İşletim Sistemleri

Deadlocks

Emir ÖZTÜRK

İşletim Sistemleri

Deadlocks

- Bilgisayar sistemlerinin bir çok kaynağı aynı anda kullanılamaz
- Yazıcının ortak çalışması
- Dosya sistemine aynı anda iki işlemin yazmaya çalışması
- Birden fazla işlem farklı kaynakları elinde tutarken başka işlemlerin kaynaklarını beklerse bu durumda deadlock oluşur

İşletim Sistemleri

Deadlocks

- Deadlock yalnızca kaynaklar arasında oluşmaz
  - Ağ üzerinde ve haberleşmede deadlock oluşması mümkündür
  - Veritabanında race condition olmasını engellemek adına kilit mekanizması kullanıldığında deadlock oluşabilir

İşletim Sistemleri

Deadlocks

- Kaynaklar iade edilebilir veya iade edilemez olarak ayrılırlar
  - Preemptable
  - Non-preemptable
- Bellek iade edilebilir bir kaynak olarak kabul edilebilir (!)

İşletim Sistemleri

Deadlocks

- Bir yazıcı ve 16 gb bellek olduğu durumda
- Bir işlem belleğin tümünü alıp yazıcıyı da aldıktan sonra yazdırmaya başlar
- İkinci işleme sıra geldiğinde işlem belleği alır fakat yazıcıyı alamaz
- Burada bellek iade edilip yazıcıya sahip işleme tekrar verilebilir
- Böylece yazıcı kullanımı tamamlanıp bu işlem sonlanabilir ve ikinci işleme tekrar sıra gelebilir

İşletim Sistemleri

Deadlocks

- Bir 3D yazıcı için ise durum farklıdır
- Bir yazdırma işlemi başladığında diğer bir işlem yazıcıyı devralırsa çıktı bozulacaktır
- İade edilebilir olup olmama durumu cihaza göre değişir
  - Swap işlemi içermeyen küçük bir sistemde bellek de iade edilebilir değildir

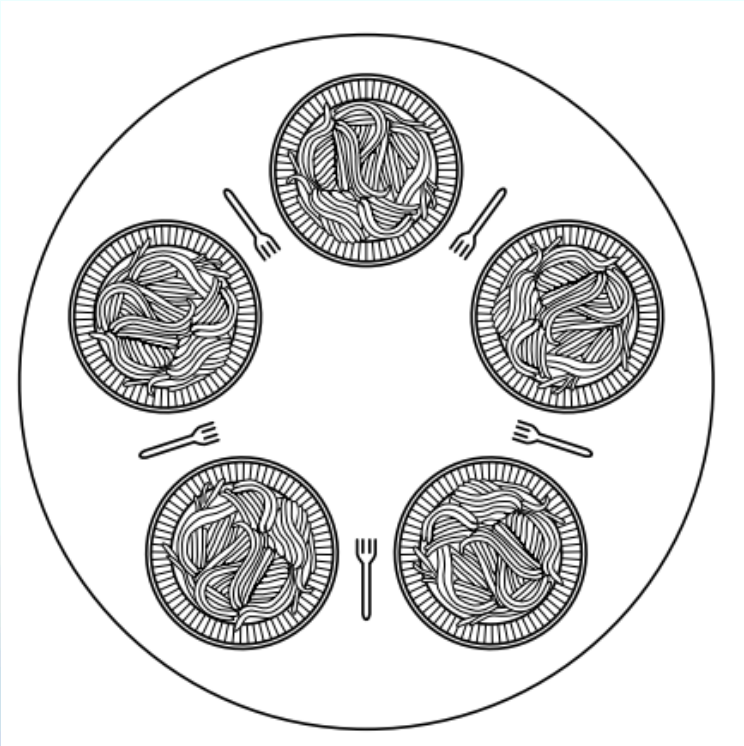
İşletim Sistemleri

Deadlocks

- Deadlock oluşmaması için kaynakların sıralı kullanımı
  - Mutex
  - Semaphore
- Paralelizm?

İşletim Sistemleri

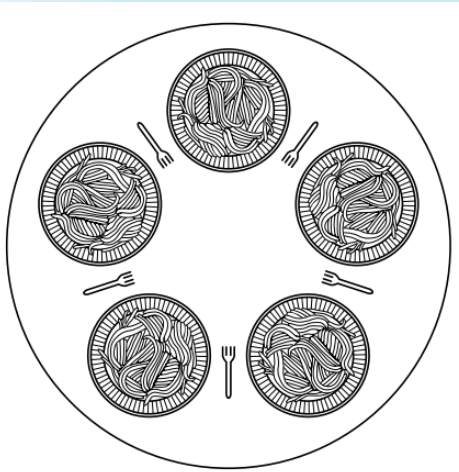
Dining philosopers problem



İşletim Sistemleri

Dining philosopers problem

- Yeme ve düşünme
- Sırayla çatal al ve çatal müsait değilse bekle
  - Herkesin sol çatalı aldığı durumda deadlock
- Sol çatalı aldıktan sonra sağdaki müsait değilse soldaki çatalı da bırakma
  - Her işlem aynı anda sol çatalı alırsa herkes sağ müsait değil deyip bırakır
  - Hiçbir işlem başlamaz
  - Starvation



İşletim Sistemleri

Deadlocks

- Mutex ve semaphore kullanımı
  - Bir felsefeci aynı anda yiyebilir
  - Sistemin 2 filozofa izin verebilmesi gerekli
- Bu problem için çözüm
  - Her filozof ancak iki yanındaki filozof yemiyorsa yiyebilir

İşletim Sistemleri

Deadlocks

- Bir deadlock oluşması için 4 maddenin sağlanması gerekir
- Bir madde bile sağlanmazsa deadlock oluşmaz ya da çözülür
  - Mutex durumu: Her kaynak bir işleme atanmalı ya da serbest olmalı
  - Tutma ve bekleme durumu: Kaynak tutan işlemler yeni kaynak isteyebilir
  - İade edilememezlik durumu: Kaynaklar işlemden zorla alınamaz
  - Döngüsel bekleme durumu: İki veya daha fazla işlemin döngüsel bir şekilde kaynağı beklemesi gerekir

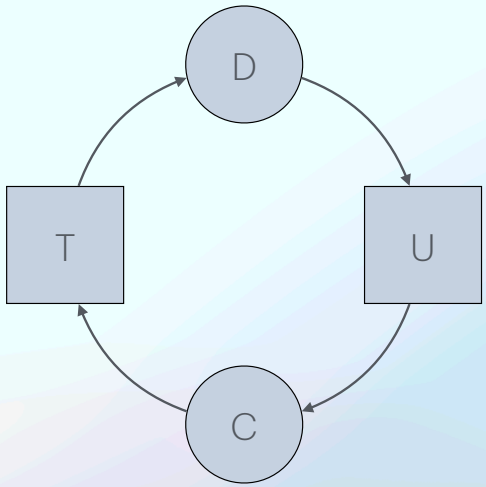
İşletim Sistemleri

Deadlock modelleme

- İşlemler daire ile gösterilir
- Kaynaklar kare ile gösterilir
- Bir işlem kaynağa sahipse ok işleme doğrudur
- Bir işlem kaynağı istiyorsa ok kaynağa doğrudur

İşletim Sistemleri

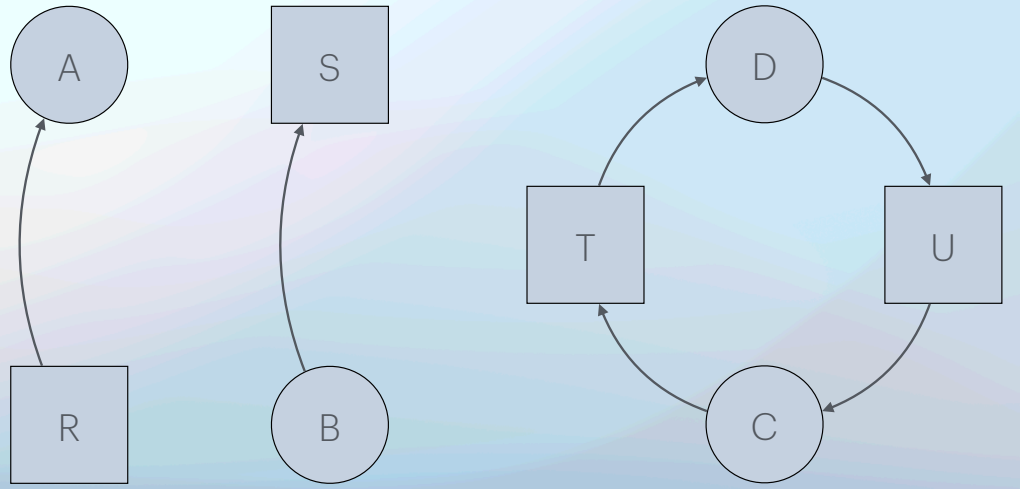
Deadlock modelleme



İşletim Sistemleri

Deadlock modelleme

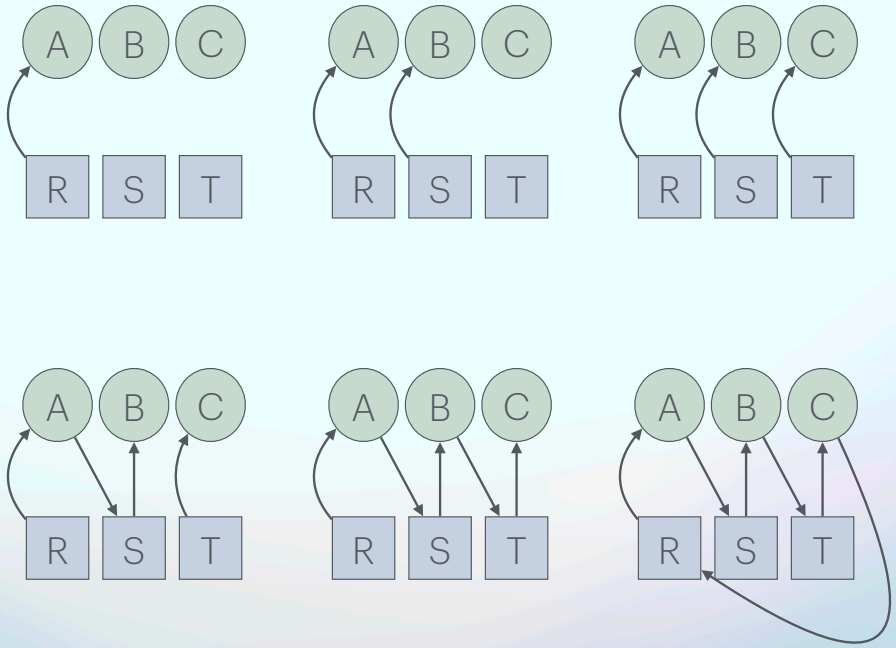
- A, B ve C işlemleri için R, S ve T kaynakları olsun
- İşletim sistemi bloklanmış olmayan istediği işlemi çalıştırabilsin
- Sırayla çalışmada sorun oluşmaz çünkü her işlem bittiğinde kaynakları da bırakmış olur
- Round robin uygulandığında ?



İşletim Sistemleri

Deadlocks

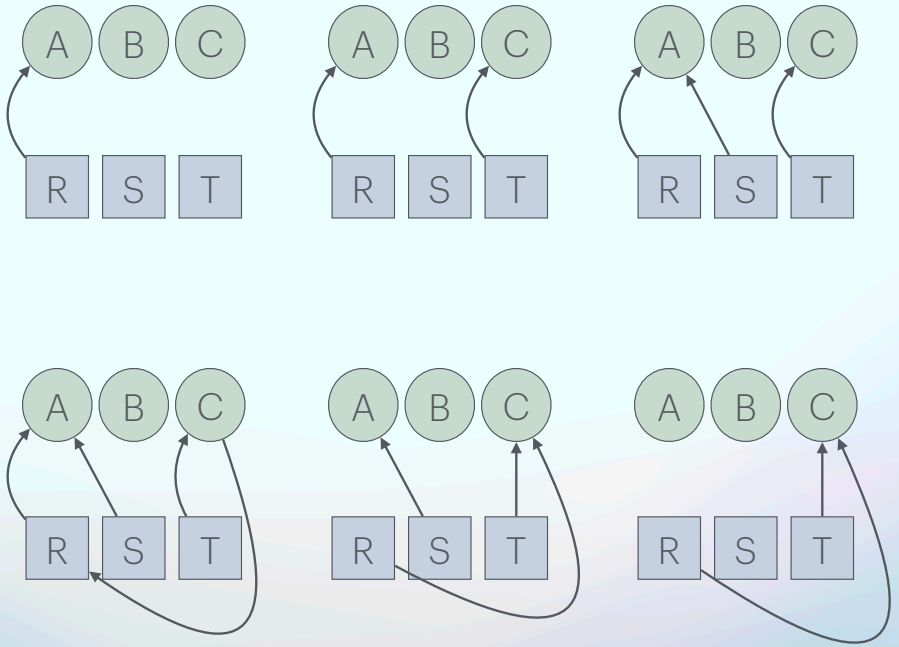
A R'yi ister
B S'yi ister
C T'yi ister
A S'yi ister
B T'yi ister
C R'yi ister
Deadlock



İşletim Sistemleri

Deadlocks

A R'yi ister
C T'yi ister
A S'yi ister
C R'yi ister
A R'yi bırakır
A S'yi bırakır
Deadlock yok



İşletim Sistemleri

Deadlocks

- Deadlock oluşma ihtimali olan bir sistemde her zaman bu ihtimalin gerçekleşme zorunluluğu bulunmamaktadır
- İşletim sistemleri istediği işlemi durdurup devam ettirebilir
  - Deadlock çözülene kadar bu işlemi gerçekleştirebilir
- İşletim sistemi deadlock olacağını tespit edebilirse işlem önceliklerini ayarlayabilir

İşletim Sistemleri

Deadlock çözümleri

- Deadlock durumunu çözmenin dört farklı yolu bulunur
  - Problemin görmezden gelinmesi
  - Problemin çalışma anında tespit edilip çözüm üretilmesi
  - Kaynak tahsisinin dikkatli bir şekilde yapılarak engellenmeye çalışılması
  - Önceki deadlock oluşma şartlarından birinin iptal edilebilmesi ile deadlock'un engellenmesi

İşletim Sistemleri

Deadlocks - Ostrich algoritması

- Devekuşu
- Matematikçiler problemin çözümü olmadığı için karşı
- Mühendislik tarafında problem olasılığı düşükse göz ardı edilebilir
- Daha önceki pathfinder örneği
- Çok uzun vadede oluşabilecek bir sıkıntı için performans kaybının istenmemesi

İşletim Sistemleri

Deadlock tespiti ve kurtarma

- Her kaynaktan bir adet olduğu durum
- A-G arası işlemler
- R-W arası kaynaklar
- Örnek?

İşletim Sistemleri

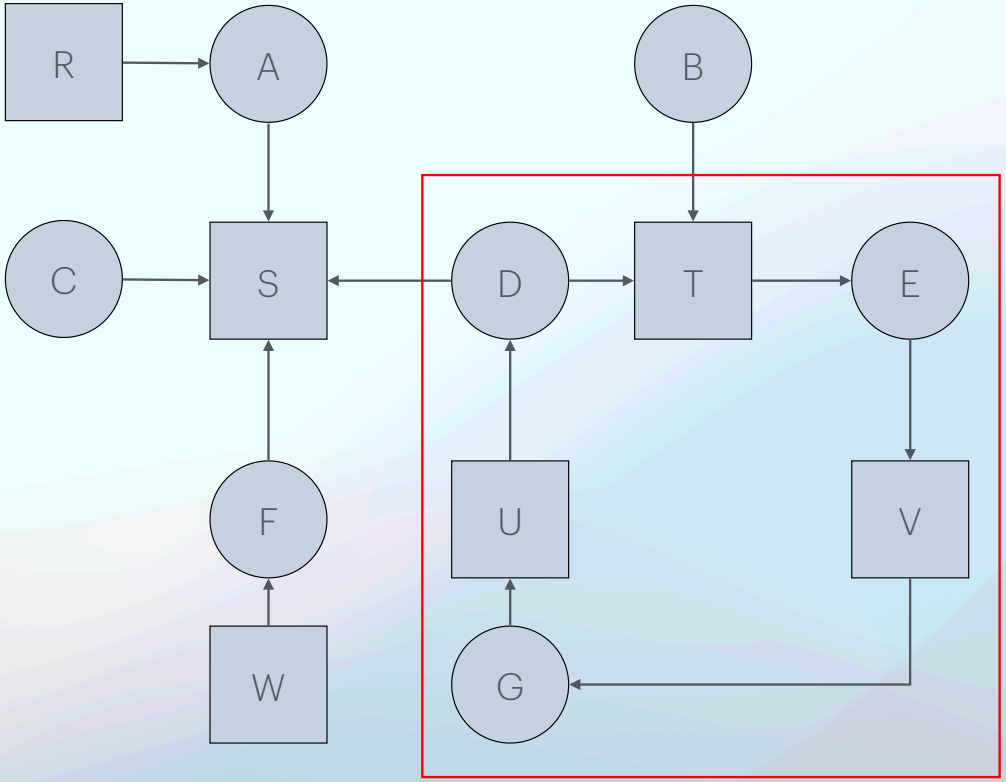
Deadlock tespiti ve kurtarma

- A R'ye sahip S'yi istiyor
- B hiçbir şeye sahip değil T'yi istiyor
- C hiçbir şeye sahip değil S'i istiyor
- D U'ya sahip S ve T'yi istiyor
- E T'ye sahip V'yi istiyor
- F W'ya sahip S'i istiyor
- G V'ye sahip U'yu istiyor

İşletim Sistemleri

Deadlock tespiti ve kurtarma

- A R'ye sahip S'yi istiyor
- B hiçbir şeye sahip değil T'yi istiyor
- C hiçbir şeye sahip değil S'i istiyor
- D U'ya sahip S ve T'yi istiyor
- E T'ye sahip V'yi istiyor
- F W'ya sahip S'i istiyor
- G V'ye sahip U'yu istiyor



İşletim Sistemleri

Deadlock tespiti ve kurtarma

- Graftan görsel olarak deadlock olma ihtimalini çıkartmak kolay olsa da bir algoritmaya ihtiyaç vardır

1. Grafiğin N adet düğümü için aşağıdaki beş adımı başlangıç düğümü olarak kullanarak gerçekleştir.

2. L'yi boş bir liste olarak başlat ve tüm bağları işaretsiz olarak belirt.

3. Mevcut düğümü L'nin sonuna ekle ve düğümün şu anda L'de iki kez bulunup bulunmadığını kontrol et. Eğer öyleyse, grafik bir döngü içerir ve algoritma sona erer.

4. Verilen düğümden çıkış yapan bir yere bağlanmayan bağlantıların olup olmadığını kontrol et.

   Eğer varsa, adıma 5'e git

   Yoksa, adım 6'ya git.

5. Rastgele bir işaretsiz çıkış bağını seç ve işaretle. Sonra yeni mevcut düğüme git ve adım 3'e git.

6. Eğer bu düğüm başlangıç düğümü ise, grafik herhangi bir döngü içermez ve algoritma sona erer. Aksi takdirde çıkmaz noktaya ulaşılır. Bu noktayı kaldır ve bir önceki düğüme geri dön ve adım 3'e git.

İşletim Sistemleri

Deadlock tespiti ve kurtarma

- Her kaynaktan birden fazla bulunduğu durum
- Bir kaynak vektörü oluşturulabilir
- [E1, E2, E3]
- Her eleman bir türden kaynağın ne kadar elde bulunduğunu tutar
- Mevcut durumda tahsis edilen kaynaklar için C matrisi
- Mevcut durumda istenilen kaynaklar için R matrisi

İşletim Sistemleri

Deadlock tespiti ve kurtarma

Bulunan kaynaklar
($E_1, E_2, E_3, \ldots, E_m$)
Tahsis matrisi

$$\begin{bmatrix} C11 & C12 & C13 & \ldots & C1m \\ C21 & C22 & C23 & \ldots & C2m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ Cn1 & Cn2 & Cn3 & \ldots & Cnm \end{bmatrix}$$

n'inci işleme yapılan tahsis

Alınabilecek kaynaklar
($A_1, A_2, A_3, \ldots, A_m$)
İstek matrisi

$$\begin{bmatrix} R11 & R12 & R13 & \ldots & R1m \\ R21 & R22 & R23 & \ldots & R2m \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ Rn1 & Rn2 & Rn3 & \ldots & Rnm \end{bmatrix}$$

İşlem 2'nin ihtiyacı olan kaynaklar

İşletim Sistemleri

Deadlock tespiti ve kurtarma

- Bu matrisler incelenerek hangi işlemin çalıştırılabileceği tespit edilebilir

Teyp Yazıcı Tarayıcı Kamera

E = (4 2 3 1)

Tahsis matrisi

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

Teyp Yazıcı Tarayıcı Kamera

A = (2 1 0 0)

İstek matrisi

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

İşletim Sistemleri

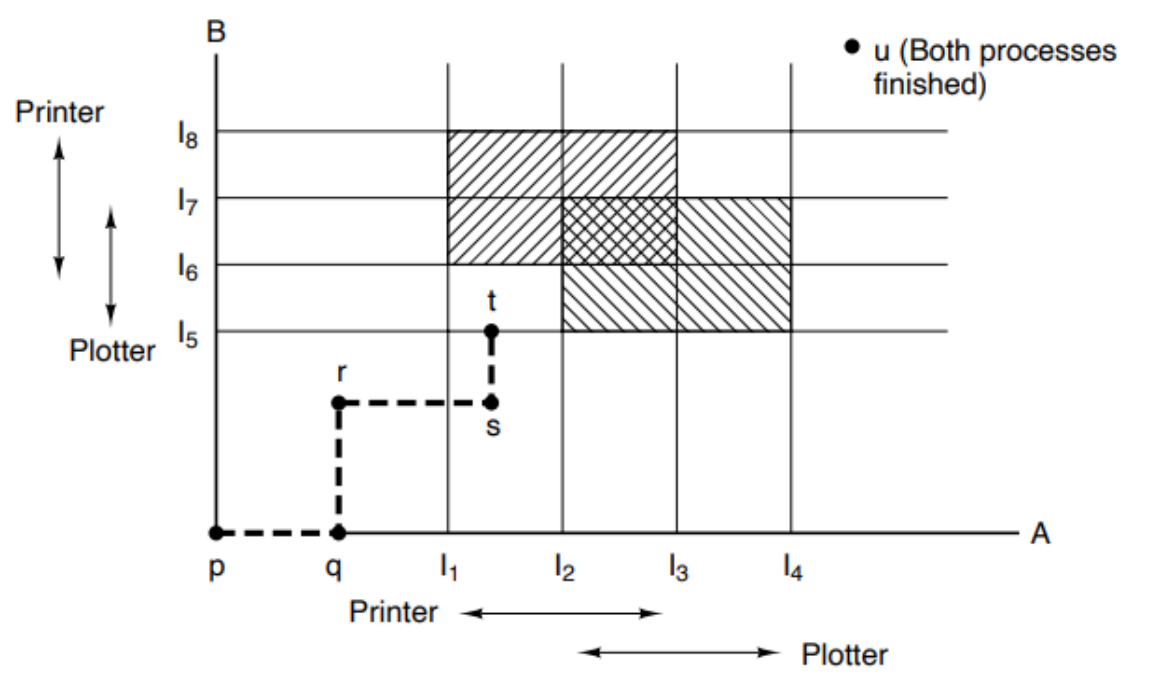
Deadlock tespiti ve kurtarma

- Deadlock durumundan kurtulmak için farklı yollar bulunur
  - İade (preemption)
  - Rollback
    - Checkpoint
  - İşlem sonlandırma (Kill)

İşletim Sistemleri

Deadlock oluşmadan engelleme

- Kaynak rotaları



İşletim Sistemleri

Deadlock oluşmadan engelleme

- Tüm işlemler ihtiyaç duyduğu kaynağı baştan istese bile kilit oluşmayacak bir durum bulunursa güvenli durum olarak adlandırılır
- Aksi takdirde durum güvensizdir

	Var	Max
A	3	9
B	2	4
C	2	7

Boş: 3

	Var	Max
A	3	9
B	4	4
C	2	7

Boş: 1

	Var	Max
A	3	9
B	0	-
C	2	7

Boş: 5

	Var	Max
A	3	9
B	0	-
C	7	7

Boş: 0

	Var	Max
A	3	9
B	0	-
C	0	-

Boş: 7

10 kaynak için

	Var	Max
A	3	9
B	2	4
C	2	7

Boş: 3

	Var	Max
A	4	9
B	2	4
C	2	7

Boş: 2

	Var	Max
A	4	9
B	4	4
C	2	7

Boş: 0

	Var	Max
A	4	9
B	-	-
C	2	7

Boş: 4

İşletim Sistemleri

Deadlock oluşmadan engelleme

- Mutex durumu engellenirse
  - Veri bütünlüğü kaybolur
- Tutma ve bekleme durumu engellenirse
  - İhtiyaç duyduğu her kaynak işleme başta verilebilir
  - Her işlem ne kadar kaynak ihtiyacı duyacağını baştan bilemeyebilir
  - Her kaynağı istese de kullanmadığı zamanlar kayıp olur

İşletim Sistemleri

Deadlock oluşmadan engelleme

- İade edememe durumunu engelleme
  - Yazıcı gibi kaynakların işlem değiştiğinde işi yarım bırakma ihtimali var
  - Bu durumda yazıcı işleri sanal bir klasörde birikebilir
  - Diskin dolması deadlock oluştursa da çok daha az olasıdır
  - Her kaynak iade edilebilir olmayabilir
    - Veritabanı kayıt güncellemesi

İşletim Sistemleri

Deadlock oluşmadan engelleme

- Döngüsel bekleme durumunu engelleme
  - İşlemlerin kaynak isteme sırasına bir kural koyulabilir
  - Önce yazıcı sonra kayıt cihazı istenebilir ama önce kayıt cihazı sonra yazıcı istenemez
  - Graf böylece hiçbir zaman döngü içermeyeceğinden deadlock oluşmaz
  - Sıralama her işleme optimum derecede uygunluk sağlamayacaktır

İşletim Sistemleri

Two Phase locking

- Deadlock tespiti ve engellemesi için kullanılabilir
- Birinci denemede kaynak alınıyormuş gibi yapılıp deadlock oluşup oluşmadığı kontrol edilir
- Deadlock oluşuyorsa kaynak geri verilir

İşletim Sistemleri

Haberleşmede deadlock

- Kaynakların yönetimi uzak bir cihazda mümkün değil
- Başka bir çözüm bulunmalı
- Timeout

İşletim Sistemleri

Livelock

- Bir kaynağı tutarken diğer kaynağı bekleyen bir işleme öncelik tanınabilir
- Diğer işlem de aynı anda aynı işlemi gerçekleştirirse
- İki işlem de tüm kaynakları bırakır ve birbirini bekler
- Livelock