İşletim Sistemleri Uygulama 7

Örnek Senkronizasyon Problemleri

Bilgisayar Mühendisliği

İstanbul Teknik Üniversitesi
34469 Maslak, İstanbul

March 30, 2011

Bugün

İşletim Sistemleri Uygulama 7

- Uygulamada adım adım semafor kullanımı
- Örnek Senkronizasyon Problemleri - H_2O
- Örnek Senkronizasyon Problemleri - Sushi Bar
- Örnek Senkronizasyon Problemleri - Parti Problemi

Problem

Amaç, bir yazıcı denetim uygulaması yazmaktır. Sistemde sadece bir yazıcı vardır. Bu yazıcı yazdığı her sayfaya karşılık ekrana bir karakter basacak biçimde benzetilmiştir. Bu uygulamaya kısa zaman içinde iki ayrı belgeden yüzer tane basma görevi verilmiştir. Her iki tür belgenin sayfaları birbirine karışmadan baskı işi kotarılmalıdır.

Senkronizasyon olmadan

```
1   // int s; // currently not used

    void* printThis(void* typ){
            int i,j;
            char* str=(char)typ=='a'?"abcdefghij":"0123456789";
6           // Create two types of content for print jobs
            for(i=0;i<100;i++){ // 100 separate print jobs
                    for(j=0;j<10;j++){ // of 10 pages each.
                            printf("%c", str[j]);
                            // Each character represents a page
11                  }
            }
            pthread_exit(NULL);
    }

16  int main(void){
            pthread_t a,n;
            setvbuf(stdout, (char*)NULL, _IONBF, 0); // no-buffer printf

            // s=1; // currently not used
21
            printf("I'm the NO-SYNC printer manager.\n");

            // Run two threads on printThis function with separate params
            pthread_create(&a, NULL, printThis, (void *)'a');
26          pthread_create(&n, NULL, printThis, (void *)'n');

            // Wait for the threads to finish
            pthread_join(a, NULL);
            pthread_join(n, NULL);
31
            pthread_exit(NULL);
            return 0;
    }
```

Senkronizasyon olmadan oluşan örnek çıktı

```
    I'm the NO-SYNC printer manager.
    Oh! Two hundred jobs are submitted, and I got only one printer!
 3
    abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
    abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabc0123456
    7890123456789012345678901234567890123456789012345678901234567890123456789012345
    7890123456789012345678901234567890123456789012345678901234567890123456789012345
 8  7890123456789012345678901234567890123456789012345678901234567890123456defghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
13  efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
18  efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcd
    efghijabcdefghij67890123456789012345678901234567890123456789012345678901234567890
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
23  1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
28  1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789012345678901234567890123456789
    1234567890123456789012345678901234567890123456789
```

Senkronizasyon denemesi

```
1   int s;

    void* printThis(void* typ){
            int i,j;
            char* str=(char)typ=='a'?"abcdefghij":"0123456789";
6           for(i=0;i<100;i++){
                    if(s>0){ // Checking for the value of s then,
                            s--; // modifying it to lock other threads
                            for(j=0;j<10;j++){
                                    printf("%c", str[j]);
11                          }
                            s++; // Unlocking other threads.
                    }
            }
            pthread_exit(NULL);
16  }

    int main(void){
            pthread_t a,n;
            setvbuf(stdout, (char*)NULL, _IONBF, 0); // no-buffer printf
21
            s=1;

            printf("I'm the DUMMY-SYNC printer manager.\n");

26          pthread_create(&a, NULL, printThis, (void *)'a');
            pthread_create(&n, NULL, printThis, (void *)'n');

            pthread_join(a, NULL);
            pthread_join(n, NULL);
31
            pthread_exit(NULL);
            return 0;
    }
```

Senkronizasyon denemesinde oluşan örnek çıktı

```
ubuntu@ubuntu:~/Desktop$ ./a.out
I'm the DUMMY-SYNC printer manager.
When a print job is submitted,
I create a new thread to print the string.
Oh! Two hundred jobs are submitted,
and I got only one printer!

abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghij
abcdefghijabcdefghijubuntu@ubuntu:~/Desktop$
```

Başarılı senkronizasyon

```
1   void sem_signal(int semid, int val){
        struct sembuf semafor;
        semafor.sem_num=0;
        semafor.sem_op=val;
        semafor.sem_flg=1;
6       semop(semid, &semafor,1);
    }

    void sem_wait(int semid, int val){
        struct sembuf semafor;
11      semafor.sem_num=0;
        semafor.sem_op=(-1*val);
        semafor.sem_flg=1;
        semop(semid, &semafor,1);
    }
16
    void createSemaphore(char *argv[]){
        int someKey = ftok(strcat(get_current_dir_name(),argv[0]),1);
        s = semget(someKey, 1, 0700|IPC_CREAT);
        semctl(s,0,SETVAL,1);
21  }
```

Başarılı senkronizasyonla oluşan örnek çıktı

```
int s;

void* printThis(void* typ){
        int i,j;
        char* str=(char)typ=='a'?"abcdefghij":"0123456789";
        for(i=0;i<100;i++){
                sem_wait(s,1); // decrease
                for(j=0;j<10;j++){
                        printf("%c", str[j]);
                }
                sem_signal(s,1); // increase
        }
        pthread_exit(NULL);
}

int main(int argc, char *argv[]){
        pthread_t a,n;
        setvbuf(stdout, (char*)NULL, _IONBF, 0); // no-buffer printf

        createSemaphore(argv);

        printf("I'm the SEM-SYNC printer manager.\n");
        pthread_create(&a, NULL, printThis, (void *)'a');
        pthread_create(&n, NULL, printThis, (void *)'n');
        pthread_join(a, NULL);
        pthread_join(n, NULL);

        semctl(s,0,IPC_RMID,0); // Delete s

        pthread_exit(NULL);
        return 0;
}
```

Başarılı senkronizasyon

```
I'm the SEM-SYNC printer manager.
Oh! Two hundred jobs are submitted, and I got only one printer!
3
abcdefghijabcdefghijabcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
8  abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
13 0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
18 abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
23 0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
28 abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij
0123456789abcdefghij0123456789abcdefghij0123456789abcdefghij0123456789
abcdefghij01234567890123456789012345678 9ubuntu@ubuntu:~/Desktop$
```

Su(H_2O) oluşumu problemi

- Su molekülünün oluşumunu iki tip iplik(thread) kullanarak modellemek istiyoruz
- 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
- 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.

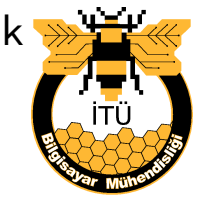
Su(H_2O) oluşumu problemi

Kullanılması gereken temel bileşenler ve ilk değerleri

- `mutex = Semafor(1)` Mutex: Karşılıklı dışlama(MUTual EXclusion)
- `oksijen = 0` Mutex korumalı sayaç
- `hidrojen = 0` Mutex korumalı sayaç
- `barrier = Barrier(3)` bag() fonksiyonunu çağıran üç iplik de eşiğe vardığı zaman bir sonraki 3lü grubun çalışmaya başlamasına izin verilir
- `oxyQueue = Semafor(0)` Oksijen ipliklerinin beklediği semafor
- `hydroQueue = Semafor(0)` Hidrojen ipliklerinin beklediği semafor

Kuyruk yapılarına bağlı temel fonksiyonlar ise aşağıdaki gibi ismlendirilmiştir.

- `oxyQueue.wait()` : Oksijen kuyruğuna katıl
- `hydroQueue.wait()` : Hidrojen kuyruğuna katıl
- `oxyQueue.signal()` : Oksijen kuyruğundan bir oksijen ipliğini serbest bırak
- `hydroQueue.signal()` : Hidrojen kuyruğundan bir hidrojen ipliğini serbest bırak



Su(H_2O) oluşumu problemi

- Başlangıçta `hydroQueue` ve `oxyQueue` 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 için özet kod:

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

    oxyQueue.wait()
    bag()
13  barrier.wait()
    mutex.signal()
```

Su(H_2O) oluşumu problemi

Hidrojen için özet kod:

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

11  hydroQueue.wait()
    bag()
    barrier.wait()
```

Sushi bar problemi

Imagine a sushi bar with 5 seats. If you arrive while there is an empty seat, you can take a seat immediately. But if you arrive when all 5 seats are full, that means that all of them are dining together, and you will have to wait for the entire party to leave before you sit down.

Çözümde kullanılacak değişkenler

```
    eating = waiting = 0 // keep track of the number of threads
2   mutex = Semaphore(1) // mutex protects both counters
    block = Semaphore(0) // incoming customers' queue(regular meaning)
    must_wait = False // indicates that the bar is full
```

Çözüm denemesi

```
1   mutex.wait()
    if must_wait:
            waiting += 1
            mutex.signal()
            block.wait()
6
            mutex.wait()      // reacquire mutex
            waiting -= 1

    eating += 1
11  must_wait = (eating == 5)
    mutex.signal()

    // eat sushi

16  mutex.wait()
    eating -= 1
    if eating == 0:
            n = min(5, waiting)
            block.signal(n)
21          must_wait = False
    mutex.signal()
```

Bir çözüm

```
    mutex.wait()
    if must_wait:
3           waiting += 1
            mutex.signal()
            block.wait()
    else:
            eating += 1
8           must_wait = (eating == 5)
            mutex.signal()

    // eat sushi

13  mutex.wait()
    eating -= 1
    if eating == 0:
            n = min(5, waiting)
            waiting -= n
18          eating += n
            must_wait = (eating == 5)
            block.signal(n)
    mutex.signal()
```

Bir başka çözüm

```
    mutex.wait()
    if must_wait:
            waiting += 1
4           mutex.signal()
            block.wait()    // when we resume, we have the mutex
            waiting -= 1

    eating += 1
9   must_wait = (eating == 5)
    if waiting and not must_wait:
            block.signal()     // and pass the mutex
    else:
            mutex.signal()
14
    // eat sushi

    mutex.wait()
    eating -= 1
19  if eating == 0: must_wait = False

    if waiting and not must_wait:
            block.signal()     // and pass the mutex
    else:
            mutex.signal()
```

Parti Problemi

Yurtta yapılacak bir parti için aşağıdaki kısıtlar geçerlidir:

- Bir odada aynı anda birçok öğrenci bulunabilir
- Yurt müdiresi bir odaya iki durumda dalabilir
  - Odada öğrenci yoksa, odayı aramak için
  - Odada 50den fazla öğrenci varsa partiyi dağıtmak için
- Müdire odadayken başka öğrenci odaya giremez ancak öğrenciler odadan ayrılabilir
- Müdire odayı ancak oda boşalınca terk eder.
- Sadece bir müdire teftişten sorumludur.

Bütün bu kısıtları karşılayan bir simülsayon yazınız.

Parti Problemi

Kullanılması gereken temel bileşenler ve ilk değerleri

- `ogrenci = 0` Öğrenci sayısı
- `mudire = ‘‘odada değil’’` Müdire durumunu tutan bir `enum`
- `mutex = Semafor(1)` Öğrenci ve müdire durumlarını korumak için
- `turnike = Semaphore(1)` Dekan odadayken öğrencileri dışarıda tutmak için
- `odaHazir = Semaphore(0)` Müdire dışarıda ve oda boşsa
- `ogrencilerCikti = Semaphore(0)` Müdire içeride ve oda boşaldıysa

Parti Problemi

Mudire için özet kod:

```
1   mutex.wait()
            if ogrenci > 0 and ogrenci < 50:
                    mudire = bekliyor
                    mutex.signal()
                    odaHazir.wait()     # and get mutex from the student.
6
            # ogrenci say s  0 ya da >= 50 olmal

            if ogrenci >= 50:
                    mudire = odada
11                  dagit()
                    turnike.wait()  # girisi kilitle
                    mutex.signal()
                    ogrencilerCikti.wait()
                    turnike.signal()          # giris kilidini ac
16
            else:   # ogrenci sayisi 0 olmali
                    ara()

21  mudire = odada degil
    mutex.signal()
```

Parti Problemi

Öğrenci için özet kod:

```
mutex.wait()
        if mudire == odada:
                mutex.signal()
                turnike.wait()
                turnike.signal()
                mutex.wait()
        ogrenci += 1

        if ogrenci == 50 and mudire == bekliyor:
                odaHazir.signal()                    # mutexi mudireye aktar
        else:
                mutex.signal()
party()

mutex.wait()
        ogrenci -= 1

        if ogrenci == 0 and mudire == bekliyor:
                odaHazir.signal()                    # odaBos u mudireye aktar
        elif ogrenci == 0 and mudire == odada:
                # ogrencilerCikti yi mudireye aktar
                ogrencilerCikti.signal()
        else:
                mutex.signal()
```