Bölüm 7: 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)
- Önleme yok (no preemption)
- Döngüsel bekleme koşulu. (circular wait)

Kilitlenme Önleme

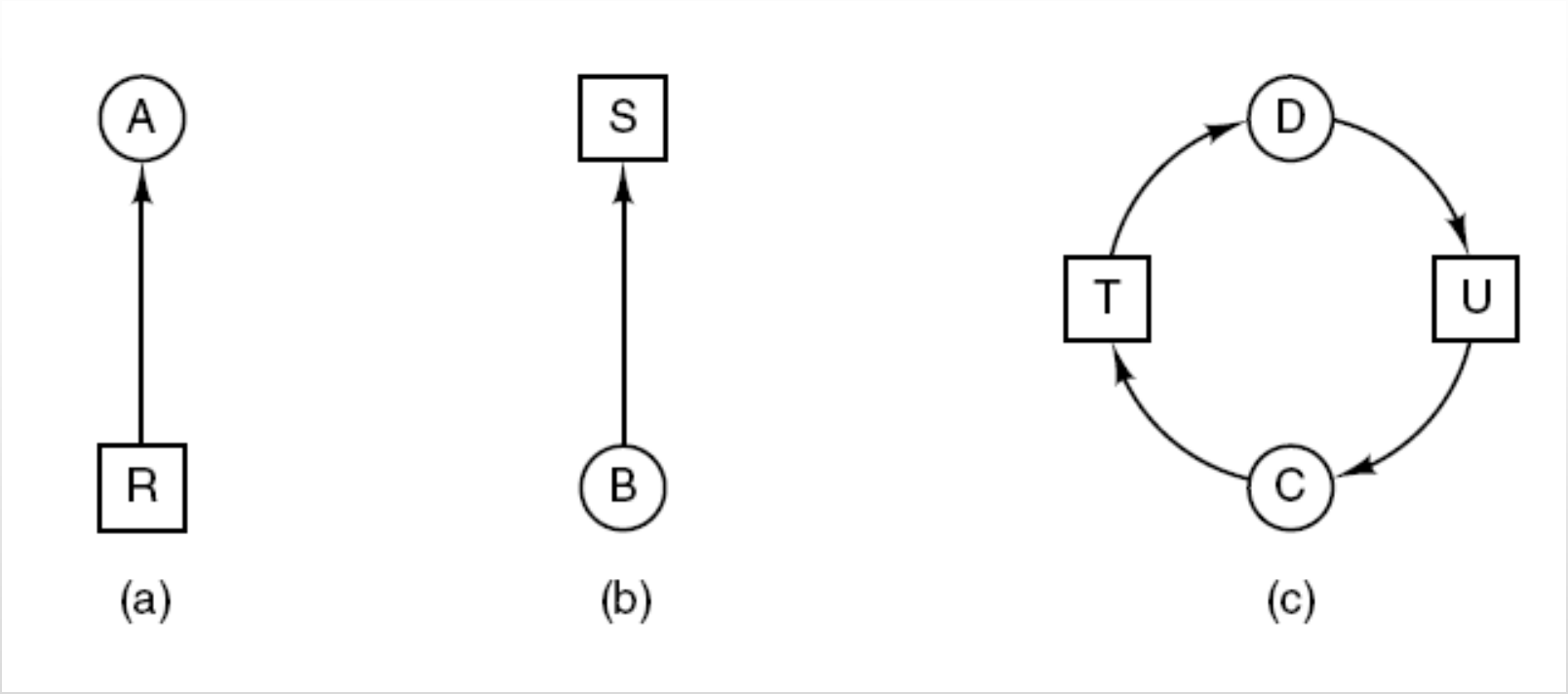
- **Karşılıklı Dışlama**: Aynı anda yalnızca bir süreç bir kaynağa erişebilir
- **Tut ve Bekle**: Bir kaynağı tutan bir süreç, yalnızca ilk kaynakla işini bitirdiğinde ihtiyaç duyduğu başka bir kaynak için istekte bulunmalıdır.
- **Önleme Yok**: Bir kaynağı tutan bir süreç, kaynaktan zorla çıkarılamaz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

Kilitlenme Tespiti

- Kaynak Tahsisi Grafiği: Süreçlerin ve kaynakların, kaynak tahsisini temsil eden yönlendirilmiş kenarlara sahip bir grafikte temsili
- Döngüsel Tespit: Kilitlenmeyi belirlemek için kaynak tahsisi grafiğinde bir döngünün tanımlanması
- Bekleme Grafiği: Kaynak tahsisi 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.

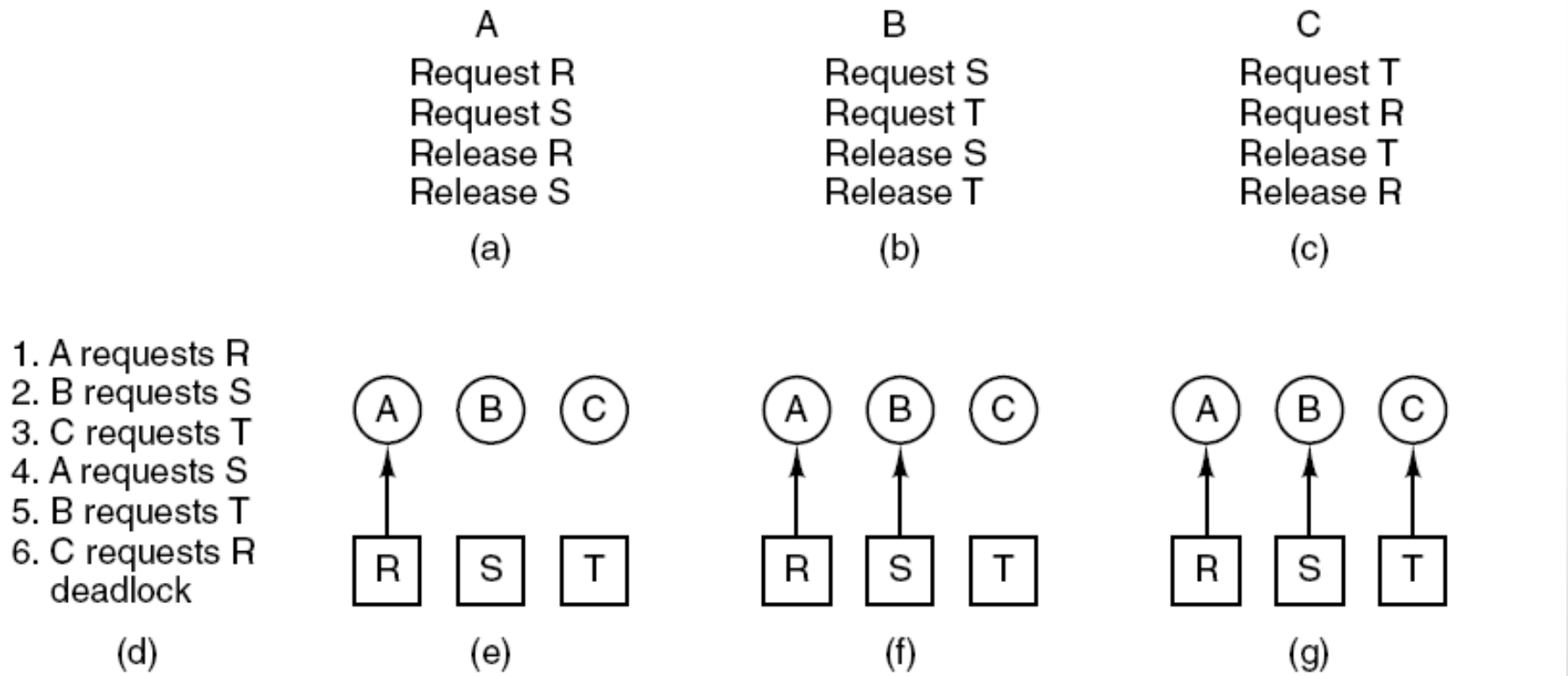
Kilitlenme Modeli

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

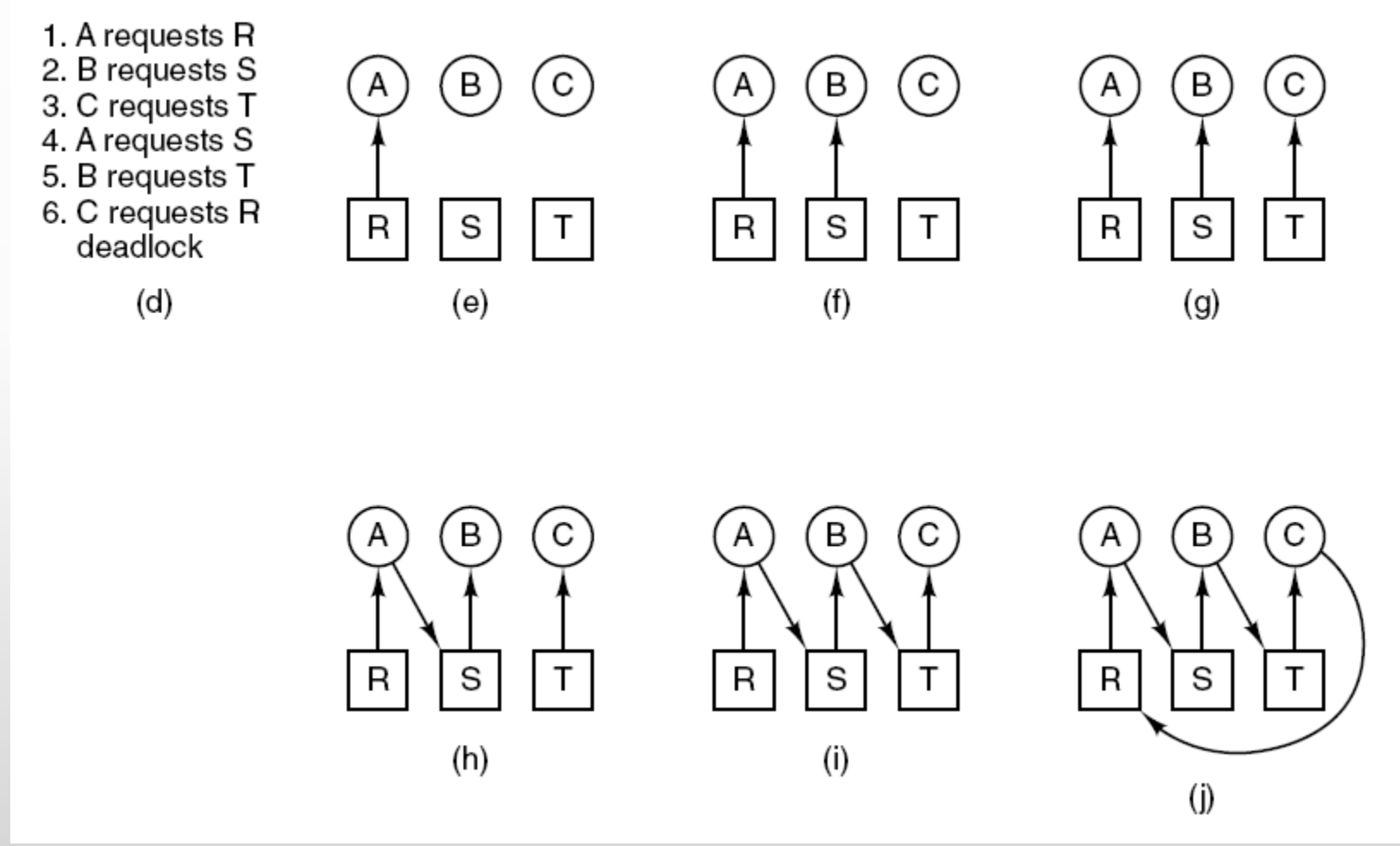


Kilitlenme Nasıl Oluşur?

- .

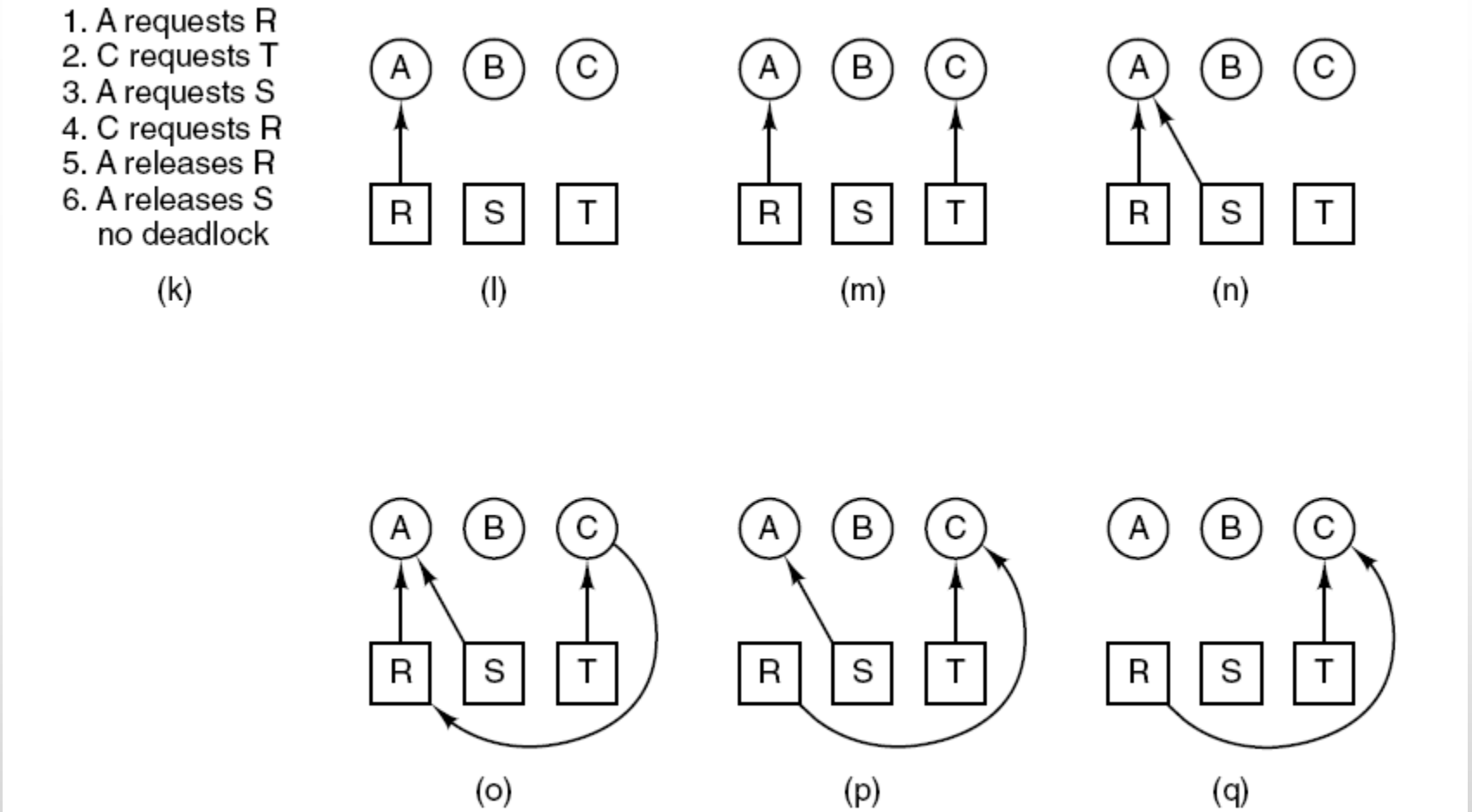


Kilitlenme Nasıl Oluşur?



Kilitlenme Nasıl Önlenebilir?

- .



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.

Ostrich Algoritması

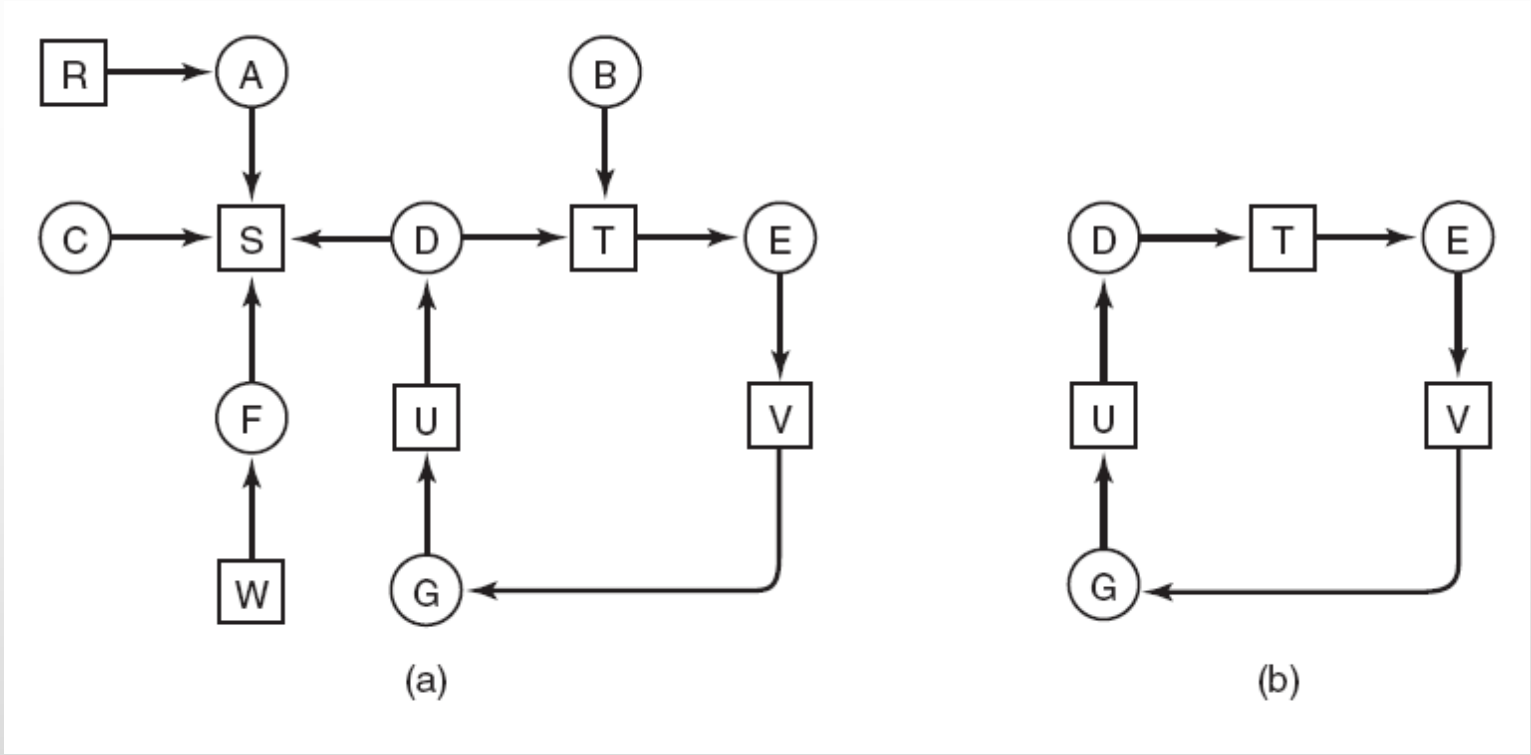
- Devekuşu algoritması, kilitlenmelerden kaçınma ve kilitlenmeleri tespit etme arasında bir denge kurmayı amaçlar
- Sistemin kilitlenmeleri geçici olarak yok saymasına ve istekleri işlemeye devam etmesine olanak tanır. Bir kilitlenme meydana gelirse sistem, pasif bir moda girer.
  - Normal modda sistem, kilitlenme olup olmadığını kontrol etmeden istekleri her zamanki gibi işler.
  - Pasif modda sistem, kilitlenmeleri periyodik olarak kontrol eder. Bir kilitlenme algılanırsa, sistem kaynakları serbest bıraktıktan sonra normal moda döner.
  - Pasif modda, kaynaklar bir süreci sonlandırarak veya bir kaynağı bir süreçten önceden alarak serbest bırakılabilir.

Kilitlenme Tespiti

- Her süreci ve kaynağı temsil etmek için çizgede bir köşe oluştur.
- Süreç ile kaynak arasındaki ilişkiyi temsil edecek şekilde çizgeye bir kenar ekle.
- Çizgede döngüleri bulmak için Derinlik Öncelikli Arama (DFS) gibi algoritmaları kullan. Döngü varsa, kilitlenme olduğu anlamına gelir.
- Her işleme atanan kaynakların bilgisini bir veri yapısında (dizi) sakla.
- Süreçler kaynak talep ettikçe ve serbest bıraktıkça kaynak tahsisi çizgesini güncelle.
- Kilitlenme algılanırsa, kilitlenmeyle ilgili süreçlerinden birini iptal et, veya süreçler tarafından tutulan kaynakların bir kısmını boşalt.

Kilitlenme Tespiti – Her Türden Bir Kaynak

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



Kilitlenme Tespit Algoritması

1. Grafikteki her bir N düğümü için, başlangıç düğümü N olacak şekilde aşağıdaki beş adımı gerçekleştirin.

2. L'yi boş liste olarak ilklendir, tüm okları işaretlenmemiş olarak ata.

3. 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

4. Verilen düğümden giden işaretlenmemiş bir ok olup olmadığına bak. Varsa, 5. adıma git; yoksa, 6. adıma git.

5. Rastgele işaretlenmemiş bir ok seç ve işaretle. Ardından onu yeni geçerli düğüme kadar takip et ve 3. adıma git.

6. 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
$(E_1, E_2, E_3, \ldots, E_m)$

Current allocation matrix

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Row n is current allocation to process n

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

Request matrix

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$$

Row 2 is what process 2 needs

Kilitlenme Tespit Algoritması

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

Kilitlenme Tespiti İçin Bir Örnek

- .

Tape drives Plotters Scanners CD Roms

E = (4 2 3 1)

Tape drives Plotters Scanners CD Roms

A = (2 1 0 0)

Current allocation matrix

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

Request matrix

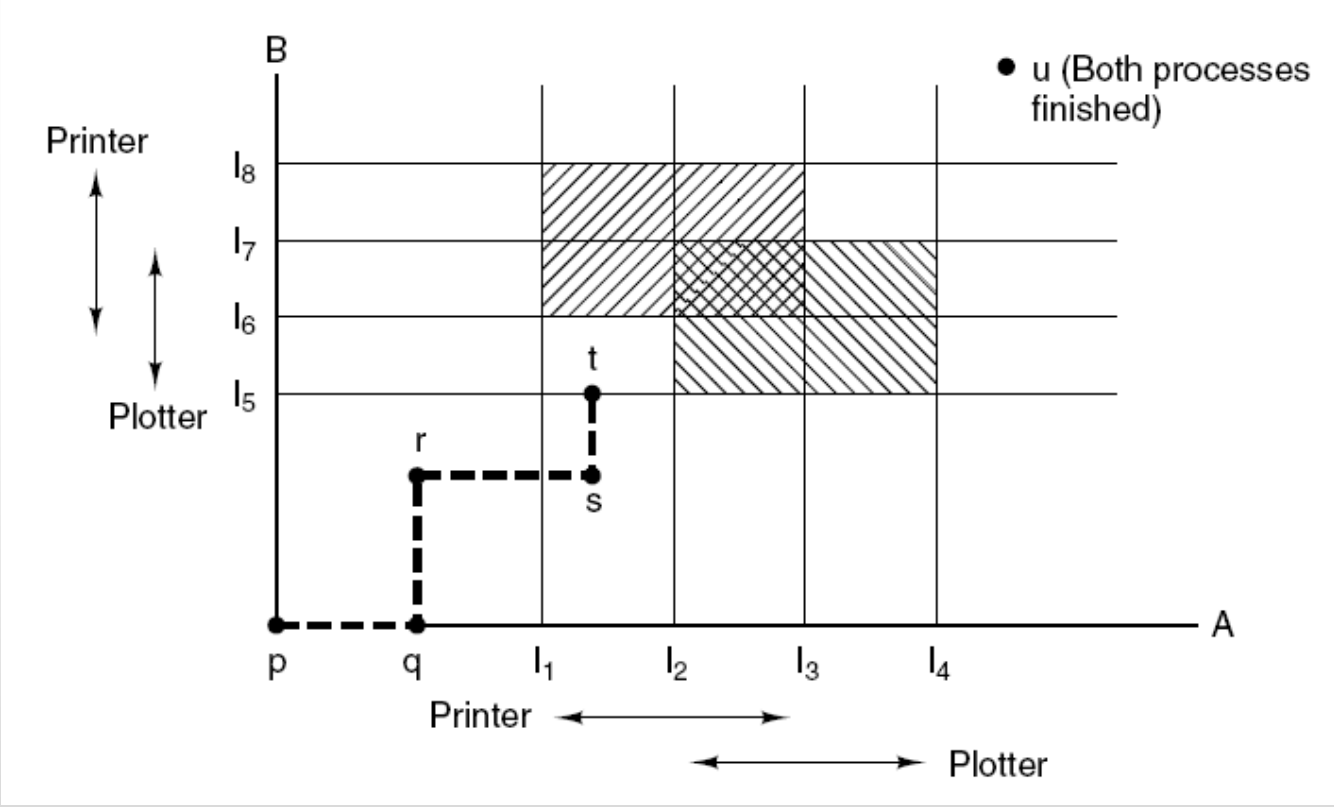
$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Kilitlenmeden Kurtarma (recovery)

- Önleme (prevention): süreçlerin kaynakları talep etmesini kısıtlar
- Tespit (detection): kilitlenmenin ne zaman meydana geldiğini tespit etmek için sistemi izler
- Geri alma (rollback): önceki güvenli duruma geri dönmeyi ve kilitlenmeye neden olan işlemi yeniden başlatmayı içerir
- Sonlandırma (termination): öldürme (killing) süreçlerini içerir
- Kaynak önalımı (preemption): birden fazla kaynağı tutan bir süreçten kaynakları alıp başka bir sürece vermeyi içerir.
- Kaynak tahsisi ayarlaması: bir sürece tahsis edilen kaynakların sayısının dinamik olarak ayarlanmasını içerir.

Kilitlenmeden Kaçınma

- İki süreç kaynağı eğrisi (trajectory)



Güvenli ve Güvensiz Durumlar

- (a)'daki durumun güvenli olduğunun gösterimi, 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)'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

- Mevcut kaynakları ve her bir sürecin maksimum talebini takip eder.
- Süreçler belirli miktarda kaynak ister, eğer istek sistemi güvenli olmayan bir durumda bırakmıyorsa, kaynaklar verilir.
- Süreç talebi, güvenli olmayan bir duruma yol açıyorsa, sistem durumu tekrar güvenli hale gelene kadar ertelenir.
- Tahsis ve kaynak ihtiyacını takip etmek için bir matris kullanır.
- Kilitlenmeye yol açmadan tamamlayabilen bir dizi süreç varsa, durum güvenli kabul edilir.
- Algoritma, tüm süreçler tamamlandığında veya bir 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 gereken her kaynağın maksimum miktarı atanır.
- Algoritma, şu anda her sürece tahsis edilen kaynakları ve mevcut kaynakları izler.
- Kaynak talep eden bir sürece, yalnızca yeterli miktarda kullanılabilir olması ve tahsisin bir kilitlenmeye neden olmuyorsa kaynak verilmesini sağlar.
- Algoritmanın kaynakları bir sürece ayırmasını yeterli kaynaklar sağlanana kadar ertelemesi gerekebilir.
- Kaynak tahsisini izlemek ve potansiyel kilitlenmeleri kontrol için sürekli arka planda 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

- Karşılanmayan kaynağının tamamı ≤ 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 işaretle, 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 tekrarla.

Kilitlenme Önleme

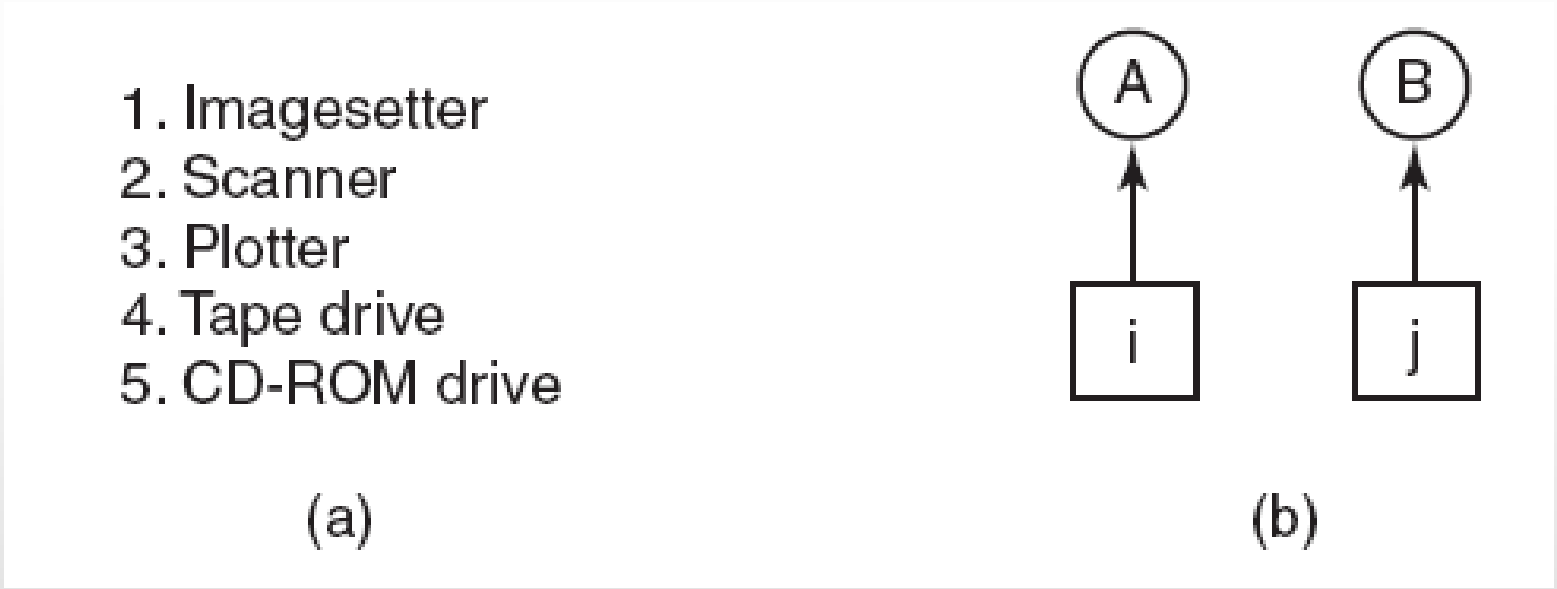
- Karşılıklı dışlama koşuluna saldırmak (mutual exclusion)
  - Her seferinde yalnızca bir süreç kaynak isteyebilir ve kaynak tahsis edilebilir.
  - Kaynakları sıralayarak veya bir kaynağın birden çok örneğine (instance) izin vererek yapılabilir.
- Tut ve bekle durumuna saldırmak (hold and wait)
  - Süreçlerin başkalarını beklerken kaynakları tutmasını önle.
  - Bir sürecin yürütmeye başlamadan önce ihtiyaç duyduğu tüm kaynakları talep etmesi istenerek yapılabilir.

Kilitlenme Önleme

- Ön alım yok koşuluna saldırmak (no preemption)
  - Kaynakların daha yüksek öncelikli bir süreç tarafından önceden kullanılmasından kaçın.
  - Bu, bir işlemin bitene kadar kaynakları tutmasına izin vererek veya bir işlem bloke olursa kaynakları serbest bırakarak yapılabilir.
- Döngüsel bekleme koşuluna saldırmak (circular wait)
  - Kesin bir kaynak tahsisi sıralaması tanımlayarak döngüsel beklemeyi önler.
  - Bu, kaynakları numaralandırarak veya bir kaynak hiyerarşisi kullanarak yapılabilir.

Döngüsel Bekleme Koşuluna Saldırmak

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



Kilitlenme Önleme Yaklaşımları

Durum	Yaklaşım
Karşılıklı dışlama	Her şeyi biriktir
Tut ve bekle	Başlangıçta tüm kaynakları talep edin
Ön alım yok	Kaynakları ortadan kaldırın
Dairesel bekleme	Kaynakları sayısal olarak sıralayın

Diğer Sorunlar

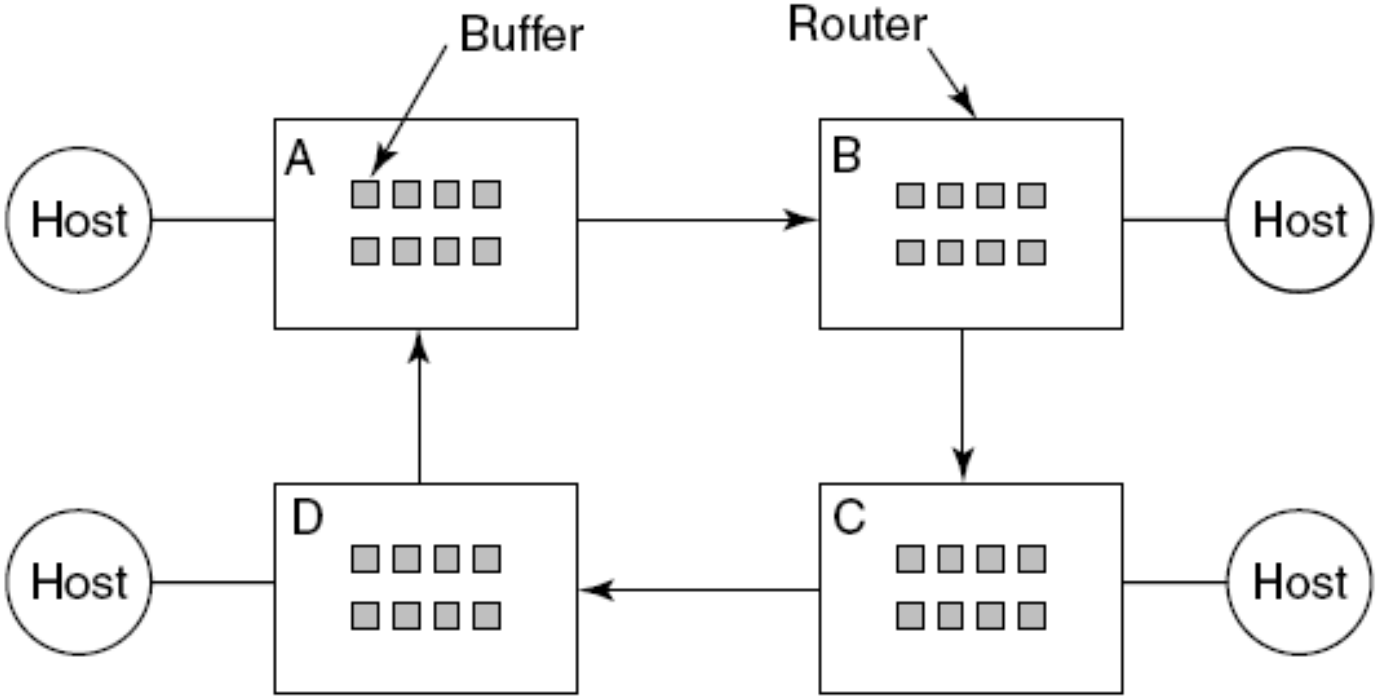
- İki fazlı kilitleme (two phase locking)
  - Yalnızca gerektiğinde kilitlerin alınır ve mümkün olan en kısa sürede serbest bırakılır
  - Veritabanı sistemlerinde kilitlenmelerin olmamasını sağlayan bir protokol.
- İletişim kilitlenmeleri (communication deadlocks)
  - Birden çok süreç veya iş parçacığının iletişim kurduğu ve birbirinden yanıt beklediği sistemlerde oluşur.

Diğer Sorunlar

- Canlı kilit (live lock)
  - İki veya daha fazla sürecin, her biri diğerinin ileriye doğru bir adım atmasını beklediği için ilerleyemediği bir durum.
- Açlık (starvation)
  - Bir sürecin yürütmesini tamamlamak için ihtiyaç duyduğu kaynakları elde etmesi engellendiğinde ortaya çıkar.

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