

Bölüm 6: Kilitlenme

İşletim Sistemleri

Öncelikli ve Öncelikli Olmayan Kaynaklar

- Bir kaynağı kullanmak için gereken olayların sırası:
- Kaynak iste
- Kaynağı kullan
- Kaynağı serbest bırak

Kaynak Edinimi

- Kaynakları korumak için semafor kullanma. (a) Bir kaynak. (b) İki kaynak.

```
typedef int semaphore;  
semaphore resource_1;
```

```
void process_A(void) {  
    down(&resource_1);  
  
    use_resource_1();  
    up(&resource_1);  
  
}
```

```
typedef int semaphore;  
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources();  
    up(&resource_2);  
    up(&resource_1);  
  
}
```

Kilitlenme İçermeyen Kod

```
typedef int semaphore;  
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources();  
    up(&resource_2);  
    up(&resource_1);  
}
```

```
void process_B(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources();  
    up(&resource_2);  
    up(&resource_1);  
}
```

Olası Bir Kilitlenme İçeren Kod

```
typedef int semaphore;  
semaphore resource_1;  
semaphore resource_2;
```

```
void process_A(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources();  
    up(&resource_1);  
    up(&resource_2);  
}
```

```
void process_B(void) {  
    down(&resource_1);  
    down(&resource_2);  
    use_both_resources();  
    up(&resource_2);  
    up(&resource_1);  
}
```

Kilitlenme

- Kilitlenme resmi olarak şu şekilde tanımlanabilir:
- Her bir süreç, aynı kümedeki başka bir sürecin neden olabileceği bir olayı bekliyorsa, bu süreç kümesi kilitlenir.

Kaynak Kilitlenme Koşulları

- Karşılıklı dışlama koşulu (mutual exclusion)
- Tut ve bekle durumu (hold and wait)
- Ön alım şartı yok (no preemption)
- Dairesel bekleme koşulu. (circular wait)

Kilitlenme Önleme

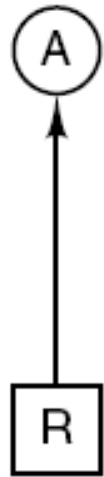
- Karşılıklı Dışlama: Aynı anda yalnızca bir süreç bir kaynağa erişebilir
- Tut ve Bekle: Bir kaynağı tutan bir süreç, yalnızca ilk kaynakla işini bitirdiğinde ihtiyaç duyduğu başka bir kaynak için istekte bulunmalıdır.
- Önleme Yok: Bir kaynağı tutan bir süreç, kaynaktan zorla çıkarılamaz
- Döngüsel Bekleme: Süreçler, her süreç bir sonraki süreç tarafından tutulan bir kaynağı bekleyecek şekilde döngüsel bir bekleme listesinde sıralanmalıdır.

Kilitlenme Tespiti

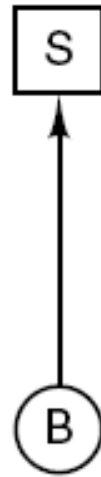
- Kaynak Tahsis Grafiği: Süreçlerin ve kaynakların, kaynak tahsisini temsil eden yönlendirilmiş kenarlara sahip bir grafikte temsili
- Döngüsel Tespit: Kilitlenmeyi belirlemek için kaynak tahsis grafiğinde bir döngünün tanımlanması
- Grafiği Bekle: Kaynak tahsis grafiğine benzer, ancak yönlendirilmiş kenarlar süreçler arasındaki bekleme ilişkilerini temsil eder
- Banker Algoritması: Bir durumun güvenli mi yoksa kilitlenmiş mi olduğunu belirlemek için maksimum ihtiyaç hakkındaki bilgileri kullanan algoritma.

Kilitlenme Modeli

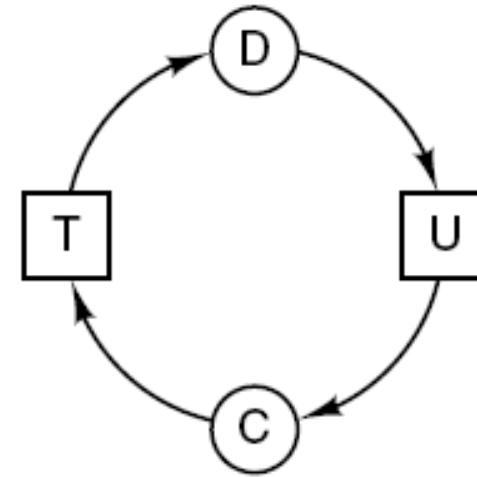
- Kaynak tahsis çizgeleri. (a) Bir kaynağı tutma. (b) Bir kaynak isteme. (c) Kilitlenme.



(a)



(b)



(c)

Kilitlenme Nasıl Oluşur?

• .

A
Request R
Request S
Release R
Release S

(a)

B
Request S
Request T
Release S
Release T

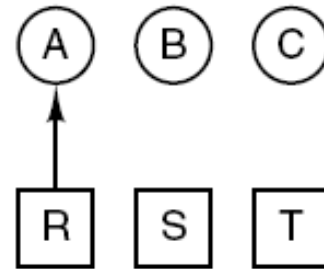
(b)

C
Request T
Request R
Release T
Release R

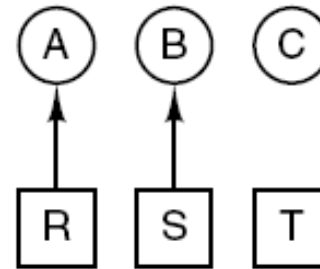
(c)

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
deadlock

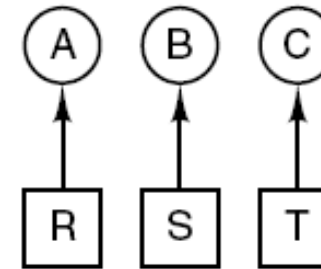
(d)



(e)



(f)

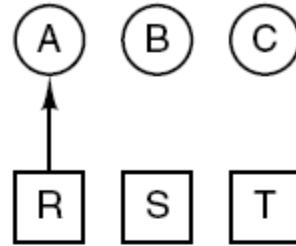


(g)

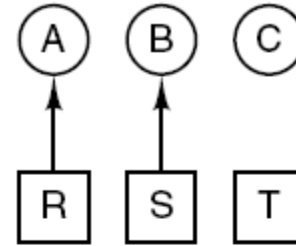
Kilitlenme Nasıl Oluşur?

- 1. A requests R
- 2. B requests S
- 3. C requests T
- 4. A requests S
- 5. B requests T
- 6. C requests R
deadlock

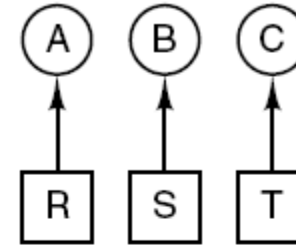
(d)



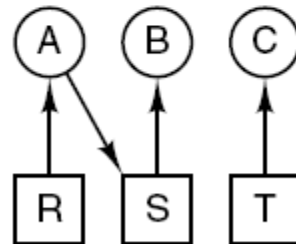
(e)



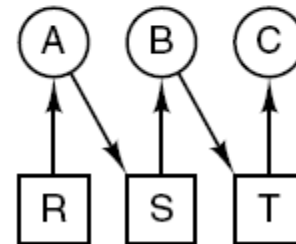
(f)



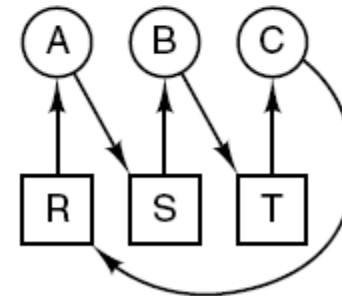
(g)



(h)



(i)

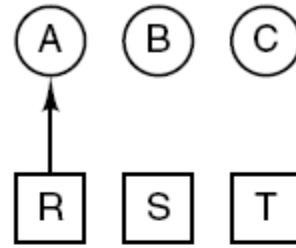


(j)

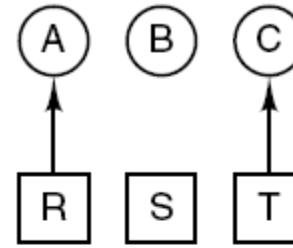
Kilitlenme Nasıl Önlenabilir?

- 1. A requests R
- 2. C requests T
- 3. A requests S
- 4. C requests R
- 5. A releases R
- 6. A releases S
no deadlock

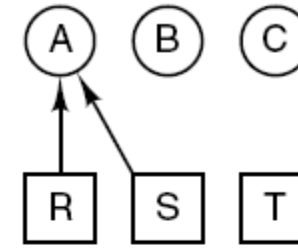
(k)



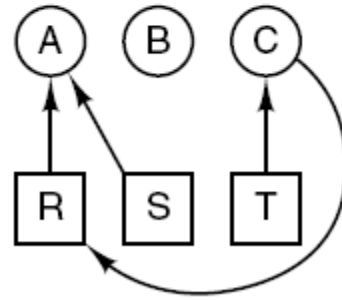
(l)



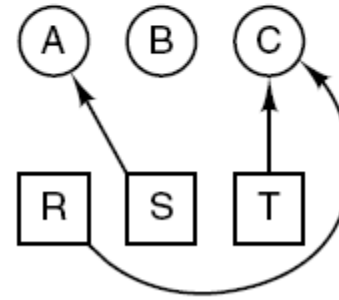
(m)



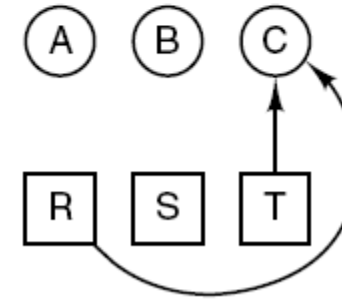
(n)



(o)



(p)



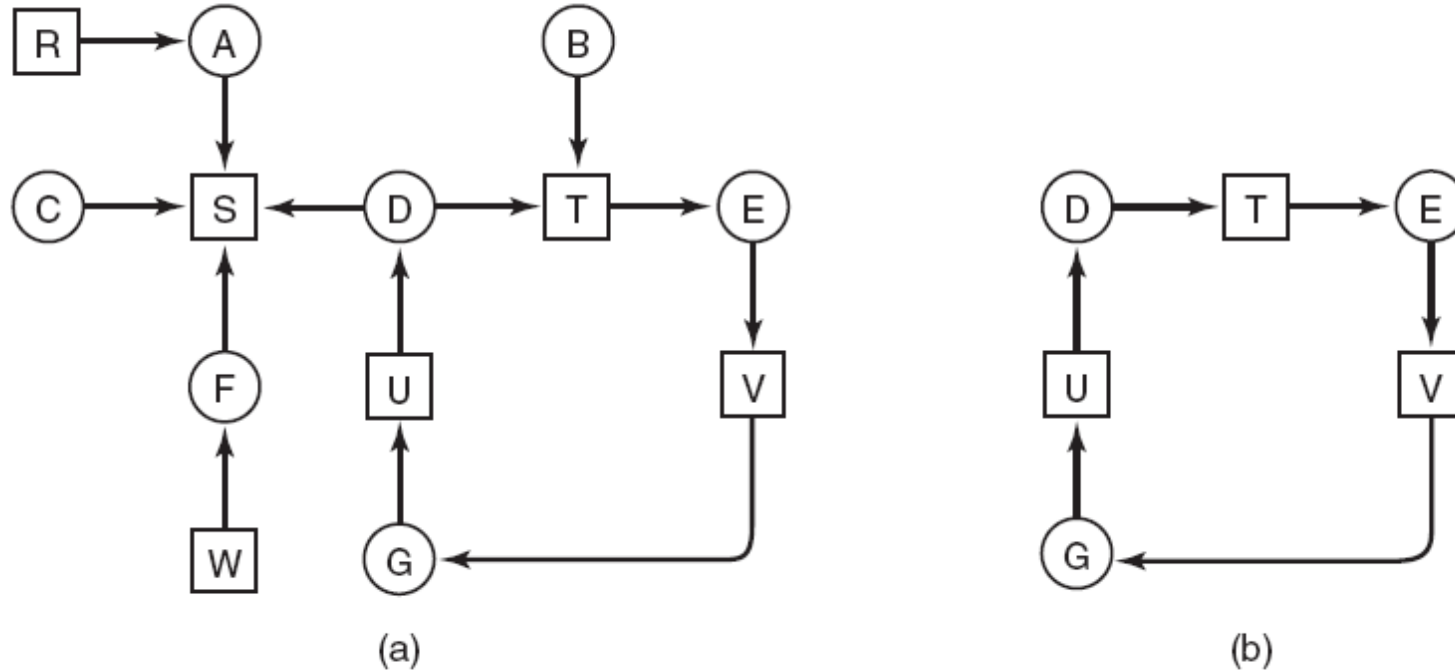
(q)

Kilitlenmelerle Başa Çıkma Stratejileri

- Sorunu görmezden gelme
- Tespit ve kurtarma. Kilitlenmeler yaşansın, tespit et, harekete geç
- Dikkatli kaynak tahsisi ile dinamik kaçınma.
- Gerekli dört koşuldan birini yapısal olarak reddederek önleme.

Kilitlenme Tespiti – Her Türden Bir Kaynak

- (a) Bir kaynak grafiği. (b) (a)'dan çıkarılan bir döngü.



Kilitlenme Tespit Algoritması

- Grafikteki her bir N düğümü için, başlangıç düğümü N olacak şekilde aşağıdaki beş adımı gerçekleştirin.
- L'yi boş liste olarak ilklendir, tüm okları işaretlenmemiş olarak ata.
- Mevcut düğümü L'nin sonuna ekle, düğümün L'de iki kez olup olmadığını kontrol et. Varsa, çizge bir döngü içerir ve algoritma sona erer.

Kilitlenme Tespit Algoritması

- Verilen düğümden giden işaretlenmemiş bir ok olup olmadığına bak. Varsa, 5. adıma git; yoksa, 6. adıma git.
- Rastgele işaretlenmemiş bir ok seç ve işaretle. Ardından onu yeni geçerli düğüme kadar takip et ve 3. adıma git.
- Eğer düğüm ilk düğüm (initial) ise, çizge herhangi bir döngü içermez, algoritma sona erer. Aksi takdirde çıkmaz sokak. Düğümü listeden çıkar, önceki düğüme geri dön, bu düğümü geçerli düğüm yap, 3. adıma git.


Kilitlenme Tespiti – Her Türden Çoklu Kaynak

- Kilitlenme tespit algoritması dört veri yapısına ihtiyaç duyar

Resources in existence
($E_1, E_2, E_3, \dots, E_m$)

Resources available
($A_1, A_2, A_3, \dots, A_m$)


Current allocation matrix



| | | | | |
|----------|----------|----------|---------|----------|
| C_{11} | C_{12} | C_{13} | \dots | C_{1m} |
| C_{21} | C_{22} | C_{23} | \dots | C_{2m} |
| \vdots | \vdots | \vdots | | \vdots |
| C_{n1} | C_{n2} | C_{n3} | \dots | C_{nm} |

Row n is current allocation
to process n

Request matrix



| | | | | |
|----------|----------|----------|---------|----------|
| R_{11} | R_{12} | R_{13} | \dots | R_{1m} |
| R_{21} | R_{22} | R_{23} | \dots | R_{2m} |
| \vdots | \vdots | \vdots | | \vdots |
| R_{n1} | R_{n2} | R_{n3} | \dots | R_{nm} |

Row 2 is what process 2 needs

Kilitlenme Tespit Algoritması

- R'nin i'inci satırının A'dan küçük veya ona eşit olduğu, işaretlenmemiş bir P_i süreci ara.
- Böyle bir süreç bulunursa, A'ya C'nin i'inci satırını ekle, süreci işaretle ve 1. adıma geri dön.
- Böyle bir süreç yoksa, algoritma sona erer.

Kilitlenme Tespiti İçin Bir Örnek

-

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

Tape drives
Plotters
Scanners
CD Roms

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

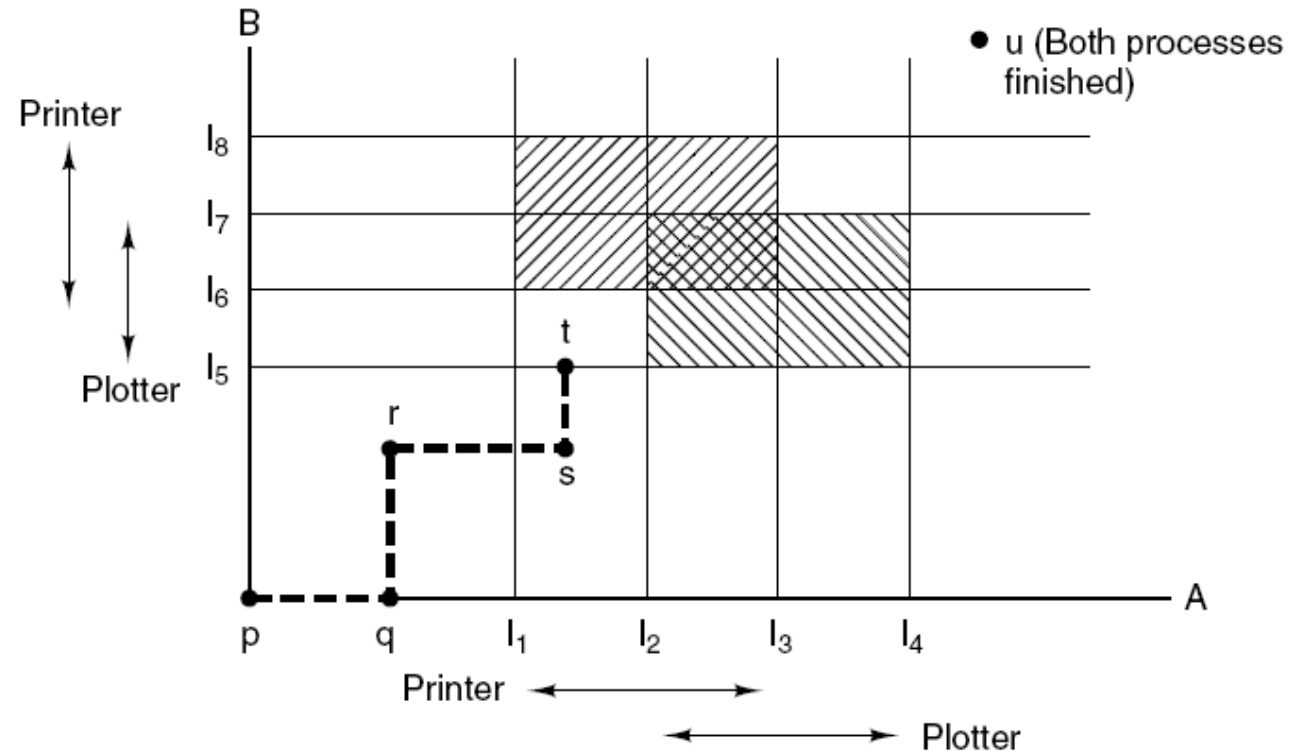
$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Kilitlenmeden Kurtarma (recovery)

- Ön alım yoluyla kurtarma (preemption)
- Geri alma yoluyla kurtarma (rollback)
- Süreçleri öldürme yoluyla kurtarma (kill process)

Kilitlenmeden Kaçınma

- İki süreç kaynağı eğrisi (trajectory)



Güvenli ve Güvensiz Durumlar

- (a)'daki durumun güvenli olduğunun gösterimi

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 2 | 4 |
| C | 2 | 7 |

Free: 3
(a)

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 4 | 4 |
| C | 2 | 7 |

Free: 1
(b)

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 0 | – |
| C | 2 | 7 |

Free: 5
(c)

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 0 | – |
| C | 7 | 7 |

Free: 0
(d)

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 0 | – |
| C | 0 | – |

Free: 7
(e)

Güvenli ve Güvensiz Durumlar

- (b)'deki durumun güvensiz olduğunun gösterimi

| Has Max | | |
|---------|---|---|
| A | 3 | 9 |
| B | 2 | 4 |
| C | 2 | 7 |

Free: 3

(a)

| Has Max | | |
|---------|---|---|
| A | 4 | 9 |
| B | 2 | 4 |
| C | 2 | 7 |

Free: 2

(b)

| Has Max | | |
|---------|---|---|
| A | 4 | 9 |
| B | 4 | 4 |
| C | 2 | 7 |

Free: 0

(c)

| Has Max | | |
|---------|---|---|
| A | 4 | 9 |
| B | — | — |
| C | 2 | 7 |

Free: 4

(d)

The Banker's Algoritması – Tek Kaynak

- Üç kaynak tahsis durumu: (a) Güvenli. (b) Güvenli. (c) Güvensiz.

| Has Max | | |
|---------|---|---|
| A | 0 | 6 |
| B | 0 | 5 |
| C | 0 | 4 |
| D | 0 | 7 |

Free: 10

(a)

| Has Max | | |
|---------|---|---|
| A | 1 | 6 |
| B | 1 | 5 |
| C | 2 | 4 |
| D | 4 | 7 |

Free: 2

(b)

| Has Max | | |
|---------|---|---|
| A | 1 | 6 |
| B | 2 | 5 |
| C | 2 | 4 |
| D | 4 | 7 |

Free: 1

(c)

The Banker's Algoritması – Çoklu Kaynak

• .

| | Process | Tape drives | Plotters | Printers | CD ROMs |
|---|---------|-------------|----------|----------|---------|
| A | 3 | 0 | 1 | 1 | |
| B | 0 | 1 | 0 | 0 | |
| C | 1 | 1 | 1 | 0 | |
| D | 1 | 1 | 0 | 1 | |
| E | 0 | 0 | 0 | 0 | |

Resources assigned

| | Process | Tape drives | Plotters | Printers | CD ROMs |
|---|---------|-------------|----------|----------|---------|
| A | 1 | 1 | 0 | 0 | |
| B | 0 | 1 | 1 | 2 | |
| C | 3 | 1 | 0 | 0 | |
| D | 0 | 0 | 1 | 0 | |
| E | 2 | 1 | 1 | 0 | |

Resources still needed

E = (6342)
P = (5322)
A = (1020)

The Banker's Algoritması – Çoklu Kaynak

- Karşılanmayan kaynağının tamamı $\leq A'$ 'ya ihtiyaç duyan R satırını arayın. Böyle bir satır yoksa, hiçbir süreç tamamlanamayacak olduğundan sistem sonunda kilitlenecektir.
- Seçilen satırdaki işlemin ihtiyaç duyduğu tüm kaynakları istediğini ve sonlandığını varsayalım. Süreci sonlandırıldı olarak işaretleyin, tüm kaynaklarını A vektörüne ekle.
- Tüm işlemler sonlandırıldı (ilklendirme durumu güvenliydi) olarak işaretlenene kadar **veya** kaynak gereksinimleri karşılanabilecek hiçbir süreç kalmayana (bir kilitlenme var) kadar 1. ve 2. adımları tekrarlayın.

Kilitlenme Önleme

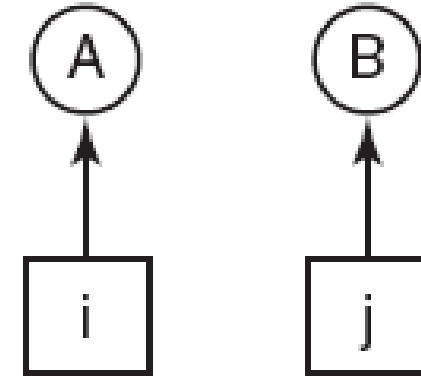
- Karşılıklı dışlama koşuluna saldırmak (mutual exclusion)
- Tut ve bekle durumuna saldırmak (hold and wait)
- Ön alım yok koşuluna saldırmak (no preemption)
- Döngüsel bekleme koşuluna saldırmak (circular wait)

Döngüsel Bekleme Koşuluna Saldırmak

- (a) Sayısal olarak sıralanmış kaynaklar. (b) Bir kaynak grafiği.

1. Imagesetter
2. Scanner
3. Plotter
4. Tape drive
5. CD-ROM drive

(a)



(b)

Kilitlenme Önleme Yaklaşımları

-

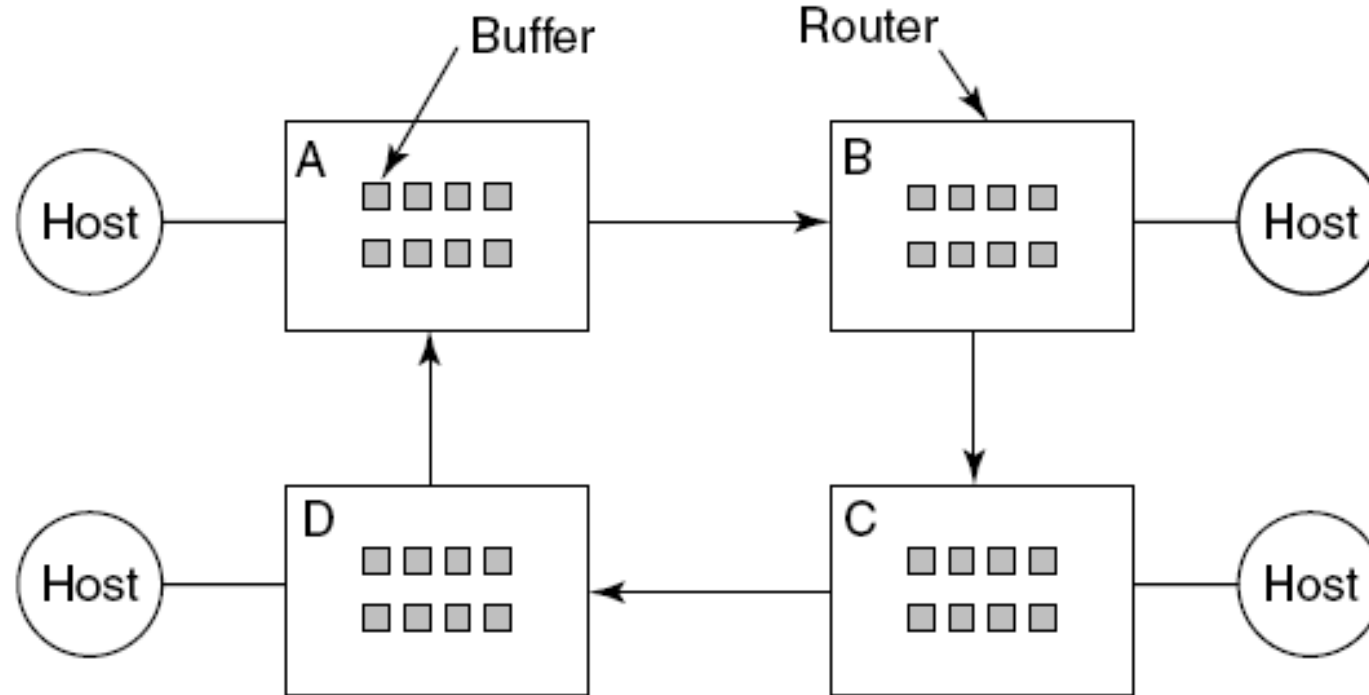
| Condition | Approach |
|------------------|---------------------------------|
| Mutual exclusion | Spool everything |
| Hold and wait | Request all resources initially |
| No preemption | Take resources away |
| Circular wait | Order resources numerically |

Diğer Sorunlar

- İki fazlı kilitleme (two phase locking)
- İletişim kilitlenmeleri (communication deadlocks)
- Canlı kilit (live lock)
- Açlık (starvation)

İletişim Kilitlenmeleri

- Bir ağda kaynak kilitlenmesi.



Canlı Kilit

- Canlı Kilit'e (livelock) yol açabilecek meşgul bekleme (busy waiting).

```
void process_A(void) {  
    enter_region(&resource_1);  
    enter_region(&resource_2);  
    use_both_resources( );  
    leave_region(&resource_2);  
    leave_region(&resource_1);  
}
```

```
void process_B(void) {  
    enter_region(&resource_2);  
    enter_region(&resource_1);  
    use_both_resources( );  
    leave_region(&resource_1);  
    leave_region(&resource_2);  
}
```

SON