slots.wait()
    shared.mutex.wait()
    print shared.cokes._Semaphore__value
    shared.mutex.signal()
    shared.cokes.signal()
```