### İşletim Sistemlerine Giriş

Ölümcül Kilitlenme (Deadlock)

### 2- Her tipten birden fazla kaynak için ölümcül kilitlenme tespiti

Matris temelli bir algoritma kullanılır.

Süreçler P1-PN olsun.

Elimizdeki kaynakların sınıflarını Ei (1<=i<=m) şeklinde isimlendirelim.

E var olan kaynak vektörüdür. Her kaynaktan var olan örnek sayısını verir. Örneğin E1 disket sınıfı olsun.

E1=2 elimizde 2 adet disket sürücüsü olduğunu göstersin.

Belirli bir anda bu kaynaklardan süreçlere tahsis edilmiş veya kullanılabilir olabilir.

A vektörü kullanılabilir olan kaynakları göstersin. Ai i. sınıf kaynaktan kullanılabilir kaç tane örnek olduğunu göstersin.

Eğer A1=0 ise, disket sürücüsünden kullanılabilir yok demektir.

Tüm disket sürücüleri şu anda kullanılıyordur.

Algoritmada iki adet matris kullanılır.

**C** = mevcut anda süreçlere tahsis edilmiş kaynaklar matrisidir.

R = süreçlerin istedikleri kaynakları ve sayılarını gösteren matristir.

C nin her satırı bir sürece karşılık gelir. Bu satırın sütunları, sürecin sahip olduğu kaynakları göstermektedir.

Cij = j kaynağından i sürecinde kaç adet var.

Rij = i sürecinin j kaynağından kaç adet istediğini gösterir.

#### Varolan kaynaklar

Resources in existence  $(E_1, E_2, E_3, ..., E_m)$ 

#### Mevcut anda kaynak tahsis matrisi Current allocation matrix

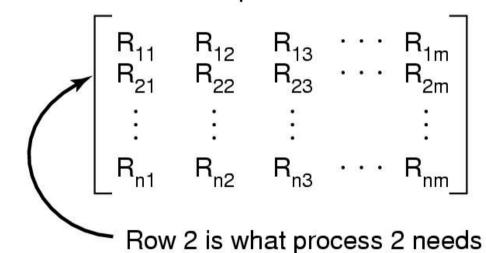
$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$$

Row n is current allocation to process n

#### Kullanılabilir kaynaklar

Resources available (A<sub>1</sub>, A<sub>2</sub>, A<sub>3</sub>, ..., A<sub>m</sub>)

#### Süreçlerin kaynak isteklerini gösteren matris Request matrix



Her sürece başlangıçta işaretlenmemiş denilir.

Algoritma ilerledikçe süreçler işaretlenir.

İşaretlenmiş süreç çalışmasını sonuna kadar tamamlayabilecek anlamına elmektedir.

Algoritma bittiğinde eğer işaretmenmemiş süreç varsa, bu süreçler ölümcül kilitlenmiştir.

### Algoritma:

- 1-İşaretli olmayan süreç ara. Bu sürecin R matrisindeki tüm istekleri A dan küçük veya eşit olmalıdır.
- 2- Bu şekilde bir süreç var ise C deki bu sürece karşılık gelen satırın değerlerini A ya ekle. Bu süreci işaretle ve 1. adıma git.
- 3-Böyle bir süreç yoksa algoritma sonlandırılır.
- Algoritma bittiğinde işaretli olmayan süreçler ölümcül kilitlidir.

#### Örnek:

disket yazıcı tarayıcı cdrom
$$E = (4 \quad 2 \quad 3 \quad 1)$$
Toplam kaynak sayıları

#### Mevcut tahsis edilmiş kaynaklar

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

#### Kullanılabilir kaynaklar

$$A = (2 \ 1 \ 0 \ 0)$$

#### Süreçlerin kaynak istekleri

Örnek: 
$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \\ disket \ yazıcı \ tarayıcı \ cdrom \end{pmatrix}$$
  $A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$   $C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$   $R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$ 

Bu örnekte 3 süreç ve 4 kaynak bulunmaktadır.

- 1. sürecin: 1 tarayıcısı
- 2. sürecin: 2 disket sürücüsü ve 1 cdromu vardır.
- 3. sürecin: 1 yazıcısı ve 2 tarayısı

R matrisinden istekleri karşılanabilen bir süreç var mı yok mu incelenir.

- 3. sürecin istekleri karşılanır. Bu süreç seçilir, işaretlenir ve sahip olduğu kaynaklar iade edilir.
- A=(2 2 0) olur. Bu adımda 2. süreç çalıştırılır.

 $A=(4 \ 2 \ 2 \ 1)$ 

Kalan süreç de çalıştırılır. Sistemde ölümcül kilitlenme yoktur.

### Algoritma:

- 1-İşaretli olmayan süreç ara. Bu sürecin R matrisindeki tüm istekleri A dan küçük veya eşit olmalıdır.
- 2- Bu şekilde bir süreç var ise C deki bu sürece karşılık gelen satırın değerlerini A ya ekle. Bu süreci işaretle ve 1. adıma git.
- 3-Böyle bir süreç yoksa algoritma sonlandırılır.
- Algoritma bittiğinde işaretli olmayan süreçler ölümcül kilitlidir.

### Ölümcül Kilitlenme (Deadlock) Onarımı

Algoritma bir ölümcül kilitlenme tespit etmiş ise ne yapılmalıdır?

### 1. Keserek onarma (Recovery through preemption)

Bazı durumlarda kaynağı geçici olarak sahibinden alıp başkasına vermek mümkün olabilir. Bu tip durumlarda sisteme el ile müdahele yapılması genellikle gerekmektedir.

Örneğin; Bir yazıcı başkasına kesilerek verilebilir. Yazıcı askıya alınır. Yazdırmış olduğu kağıtlar bir yere konulur. Yazıcı başka bir sürece verilir. Bu sürecin çalışması bittiğinde eski sürece geri dönülür.

Bu şekilde onarım oldukça zordur.

### Ölümcül Kilitlenme (Deadlock) Onarımı

2. Geri alarak onarma (Recovery through rollback) Sistem tasarımcıları ve işleticileri sistemde ölümcül kilitlenmelerin olası olduğunu biliyorsa, süreçler için periyodik olarak kontrol noktaları oluştur. Kontrol noktası sürecin mevcut durumunun bir dosya ya yazılmasıdır. Bu şekilde süreç daha sonra bu noktadan çalışmasına devam edebilir. Kontrol noktası sürecin bellek görünümünü ayrıca kaynaklarının durumunuda içermelidir. Her kontrol dosyası farklı dosyaya yazılır. Ölümcül kilitlenme halinde hangi kaynaklara ihtiyaç olduğu belirlenir. Bu kaynaklar sahibinden alınır ve ihtiyaç duyana verilir. Kaynakların işleri bittiğinde alındıkları süreç kontrol noktasından itibaren tekrar çalıştırılmaya başlanır.

## Ölümcül Kilitlenme (Deadlock) Onarımı

### 2. Öldürerek onarma (Recovery through killing process)

Kaba fakat en kolay yol bir veya birden fazla sürecin öldürülmesidir.

Bir döngü deki süreci öldürerek ölümcül kilitlenme çözülebilir.

Döngü dışındaki bir süreci öldürmek de ölümcül kilitlenmeyi çözebilir.

En iyisi tekrar çalıştırılması problem oluşturmayacak olan sürecin öldürülmesidir.

# Ölümcül Kilitlenmeden Kaçınma (Deadlock avoidance)

Ölümcül hata tespitinde, sürecin ihtiyaç duyduğu tüm kaynakları R matrisinde verdiğini varsaydık.

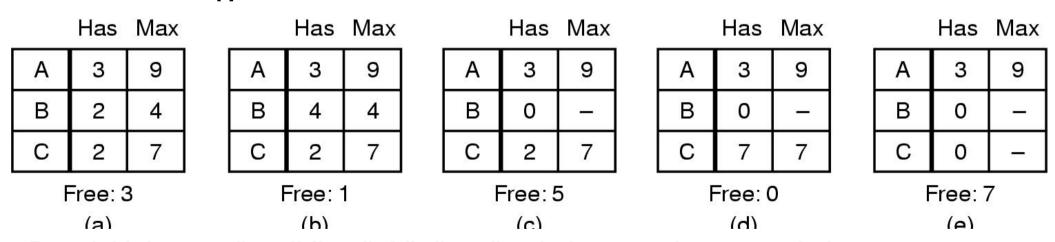
Bir çok sistemde kaynaklar belirli bir çalışma anında istenirler.

Sistem istenilen kaynağın verilmesinin güvenli olup olmadığına karar vermelidir.

### Güvenli ve Güvensiz Durum

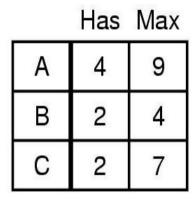
Herhangi bir zaman anında sistemin durumu E,A vektörleri ve C,R matrisleri ile gösterilir. Eğer A durumu ölümcül kilitli değilse sistem güvenlidir.

Örneğin tek bir kaynak için durumu inceleyelim. Elimizde bu kaynaktan 10 adet olsun. Aşağıda A,B,C süreçlerinin durumları gösterilmektedir.



Buradaki durum güvenlidir, çünkü tüm süreçlerin tamamlanmasına imkan veren seçim sırası bulunur. İşletim Sistemlerine Giriş - Ders07

### Güvenli ve Güvensiz Durum



Free: 2

V <del></del>	Has	Max
Α	4	9
В	4	4
С	2	7
S. <del>T.</del>		

Free: 0

	паѕ	iviax
Α	4	9
В	1	_
С	2	7

Has Max

Free: 4

Buradaki durum güvensiz bir durumdur.

### Tek kaynak için Banker Algoritması

Örneğin, kaynağımız 10 adet disket sürücüsü olsun ve elimizde 4 adet süreç bulunsun(A,B,C,D süreçleri).

Algoritma süreçlerden her istek geldiğinde, kaynağın verilmesinin güvenli olup olmadığını kontrol eder.

Güvenli ise kaynak tahsis edilir, değilse bu istek ertelenir.

	Has	Max
Α	0	6
В	0	5
С	0	4
D	0	7

Free: 10

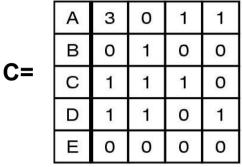
	Has	Max
Α	1	6
В	1	5
С	2	4
D	4	7

Free: 2

	F (200	Has	Max
	Α	1	6
	В	2	5
I	O	2	4
	D	4	7

Free: 1

### Birden fazla kaynak için Banker Algoritması



0 0 В 0 R= 0 D 0 0 Ε 2 0

E = (6342)

P = (5322)

A = (1020)

Sistemde var olan kaynaklar

Tahsis edilmiş kaynaklar

Kullanılabilir kaynaklar

Tahsis edilmis kaynaklar

İhtiyaç duyulan kaynaklar

- 1. R matrisinden ihtiyaç duyulan kaynaklar aranır. Eğer bu satırlarda A vektöründen daha küçük bir satır yoksa sistem öümcül kilitlidir.
- 2. Bu sürecin tüm istekleri aynı anda karşılansın. Tüm sahip oldukları A vektörüne eklenir.
- 3. 1-2 adımlarını tüm süreçler sonlandı olarak işaretleninceye kadar tekrar edelim.

### İşletim Sistemlerine Giriş

Ölümcül Kilitlenme (Deadlock)