

## Prosesler Arası Etkileşim

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan Doç. Dr. A. Şima Etaner-Uyar

## Giriş

- eş zamanlı prosesler arasındaki etkileşim nedenleri:
  - kaynak paylaşımı
  - karşılıklı haberleşme
  - senkronizasyon

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Etaner-Uyar

## Giriş

- prosesler arası etkileşim üç düzeyde
  - prosesler birbirinden habersiz (rekabet)
    - sistem kaynakları kullanımı (işletim sistemi düzenler)
  - prosesler dolaylı olarak birbirinde haberdar (paylaşma yoluyla işbirliği)
    - kaynak paylaşımı (karşılıklı dışlama ve senkronizasyon)
  - prosesler doğrudan birbirinden haberdar (haberleşme yoluyla etkileşim)

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Etaner-Uyar

## Kaynak Paylaşımı

- 2 temel durum
  - karşılıklı dışlama
    - 2 tür paylaşılan kaynak var:
      - ortak paylaşılabilenler (örn. dosya okuma)
      - ortak paylaşılamayanlar
        - fiziksel nedenlerle (örn. bazı G/Ç birimleri)
        - bir prosesin çalışması diğerini etkiliyorsa (örn. ortak bellek alanına yazma)
  - senkronizasyon
    - bir prosesin çalışmasını sürdürmesi, bir diğer prosesin belirli işlemleri tamamlamış olmasına bağlı olma durumu

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Etaner-Uyar

## Örnek

- 2 proses: Gözlemci ve Raporlayıcı
- sayaç ortak erişilen değişken (paylaşılan bellek)

```

gözlemci:          raporlayıcı:
while TRUE {       while TRUE {
    olayı_gözle;    yazdır_sayaç;
    sayaç=sayaç+1;  sayaç=0;
}                  }

```

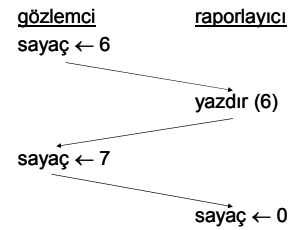
BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Örnek – Hata Oluşabilecek Durumlar

- durum 1:



BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Örnek – Hata Oluşabilecek Durumlar

- durum 2:

sayaç = sayaç + 1	YUKLE	ACC, SAYAC
	ARTIR	ACC
	YAZ	SAYAC, ACC

### Yarış durumu:

- ortak verilere erişen prosesler varsa
  - sonuçlar proseslerin çalışma hız ve sıralarına bağlı
  - her çalışmada farklı sonuç elde edilebilir
- istenmeyen durum: sistemin deterministik çalışması istenir

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Örnek – Hata Oluşabilecek Durumlar

### Proses P1:

```

while TRUE
    k=k+1;

```

### Proses P2:

```

while TRUE
    k=k+1;

```

k=0 (k:paylaşılan değişken, ilk değeri=0)

**P1 ve P2** proseslerinin farklı sıralarda kesilirmesinin k değeri üzerindeki etkisi ne olur?

**PROBLEM:** k değerinde

**tutarsızlık**

**ÇÖZÜM:** karşılıklı dışlama

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Ortak Bellek Kullanımı

- prosesler arası bellek paylaşımında (ortak değişkenler) 2 tür erişim sözkonusudur:
  - OKUMA (karşılıklı dışlama gerektirmez)
  - YAZMA (karşılıklı dışlama koşulları gereklidir)
- veri tutarlılığı açısından gerekli olan
  - Yazma erişiminin karşılıklı dışlama koşulları altında gerçekleşmesi
  - işlemlerde senkronizasyonun sağlanması

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Senkronizasyon

- proseslerin çalışma hızları, zamanları ve sıraları önceden kestirilemez
  - buna bağlı işlem yapılmamalı
- İşbirliği yapan proseslerin, çalışmalarının belirli noktalarında, işlemlerini senkronize etmeleri gerekebilir
  - örnek: bir proses, bir diğer prosesin üreteceği sonuç bilgisini elde edemeden çalışmasını sürdüremez

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama

### Tanım:

- Bir prosese ilişkin program kodunda, paylaşılan kaynaklar üzerinde işlem yapılan kısma kritik bölüm (KB) denir.
- Aynı kaynağa erişen KB kod parçasını bir anda bir prosesin yürütmesine izin verilir, diğer proseslerin benzer istekleri bekletilir  
⇒ karşılıklı dışlama koşullarında çalışma
- karşılıklı dışlama – mutual exclusion (mx)

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama - Örnek

### P1:

```
while TRUE {
  <KB olmayan kod>
  mx_begin
  <KB işlemleri>
  mx_end
  <KB olmayan kod>
}
```

### P2:

```
while TRUE {
  <KB olmayan kod>
  mx_begin
  <KB işlemleri>
  mx_end
  <KB olmayan kod>
}
```

BLG 312 – Bilgisayar İşletim Sistemleri

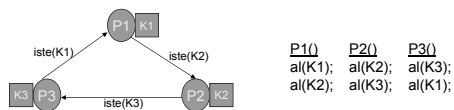
Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama – Olası Sorunlar

- ölümcül kilitlenmeye yol açabilecek durum:
  - aynı kaynaklara ihtiyaç duyan birden fazla proses var
  - her proses elde etmiş olduğu ve diğerinin de ihtiyaç duyduğu kaynağı bırakmıyor

**Örnek:** 3 proses ve 3 kaynak



BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama

- mx\_begin**
  - KB'sine girmiş ve henüz **mx\_end** yürütmemiş proses var mı?
    - evet:** **mx\_begin** yürütme isteğinde bulunan prosesin çalışmasını engelle
    - hayır:** **mx\_begin** yürütme isteğinde bulunan prosesin KB'sine ilerlemesine izin ver ve diğer prosesleri bu durumdan haberdar edecek işaret bırak
- mx\_end**
  - KB'sine ilerlemek isteyip de engellenmiş proses varsa çalışmasına izin ver. Aksi durumda, hiç bir prosesin KB içinde aktif olmadığını belirten bir işaret bırak

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama Koşulları Oluşturulurken Dikkat Edilecek Noktalar

- bir anda sadece bir proses KB içinde aktif olmalı
- KB'sine girmek isteyen bir proses, KB içinde aktif olan başka proses yoksa engellenmemeli
- KB'si dışında aktif olan bir proses, bir diğer prosesin KB'sine ilerlemesini engellememeli
- proses sayısı ve hızı ile ilgili kabuller yapılmamalı
- bir proses KB'si içinde sonsuza kadar kalamamalı
- bir proses KB'sine girmek için sonsuza kadar beklememeli

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Karşılıklı Dışlama Koşulları Nasıl Gerçeklenir

- yazılıma dayalı çözümler
- donanıma dayalı çözümler
- yazılım ve donanımı birlikte kullanan çözümler

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Yazılım ile Çözüm

- bir prosesin bir KB içinde olup olmadığını gösteren bayrak (meşgul) kullan:  
meşgul ← TRUE : KB'si içinde aktif proses var  
meşgul ← FALSE : KB'si içinde aktif proses yok
- **mx\_begin:** while (meşgul);  
meşgul = TRUE;  
/\* KB içinde çalışan proses işini bitirene kadar bekle ve KB'e ilerle \*/
- **mx\_end:** meşgul = FALSE;  
/\* KB içinde olmadığını gösteren işaret bırak \*/

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Yazılım ile Çözüm

- Önerilen çözüm HATALI - karşılıklı dışlama koşullarını sağlayamaz
- Çözüm olarak kullanılan *meşgul* değişkeninin kendisi de, erişimi denetim altında tutulmak istenen bir ortak bir değişken!  
■ Hatalı çalışma örneği:  
■ meşgul=False  
- P1: while (meşgul)—FALSE bulur ve çevrimden çıkar ve hemen çalışması kesilir  
- P2: while (meşgul)—FALSE bulur ve çevrimden çıkar, meşgul←TRUE ve KB'ne ilerler. Bir süre sonra kesilir  
- P1 kaldığı yerden çalışmaya başlar: meşgul←TRUE ve KB'ne ilerler  
SONUÇ: Her iki proses de KB içinde aktif→hatalı çalışma düzeni

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Meşgul Bekleme (*Busy Waiting*) Gerektiren Çözümler

```

global değişken sıra = 1;

Proses 1:          Proses 2:
lokal değişkenler  lokal değişkenler
benim_siram=1;     benim_siram=2;
diğeri_sirası=2;    diğeri_sirası=1;

mx_begin: while (sıra != benim_siram);

mx_end : sıra = diğeri_sirası;

```

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Meşgul Bekleme (*Busy Waiting*) Gerektiren Çözümler

- işlemci zamanı boşa harcanır-proses, sorgulama sonucu değişmeyeceği halde (*sorgulama sonucu ancak bir başka prosesin çalışmasıyla değişebilir*), kendine ayrılan zaman dilimini while çevrimi içinde dönerek geçirir: meşgul bekleme
- geçerli bir çözüm, ancak kısıtlamaları var:
  - prosesler sadece birbirlerini izleyen sırayla KB'ye girebilir
  - kendi KB'si dışındaki bir prosesin bir başka prosesin KB'ye girmesini engellememe ilkesine aykırı
    - örnek: proseslerden birinin daha yavaş çalışması durumunda, diğeri onu beklemek durumunda kalacaktır
  - proses sayısına bağlı çözüm - sadece iki proses için geçerli

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Meşgul Bekleme (*Busy Waiting*) Gerektiren Çözümler

- ilk geçerli çözüm: Dekker algoritması
- Petersen algoritması (1981)
  - Dekker algoritmasına benzer yaklaşım
  - daha kolay anlaşılır çözüm
  - Ancak, yine proses sayısı iki ile sınırlı

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Petersen Algoritması

- ortak değişkenler:

istek\_1, istek\_2: bool ve ilk değerler FALSE  
sıra: tamsayı ve ilk değeri "P1" veya "P2"

### Proses P1:

```
mx_begin:
  istek_1 = TRUE;
  sıra = P2;
  while (istek_2 && sıra==P2);
    < KB >
```

### Proses P2:

```
/* Proses P1'e
  benzer kod,
  ancak
  değişkenler
  farklı */
```

```
mx_end: istek_1 = FALSE;
```

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Petersen Algoritması

- değişik durumlar:
  - P1 çalışıyor, P2 pasif  
istek\_1=TRUE ve sıra=P2  
istek\_2=FALSE olduğu için P1 while döngüsünü  
geçer → P1 KB içinde
  - P1 KB'de, P2 KB'ye girmek istedi  
istek\_2=TRUE ve sıra=P1;  
istek\_1=TRUE olduğu için P2 while döngüsünde  
bekler
  - P1 mx\_end yürütünce P2 döngüden çıkar

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Petersen Algoritması

- (*değişik durumlar devam*):

- P1 ve P2 aynı anda KB'ye girmek ister

### P1:

```
a) istek_1=TRUE;
b) sıra=P2;
```

### P2:

```
c) istek_2=TRUE;
d) sıra=P1;
```

Çalışma sırası: abcd → P1 KB'ne girer  
acdb → P2 KB'ne girer

Sonuç:⇒ Aynı anda KB'e girmek isteyen proseslerden  
"sıra" değişkenine önce atama yapan KB'si  
öncelikle ilerleme hakkı kazanır.

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Donanıma Dayalı Çözümler

- kesilemeyen, tek bir komut çevriminde gerçekleşen özel makina komutları ile
  - örnek: test\_and\_set komutu
  - meşgul bekleme var: işlemci zamanı harcanır
  - bir proses KB'den çıkınca birden fazla bekleyen varsa kimin gireceği belirsiz
    - sonsuz bekleme olabilir
- kesmeleri kapatarak
  - işletim sisteminin iş sıralama algoritmasına müdahale, bu nedenle kullanıcı düzeyinde mümkün değil

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Semaforlar

- yazılım ve işl.sist. desteğine dayalı çözüm
- meşgul bekleme gerekmez
- işlemci zamanı harcanmaz
- proses sayısından bağımsız
- **semafor** adı verilen özel bir değişken
  - değişkene erişim iki özel işlem ile mümkün
  - değişkene ilk değer atamak mümkün
  - özel işlemler kesilemez işlemler
  - özel işlemler işletim sistemi tarafından gerçekleştirir

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Semaforlar

- s semafor değişkeni olsun
  - özel işlemler:
    - P işlemi (wait): KB'e girerken: mutex\_begin
    - V işlemi (signal): KB'den çıkarken: mutex\_end
- ```

P(s):          V(s):
if (s > 0)      if (s_üzerinde_bekleyen)
  s=s-1;        sıradakini_aktif_et;
else
  s_üzerinde_bekle;  s=s+1;
  
```

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Semaforlar

- semaforlar tamsayı değerleri alır ( $\geq 0$ )
- semafor değişkeni olduğunu belirten bir çağrı ile yaratılır
- semafora ilk değer atanır
- KB erişiminde kullanılan semafor sadece 0/1 değerlerini alabilir: **ikili semafor**
  - $\geq 0$  tamsayı değerler alan semaforlar: **sayma semaforu**

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Örnek: Gözlemci – Raporlayıcı

```
global değişkenler:
    sayaç: integer;
    sem: semafor;
proses gözlemci:
    olayı_gözle;
    P(sem);
    sayac=sayac+1;
    V(sem);
    ....
ana_program:
    sem=1; sayaç=0;
    canlandır(P1);
    canlandır(P2);
```

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Örnek: Gözlemci – Raporlayıcı

örnek çalışma:

P1: P(sem) ... sem=0;  
 P2: P(sem) ... sem=0 olduğu için P2 askıya  
 P1: V(sem) ... P2 sem için bekler; P2 aktive et  
 P2: V(sem) ... bekleyen yok; sem=1

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Semaforlar ile Senkronizasyon

- bir proses bir olayın gerçekleşmesini beklemek isteyebilir – askıya alınır
    - örneğin giriş işlemi bekleyen proses
  - olayın gerçekleştiğini algılayabilen bir başka proses, askıda prosesi uyarır
- ⇒ “askıya al – uyandır” senkronizasyonu

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar

## Semaforlar ile Senkronizasyon

▪ çözüm:

```
olay:semafor; olay=0;
proses P1:      proses P2:
    ...           ...
    P(olay);      ...
    ...           V(olay);
    ...           ...
```

- ikiden fazla proses de senkronize edilebilir

BLG 312 – Bilgisayar İşletim Sistemleri

Prof. Dr. Nadia Erdoğan

Doç. Dr. A. Şima Elaner-Uyar



## Semaforlar

### **DİKKAT!**

Semafor başlangıç değeri:

- Karşılıklı dışlama için 1
- Senkronizasyon için 0

## Semaforlar

- Ölümcül kilitlenmeye neden olabilecek bir durum:

$x, y$ : semafor;  $x=1; y=1$ ;

**proses 1:**

...

P (x) ;

...

P (y) ;

...

V (x) ;

V (y) ;

...

**proses 2:**

...

P (y) ;

...

P (x) ;

...

V (y) ;

V (x) ;

...

P ve V işlem  
sıralanma  
dikkat!