



Bölüm 7: Kilitlenme

İşletim Sistemleri



Sistem Modeli

- Sistem kaynaklardan oluşur.
- Kaynak türleri R_1, R_2, \dots, R_m
 - CPU döngüleri, bellek alanı, I/O cihazları gibi.
- Her kaynak tipi R_i , somut nesnelere (*instance*) W_i sahiptir.
- Her süreç bir kaynağı:
 - İster,
 - Kullanır,
 - Serbest bırakır.



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 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

- Bir dizi sürecin bloke olduğu durumdur.
 - Her süreç *bir kaynağı* tutar, ve
 - Başka bir süreç tarafından tutulan *bir kaynağı* bekler.
- Örneğin, iki tren aynı hatta ve bir ray var, trenlerin hiçbiri hareket edemiyor.
- Bazı kaynakları tutarken, diğerleri tarafından tutulan kaynakları bekleyen iki veya daha fazla süreç olduğunda ortaya çıkar.
- Örneğin,
 - **Süreç 1**, **Kaynak 1**'i tutuyor ve
 - **Süreç 2** tarafından tutulan **Kaynak 2**'yi bekliyor ve
 - **Süreç 2**, **Kaynak 1**'i bekliyor.



Kilitlenme Koşulları

- Karşılıklı dışlama (*mutual exclusion*)
 - Bir kaynağı aynı anda yalnızca bir iş parçacığı kullanabilir.
- Tut ve bekle durumu (*hold and wait*)
 - Bir kaynağı tutan iş parçacığı, diğerlerinin tuttuğu kaynakları bekliyor.
- Kesilme yok (*no preemption*)
 - Kaynak, yalnızca onu tutan iş parçacığı tarafından serbest bırakılabilir.
- Döngüsel bekleme (*circular wait*)
 - T_0 , T_1 tarafından tutulan bir kaynağı bekler,
 - T_1 , T_2 tarafından tutulan bir kaynağı bekler, ...



Kilitlenme Önleme

- **Karşılıklı Dışlama:** Aynı anda yalnızca bir süreç bir kaynağa erişebilir.
- **Tut ve Bekle:** Süreç, yalnızca bir kaynağı serbest bıraktığında başka bir kaynak için istekte bulunmalıdır.
- **Kesilme Yok:** Bir kaynağı tutan süreç, kaynağı zorla bırakamaz.
- **Döngüsel Bekleme:** Süreçler, bir sonraki süreç tarafından tutulan bir kaynağı bekleyecek şekilde, döngüsel bekleme listesinde sıralanmalıdır.



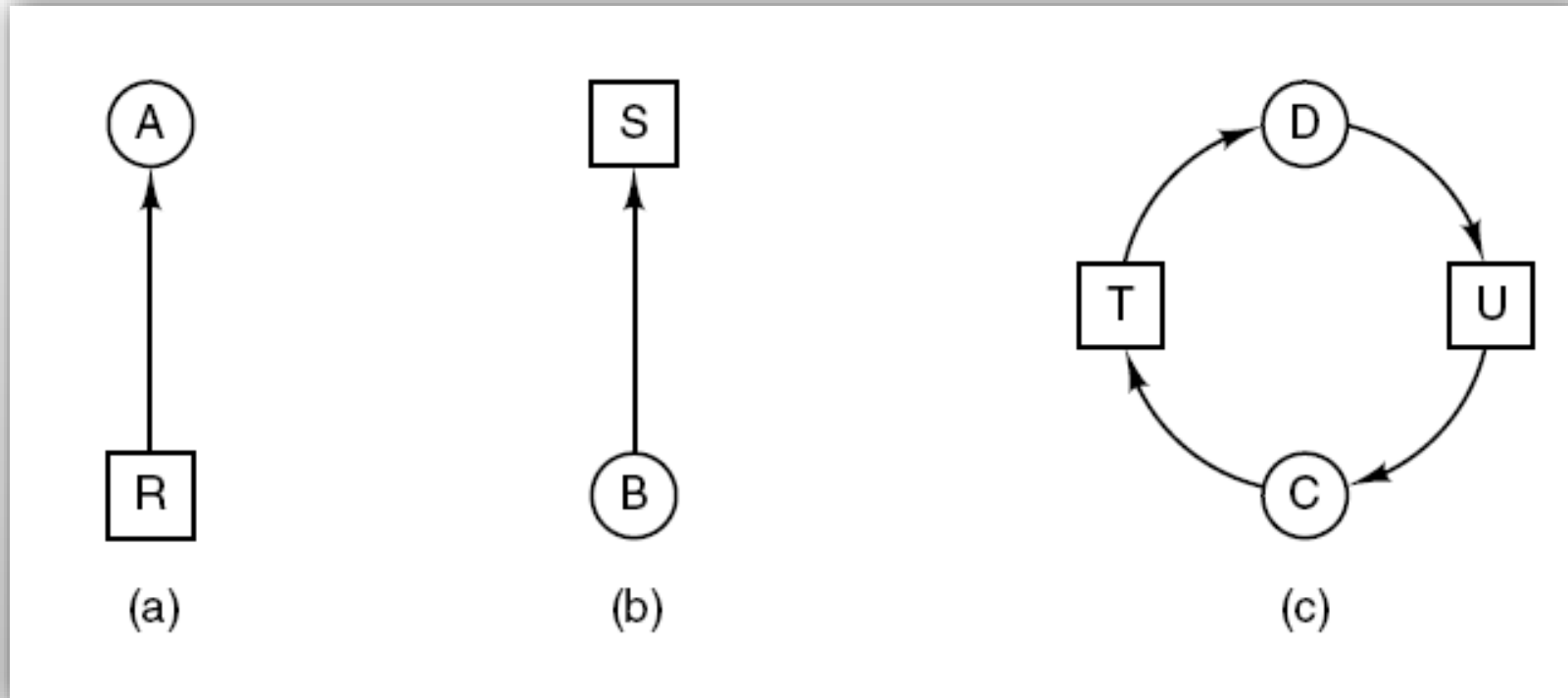
Kilitlenme Tespiti

- **Kaynak Tahsis Grafiği:** Süreçlerin ve kaynakların, kaynak tahsisini temsil eden yönlü kenarlara sahip bir çizge.
- **Döngüsel Tespit:** Kaynak tahsis grafiğinde bir döngünün bulunması.
- **Bekleme Grafiği:** Kaynak tahsis grafiğine benzer, ancak yönlü kenarlar süreçler arasındaki bekleme ilişkilerini temsil eder.
- **Banker Algoritması:** Bir durumun güvenli mi, kilitlenmiş mi olduğunu belirlemek için, maksimum ihtiyaç hakkındaki bilgileri kullanan algoritma.



Kaynak Tahsis Çizgesi

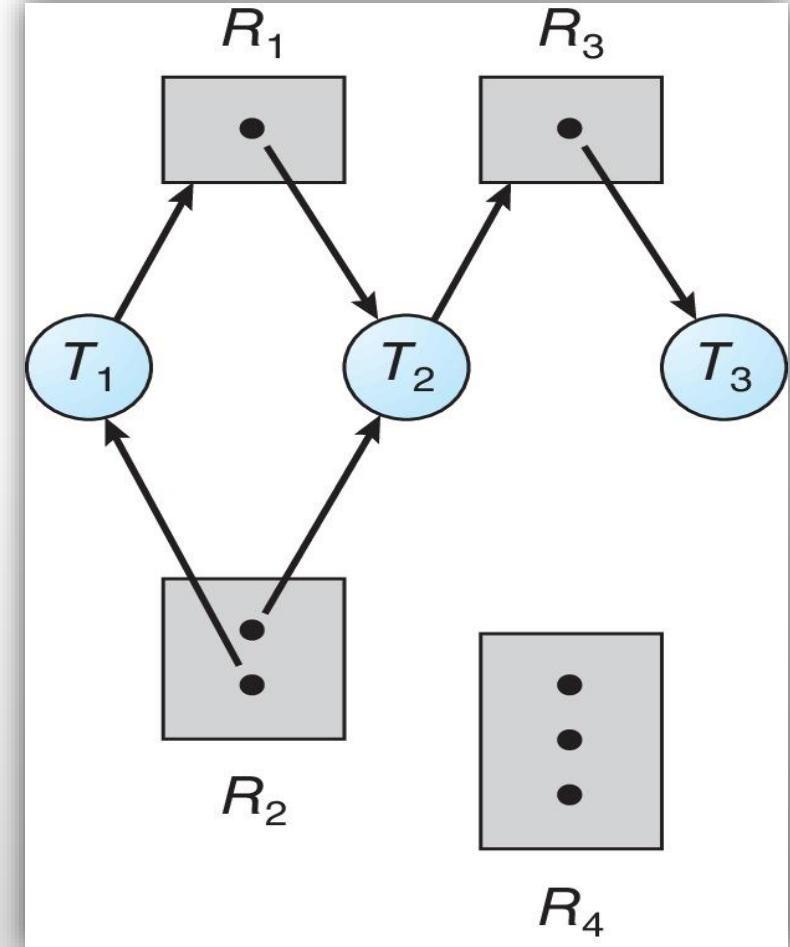
- (a) Bir kaynağı tutma. (b) Bir kaynak isteme. (c) Kilitlenme.





Kaynak Tahsis Çizgesi

- Bir tane R_1 kaynağı,
- İki tane R_2 kaynağı,
- Bir tane R_3 kaynağı,
- Üç tane R_4 kaynağı,
- T_1 , bir R_2 kaynağını tutar, ve
 - R_1 kaynağını bekler.
- T_2 , R_1 kaynağını,
 - bir R_2 kaynağını tutar, ve
 - R_3 kaynağını bekler.
- T_3 , R_3 kaynağını tutar.





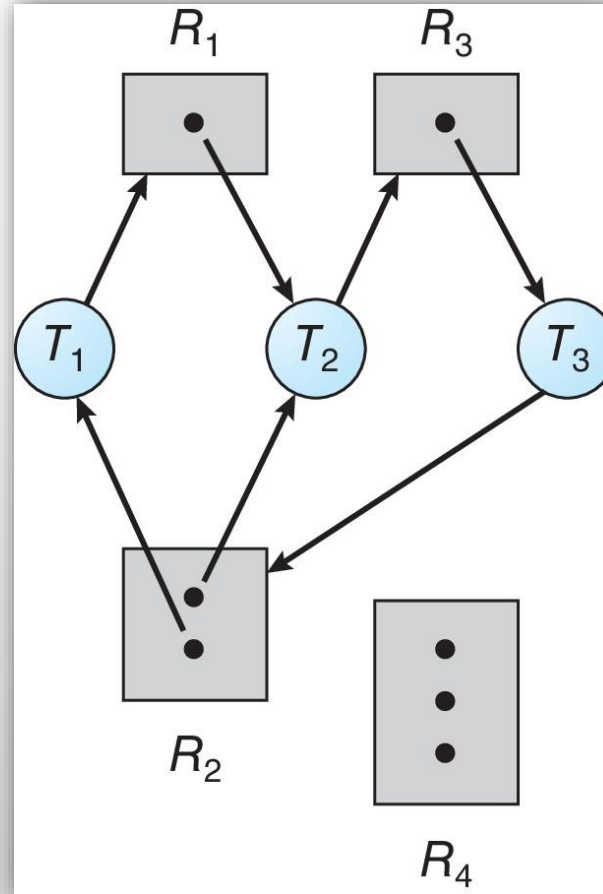
Kaynak Tahsis Çizgesi

- Çizge, döngü içermiyorsa,
 - kilitlenme **yok**.
- Döngü içeriyorsa,
 - o kaynak türünde yalnızca bir kaynak varsa, kilitlenme **var**.
 - o kaynak türünde birçok kaynak varsa, kilitlenme **olasılığı var**.



Kilitlenme İçeren Kaynak Tahsis Çizgesi

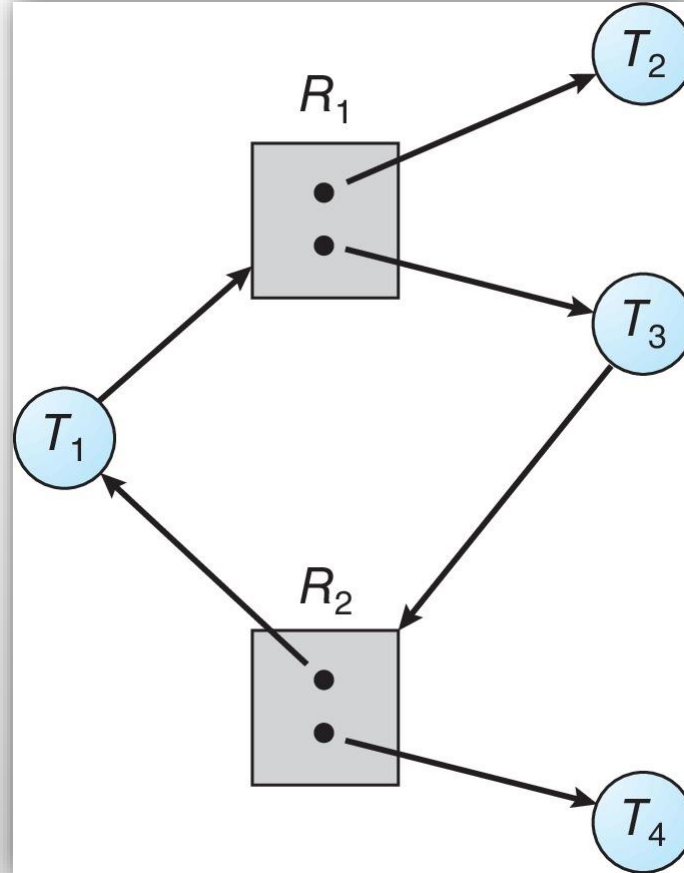
■ .





Kilitlenme İçermeyen Kaynak Tahsis Çizgesi

■ .





Kilitlenme Nasıl Oluşur?

■ .

A
Request R
Request S
Release R
Release S

(a)

B
Request S
Request T
Release S
Release T

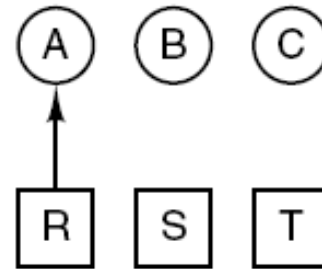
(b)

C
Request T
Request R
Release T
Release R

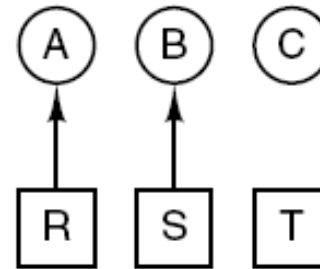
(c)

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

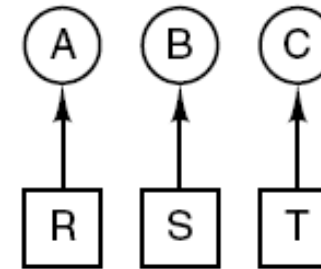
(d)



(e)



(f)



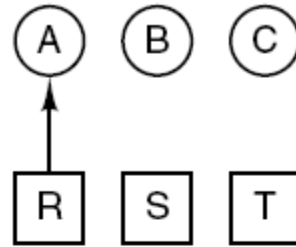
(g)



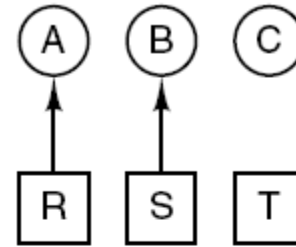
Kilitlenme Nasıl Oluşur?

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

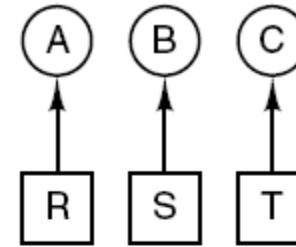
(d)



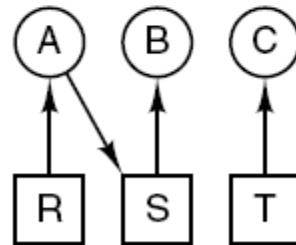
(e)



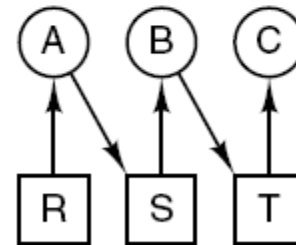
(f)



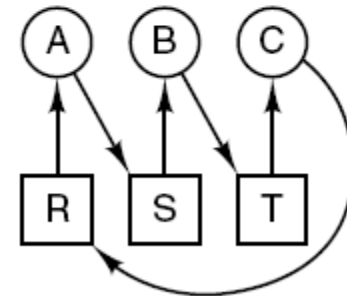
(g)



(h)



(i)



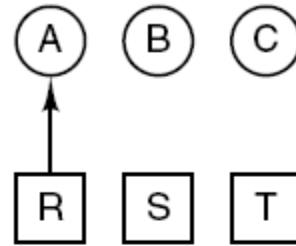
(j)



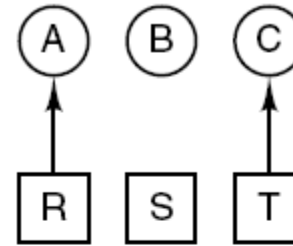
Kilitlenme Nasıl Önlenebilir?

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

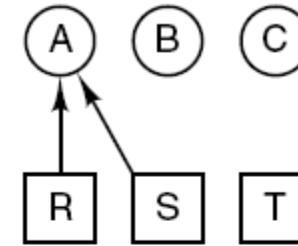
(k)



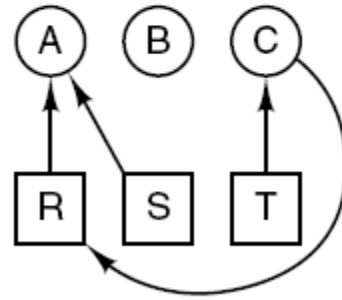
(l)



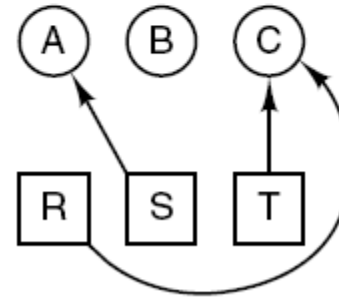
(m)



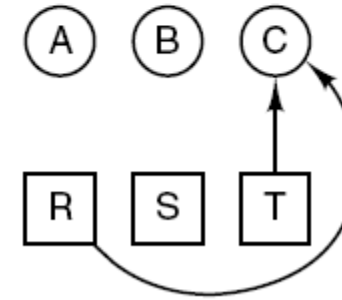
(n)



(o)



(p)



(q)



Kilitlenmelerle Başa Çıkma Stratejileri

- Sorunu görmezden gelme. 😊
- Sorunu tespit ve kurtarma.
- Kaynak tahsisi dikkatli yapılarak, dinamik olarak kilitlenmeden kaçınma.
 - en büyük kozumuz yakalanmamak 😊
- Gerekli dört koşuldan biri reddedilerek önlenir.
 - Karşılıklı dışlama (*mutual exclusion*)
 - Tut ve bekle durumu (*hold and wait*)
 - Kesilme yok (*no preemption*)
 - Döngüsel bekleme (*circular wait*)



Devekuşu (Ostrich) Algoritması

- Deve kuşları, kafalarını toprağa gömüp sorunları görmezden gelir.
- Kilitlenmeleri tespit ve kaçınma arasında bir denge kurmayı amaçlar.





Devekuşu (Ostrich) Algoritması

- Kilitlenmeleri geçici olarak yok sayar ve istekleri işlemeye devam eder.
- Kilitlenme meydana geldiğinde sistem, pasif bir moda girer.
 - Normal mod,
 - kilitlenme olup olmadığı kontrol edilmeden istekler işlenir.
 - Pasif mod,
 - kilitlenmeler periyodik kontrol edilir.
 - kilitlenme algılanırsa, kaynak serbest bırakılır, normal moda döner.
- Problem nadiren gerçekleşirse tercih edilebilir.
- Tespit ve önleme maliyetleri düşüktür.



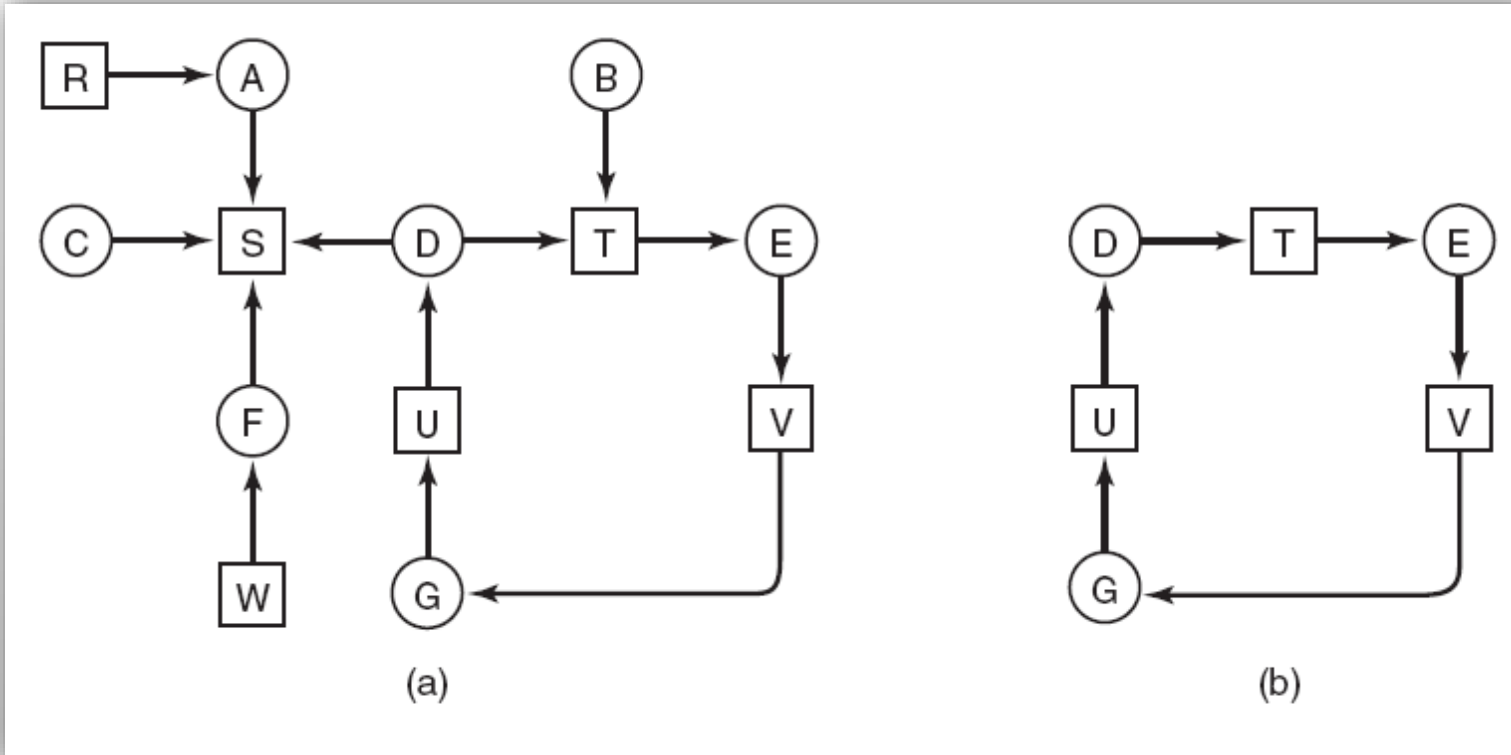
Kilitlenme Tespiti

- Her bir süreç ve kaynak, çizgede bir *düğüm* ile temsil edilir.
- Süreç ile kaynak arasındaki ilişkiyi bir *kenar* temsil eder.
- Döngü tespiti için Derinlik Öncelikli Arama (DFS) gibi algoritmalar kullanır.
- Döngü varsa, kilitlenme vardır.
- Her sürece atanan kaynaklar bir veri yapısında tutulur.
- Süreçler kaynak alıp, serbest bıraktıkça kaynak tahsisi çizgesi güncellenir.
- Kilitlenme algılanırsa,
 - ilgili süreçlerden biri iptal edilir, veya
 - süreçler tarafından tutulan kaynakların bir kısmı serbest bırakılır.



Kilitlenme Tespiti – Her Türden Bir Kaynak

- (a) Bir kaynak çizgesi. (b) (a)'da tespit edilen bir döngü.





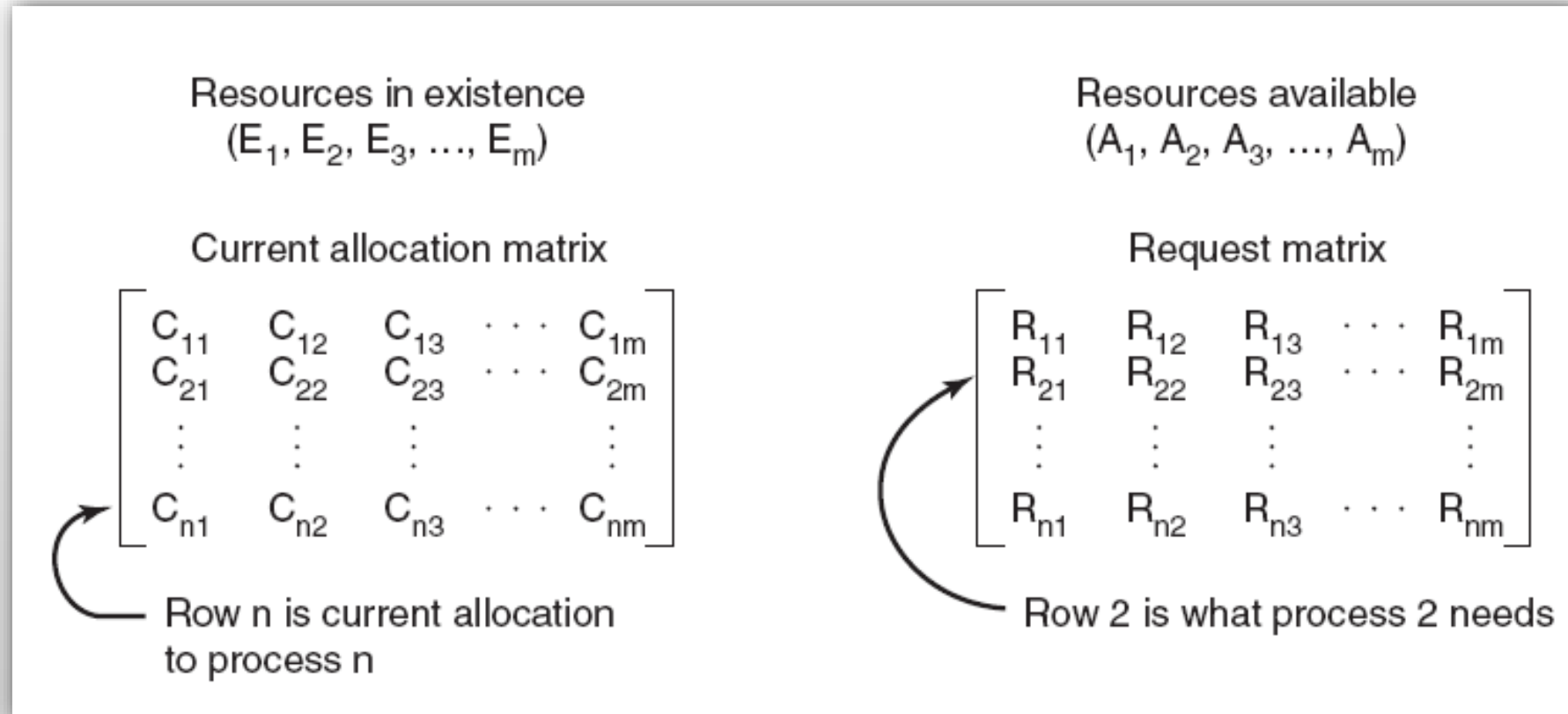
Kilitlenme Tespit Algoritması

1. Çizgedeki her bir N düğümü için, N düğümünden başlayarak;
2. L 'yi boş liste olarak, tüm okları işaretlenmemiş olarak ata.
3. Mevcut düğümü L 'nin sonuna ekle.
 1. Düğüm L 'de iki kez varsa, çizge döngü içerir, algoritma sona erer.
4. Mevcut düğümünden giden işaretlenmemiş ok,
 1. varsa, 5. adıma git;
 2. yoksa, 6. adıma git.
5. Oklardan birini seç, işaretle, okun gösterdiği düğüme git, 3. adıma git.
6. Başlangıç düğüme gelindi ise, çizge döngü içermez, algoritma sona erer.
7. Aksi takdirde düğümü listeden çıkar, önceki düğüme dön, 3. adıma git.



Kilitlenme Tespiti – Her Türden Çoklu Kaynak

- Kilitlenme tespit algoritması dört tane veri yapısı kullanır.





Kilitlenme Tespit Algoritması

C: *current allocation matrix* (var olan kaynakları tutar).

R: *request matrix* (istekte bulunulan kaynakları tutar).

A: *available resource count* (serbest olan kaynak sayısı).

1. **R** matrisinin, **i** satırında, $\leq \mathbf{A}$, işaretlenmemiş bir **P_i** süreci ara.
2. Bulunursa, **C**'nin **i**'inci satırına **A**'yı ekle, süreci işaretle, 1. adıma dön.
3. 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)

- Önleme (*prevention*): süreçlerin kaynak talep etmesini kısıtlar.
- Tespit (*detection*): kilitlenmeyi tespit etmek için sistemi izler.
- Geri alma (*rollback*): güvenli duruma geri dönmeyi sağlar.
- Sonlandırma (*termination*): süreci sonlandırır (*killing*).
- Çağrı üstünlüğü (*preemption*): kaynağı bir süreçten alıp başkasına verir.
- Kaynak tahsis ayarlaması (*setting allocated resource count*): sürece tahsis edilen kaynakların sayısının dinamik olarak ayarlanmasını içerir.



Güvenli ve Güvensiz Durumlar

- (a) tablosunda verilen durum güvenli,
- Süreçler B->C->A sırasında çalıştırılır.

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) tablosunda verilen durum güvensiz.

	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

- Mevcut kaynakları ve her bir sürecin maksimum talebini takip eder.
- Sürecin talebi sistemi güvensiz bir durumda bırakmıyorsa, kaynaklar verilir.
- Sürecin talebi, güvensiz bir duruma yol açıyorsa, ertelenir.
- Tahsis ve kaynak ihtiyacını takip etmek için bir matris kullanır.
- Kilitlenmeye yol açmadan sonlanan süreç varsa durum güvenli kabul edilir.
- Algoritma,
 - tüm süreçler tamamlandığında veya
 - kilitlenme meydana geldiğinde sona erer.



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

- Her sürece, bir dizi kaynak türü ve talep edilen maksimum miktarı atanır.
- Algoritma, her sürece tahsis edilen kaynakları ve mevcut kaynakları izler.
- Sürecin talebi,
 - yeterli miktarda kaynak varsa ve
 - kilitlenmeye neden olmuyorsa karşılanır.
- Algoritma, kaynak tahsisini yeterli kaynak sağlanana kadar erteleyebilir.
- Kaynak tahsisini izlemek ve kontrol için arka planda sürekli çalışır.



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

R: *request matrix* (istekte bulunulan kaynakları tutar).

A: *available resource count* (serbest olan kaynak sayısı).

1. **R** matrisinde, $\leq \mathbf{A}$ kaynağa ihtiyaç duyan satır arar.
 - Böyle satır yoksa, hiçbir süreç tamamlanmayacak, sistem kilitlenecektir.
2. Seçili satırdaki süreci, işaretle, ve tüm kaynaklarını **A**'ya ekler.
3. İşaretlenmemiş ve talebi karşılanabilen süreç oldukça adımları tekrarlar.
4. Tüm süreçler işaretlendi ise kilitlenme yoktur.
5. Talebi karşılanmayan süreç varsa, kilitlenme vardır.



Kilitlenme Önleme

- **Karşılıklı dışlama** (*mutual exclusion*) koşulunu kaldırma
 - Her seferinde yalnızca bir süreç kaynak isteyebilir ve tahsis edilebilir.
 - Kaynaklar sıralanır veya kaynağın birden çok örneğine izin verilir.
- **Tut ve bekle** (*hold and wait*) durumunu kaldırma
 - Süreçlerin başkalarını beklerken kaynakları tutması önlenir.
 - Süreç yürütülmeden önce ihtiyaç duyduğu tüm kaynakları talep eder.



Kilitlenme Önleme

- **Kesilme yok** (*no preemption*) koşulunu kaldırmak
 - Kaynak, yüksek öncelikli süreç tarafından önceden kullanılmamalı.
 - Süreç bitene kadar kaynakları tutmasına izin verilir.
 - Süreç bloke olursa kaynakları serbest bırakır.
- **Döngüsel bekleme** (*circular wait*) koşulunu kaldırmak
 - Kaynak tahsis sıralaması tanımlar.
 - Kaynakları numaralandırarak veya
 - Kaynak hiyerarşisi kullanarak yapılabilir.

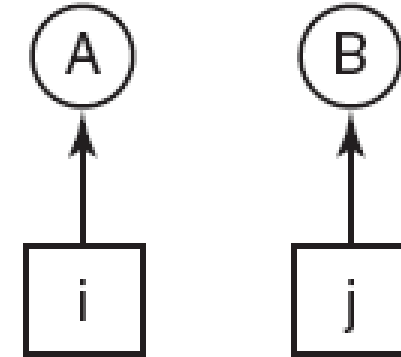


Döngüsel Bekleme Koşulunu Kaldırmak

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

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

(a)



(b)



Kilitlenme Önleme Yaklaşımları

Durum	Yaklaşım
Karşılıklı dışlama	Talepleri biriktir.
Tut ve bekle	Başlangıçta tüm kaynakları talep et.
Kesilme yok	Kaynakları serbest bırak.
Döngüsel bekleme	Kaynakları sayısal olarak sırala.



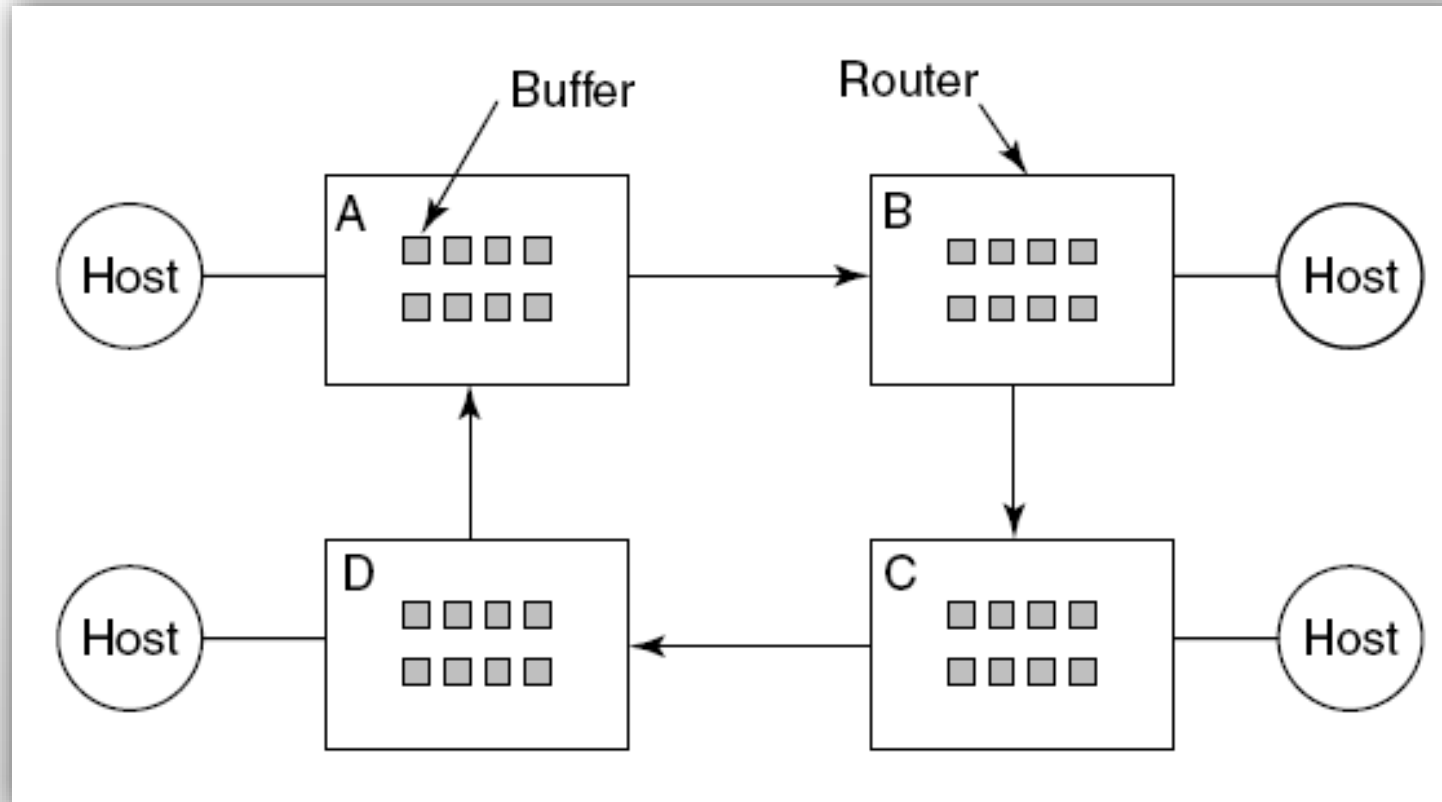
Diğer Sorunlar

- İki aşamalı kilitleme (*two phase locking*)
 - Yalnızca gerektiğinde kilit alınır ve en kısa sürede serbest bırakılır.
 - Veritabanı sistemlerinde kullanılan bir protokol.
- İletişim kilitlenmeleri (*communication deadlocks*)
 - Birden çok sürecin birbirinden yanıt beklediği sistemlerde oluşur.
- Canlı kilit (*live lock*)
 - Bir sürecin, diğerinin ileriye doğru bir adım atmasını beklediği durum.
- Açlık (*starvation*)
 - Bir sürecin sonlanmak için ihtiyaç duyduğu kaynakları,
 - elde etmesi engellendiğinde ortaya çıkar.



İletişim Kilitlenmeleri

- Bir ağda kaynak kilitlenmesi. Tampon bellekler dolu.





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