

OPERATING SYSTEMS : İşletim Sistemleri Cemal

Bir işletim sistemi, sistemini <sup>3. sınıf 1. Dönem max 5</sup> zamanla <sup>köse</sup> almalı.  
 (extended machine) **BİLGİSAYAŁ**  
 Genişletilmiş matine yöneticisi: Kullanıcıya basit yapılar sunar  
 Yolsa kullanıcı bir dosyayı açmak için tıklanmadan öte seyler yapmamalı. Sonra bir ortam sağlıyor.

(Resource manager)  
 → Görev yöneticisi Program ne kadar zaman alır?  
 Ne kadar boyutu vardır?  
 Bunları işletim sistemi en uygun şekilde tesis edecek.  
 Etkin ve güvenli kaynak yönetimi sağlanmak için -

Bilgisayar Sistem Aracıları : Components

Application Software → Birim geliştiriniz programlar  
 System Software → İşletim Sistemi Emir yorumlayıcı  
 Physical Hardware → Editor, Compiler  
 İşlemci, Matne Dil, Fiziksel Aracılar  
 Bilgisayar mimarisi

- Uygulama yazılımları için sistem yazılımları böle ortaklığı sağlıyor.
- OS, uygulama ve hardware arası bir interface oluşturur.
- \* İşletim sistemi üzerinde hesaplamalar kurulur ve koşullar. Ancak işletim sisteminin bilirseği güvenli ve etkin program yaratılır.

İşletim Sistemi Nenidir? Büyüük karmaşık bir program

- interface oluşturmak. kullanıcı - donanım arası
- hesaplamaların kaynaklarının yönetimi (güvenli ve etkin)

OS olmasaydı hardware erimeyi daha fazla bilmeliydik. programın bir işlem için rafin kodlarını.

OS sisteminin neresinde?

Biz sadece sistem çağrısı yapıyoruz - (örn. dosya aa dýyoruz)  
 İşletim sistemi arada  
 CPU

OS Tarihçesi!

Bir kaset gibi tümünü toplayorsun sırasıyla çalıştırıyorsun.

Batch system: Yığın sistemi. Programlar arkasına arkaya biribirip çalptırılır.

- 80'li yıllarda delikli kartlar vardı. Manyetik bir sistem
- \* 5. kuşakta işletim sistemi ağ ile bütünleşmiş oldu. Bu hiç öngörülmemiştir.

Spooling

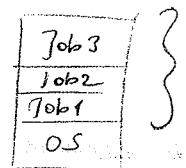
1. kusak: vacuum türpleri, plug boards 1945 - 1955  
Güç tüketimleri fazla  
Biraz ısınması gerekıyor. Anantlarla  
Yörge herası sin cos tablo herası

2. kusak: Transistor, batch system 1955 - 1965.  
Hızlı gelişir. Küçük hacim, yüksek performans, ticari olarık.

3. kusak: entegre devreler çoklu programlar. 1965 - 1980  
online spooling minicomputer

4. kusak: PC (Bellekleri 128 KB idi) 1980 → ~~1980~~  
Güç tüketimini oldukça fazla. İlk işletim sistemleri CP/M, MS Dos, UNIX.  
Network → Programları uzaktan çalıştırma, talk.

Paralel sistemlerde simülatörde FORTRAN ve C kullanılır.



multithread

Main frame OS  
Server OS  
Multiprocessor OS

- COBOL ticari.
- FORTRAN bilimsel.

5. kusak: 1990+

- internet ve intranet networking (www).
- client + server modeli.

↓  
Basit bir test  
makinelerinde çalışır  
olabilir.

↓  
Güçlü sistemlerde  
kullanılıyor.

### Önemli Kavitalar:

- OS bize basit ve güçlü bir arayüz sağlar.
  - Yüksek seviyede servisler sağlar. Sistem çağrıları ile servistere erişilebilir.
- API's (Sistem çağrıları, hizmeti) OS' ten servisleri bu yolla ister.

### Concepts:

Kernel: işletim sisteminin ana parçası. Her zaman koşar.

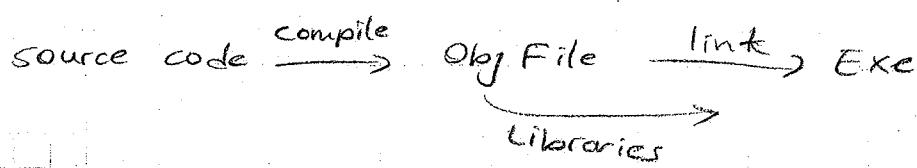
Device Drivers: Programlar I/O cihazlarına basit ve tutarlı (benzer cihazla benzer işlem yapma) erişim.

- Kernel'in parçasıdır.

Program: Static olarak diskte durur.

Process: Programın icra edilen hali.

- Program koşarken kaynaklar kullanır. Sonlanca kaynakları bırakır.



Dosya adı parametre adını kontrol ediyor açıyor sonra dosyayı okuyor kaydedip kapatıyor. Library açıyor.

opendir } Bunlar sistem çağrıları  
readdir

(sistemi çağrısı)

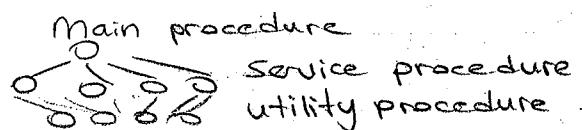
Bu fonk çağrılinca trap işlemi oluyor. Normal işletim sisteminde geçiyor parametreyi alır. Hangi servis isteniyorsa belirler, servis yeri belirlenir servis tara edilir sonucu getirir.

User modu: Birden fazla program olabılır.

Micro kernel: Client server

(tek parça) Monolithic system: OS büyük bir prosedür topluluğudur.

- Tüm prosedürler aynı seviyededir ve birbirleriyle otluşusluudur.
- SVC işletim sisteminde koir servis böyle istenir.  
(supervisor calls)



Layered system:

0. tabaka CPU yönetimi → process switching
  1. tabaka bellek " → swapping
  2. " şref operator iletişim
  3. " I/O
  4. " user program
  5. " operator
- batch systems  
"THE" OS

Her bir tabaka diğer tabakalara virtual machine gibi davranışır. Ne yaptıgı ile ilgilenmez.

Sistem çağrıları

OS

VM / 370

Hardware

Bu sistemlerin çakışmamasının sebebi?

OS tüm sistemleri elinde tutuyordu.

Büyük programı kontrol etmek zor dur.

- Driver'ı elde ederiz.
- Tekrar compile edeceğiz kernel'ı } eskiinden
- Reboot machine

→ Günümüzde plug and play

Driver takılacak otomatik olarak sistem driver'ı bulup kullanılır.

micro kernel:

Kernel sadece gerekli servisleri kullanır. Hizlasm. Fonksiyonları client.

- File server'a istek OS üzerinde gelir - kernel üzerinde  
Dagitik sistemlerde kernel kullanılır.

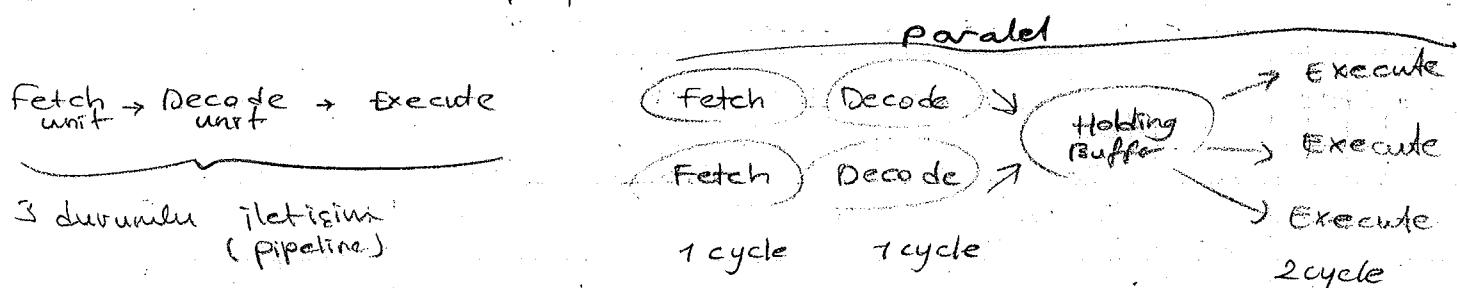
Kernel'in fonksiyonları: { CPU yönetimi,  
bellek yönetimi,  
mesajlaşma,  
IO



→ Bunker aynı servise ana kernel üzerinde erişir.  
Alt serviseye trap ile erişir.

\* microkernel File server'ı da regiszirebiliriz.  
Haberleşme protokolü değiştirmeyeceğiz.

Kernel'in harcanan performans zamanı artarsa -



3 durumlu iletişim  
(pipeline)

Parallel sistemlerde çok zaman alan durumlarda  
bırakın fastla kaynak sağlayarak aynı zamanda bottlenecks.

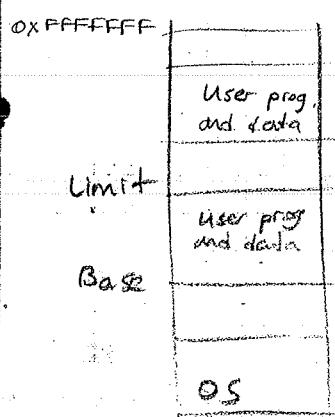
	Register	1capasite
	cache	- 1KB
	Main memory	- 1MB
Backup için	Magnetic Disk	- 64 - 512 MB
	Magnetic tape	- 5-50 GB
		- 4TB Göz kullanılmıyor.

Bellek  
hizyearsisi

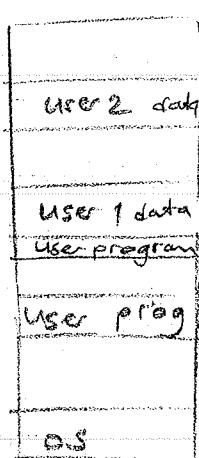
→ size artar hız peki değişmez.

→ hit problemi parallel disk ile çözülmüş.

One Base Limit Pair.

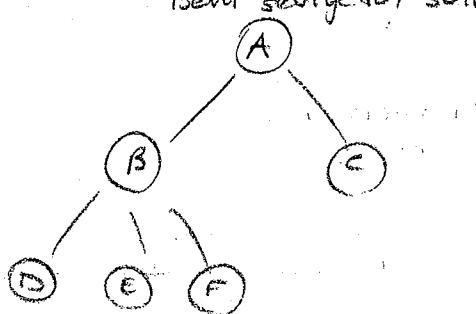


Two Base Limit Pairs



→ Burada bir program, bir kullanıcı kullanır farklı dataalar.

Adres bus'ı 32 bit.

- Level 1 cache → CPU'nun içinde.
  - Mouse keyboard'dan sonrası simdiki sistemlerde yok.
- Süreç Ağacı: Linux  
Belli seviyeden sonra bin verir.  

- Windows iken herseviye hizmet verir.  
Kullanıcı istediği gibi betirle

Deadlock: OS deadlock ile önlem almasa sistemi çözebilir.  
OS en kısa zamanda en doğru çözümü bulmaya çalışır. En ideal diye bir şey yok.

Pipe: Süreçler arası iletişim

femto exa bunları balecelsin.

En yüksek işlem hizina sahip bilgisayar peta flop

### PROCESS:

- Sıralı bir işlemci üzerindeki hizasını bir program.
- Süreç kendi içinde her zaman sıralıdır. Parallel olmasa söz konusu değil.

### 1 Program Counter:

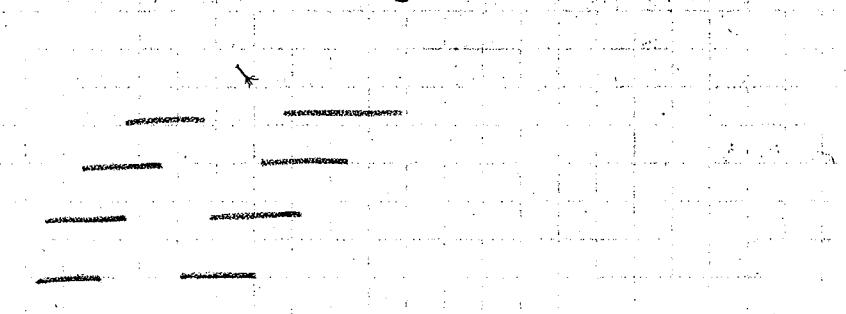
A	A Durdur
B	B hizası
C	B durdu C hizası
D	

Tekrar A hizasına geliyor.

Biri durunca diğerinin program counteri tutulur.

→ Genelde bu kullanılır.

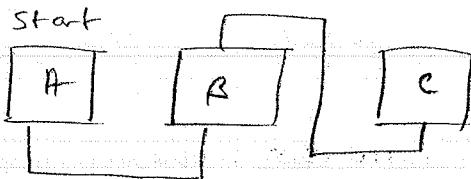
### Birden fazla program counter:



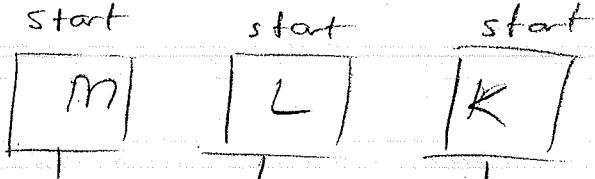
A bir süre hizasına B sonra C, D sonra A hizasına yinele devam ediyor.

Zaman paylaşımı olarak hizasını sağlıyor.

## Sequential Execution



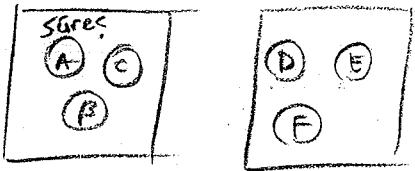
## Concurrent Execution



A sartaninda Bye geciliyo.

Terminate Terminate Terminate.

- Birbirlerinden tamamen bağımsız farklı zamanlarda yenilenebilir.



A, B, C ile D, E, F paralel  
ekendi halinde concurrent (yanayaylaşımlı)

- concurrent aynı anda hepse həşər.

Süreclerin concurrent koşması nasıl sağlanır?

Process i başlatınca code, data, process table (process control block) oluştur.  
Programda varsa süreç ile eklenir.

Process table içinde:

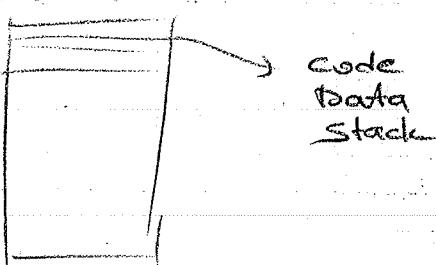
program counter içeriği, register içeriği, code data ptr,  
process escelesi, parent child processlere ptr, hangi işlenmede  
koşular (çölli işlemcili alanlarda)

- ✓ Registerləri her bir process kəllənir. Durdurulan processin  
prog counter process table in təsne sahlanır.
- ✓ Program yeniden başlatılınca (tanif halinde) restore (yaparılmaz)  
registerlərə və ilgili alanlara geri yüxtə.

Switching process: ciòh rəqəmli bir işləm  
Her zaman (yapılmamalı).

Bu işləyi olaya  
dərin.

## Process Table:



Proseslər aynı olabilsə PÜRE  
ve data farklı olmalıdır.  
Aynı code yarınca mənşəli  
değil. Paylaşılıyır. Sonuçlar  
farklı oluyor.

Process önce həqiqi huyraguna atılır. Həqiqət process hətəcək  
sonra sıra gelen deşər. (schedule) dərin.

Koşma süresi kuantum biten preempt edilir. Huyraga  
geri atılır. Kuantum bitmedən sonlanırsa silinir.  
Input output bağlarısa suspend edilir. Blocklər askıya  
alınır.

→ Bloklanan processler ancak input output hizinde continue yapılıp tekrar kuyruğa atılır.

Birden fazla telsimi varsa birden fazla RUNNING olabili

Sonlanan process yeniden başlatılarak tarifederebilir.

Sonlanan processin hellek alanları başta processlere tahsis edilir.

Kuyruğa daha öncelikli bir process gelirse kapan process preempt edilebilir.

Zaman alıcı olmayan process bloklanmas.  
swapped out: RAM'de çok yer kaplayan diske atılır.

Senhronizasyon dayı için bloklanabilir.

Prosesler yaşamalar boyunca kuyrukta kuyruğa dolasırlar. Bloklama sənse hadar süresce process deadlock'a girmisti.

scheduler: Tarifelemeyi tarifeleme algoritması kullanarak (process switching) yapar.

- interrupt handling de yapar.

### Yeni Bir Process Üretme:

fork: Kapan bir process kopyasını oluşturur.

exec: program içinde başka process

system initialization ile process üretebilir.

Parent child process yaratılır.

fork başarılıysa parent'in 2. bir kopyası

başarısızsa child 0 ise child process salıfır.

child hep 0

parent 1 2 3 artıyor.

process id = my\_threadid(); kendi id'sini görür 1 görür.

- sistemin verdiği id ler farklı

- processlerin kendisi içindeki id'leri farklı.

\* Process içinde yeni process yaratmak -

-1 olması hata ile sorulma -

## Process Sonlandırma (Termination)

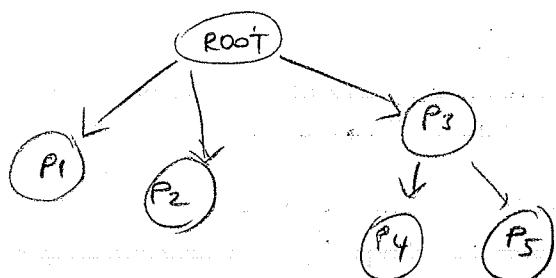
## Normal

## Error

fat at

Başka bir process ile öldürme.

## Parent child Tlishisi:



→ Child sent from parent - parent son -  
lammaz -

P<sub>4</sub>, P<sub>5</sub> in herdi childlari olatoqilir.

- UNIX gain process group designation
- Windows gain helps groups

## Process Table ierigi :

- Process Management
  - Memory "
  - File "

NUA Bellchler begrenzt  
UMA - Herhangt birthe  
tarifstrukturell erstmals espt

SORU: 4 tone işlemci boş. O arada 1.. islemcide kogar ne yapmalı? Neye göre seçim yapılır? Belirtileri birbirinden bağımsızsa önce hangisinde kastıysa orada kogar.

## PROCESS & THREADS:

- Prosesstore sistem çağrısi ile ulaşılabilir.
  - Thread (hafif process) → user level, kütüphane çağrıları var. (sistem çağrı yok)
  - kernel level (sistem çağrısı yapamaz)

TCB: Thread Control Block

Thread'i durup baslatilmek icin gerekli igerisi:

- program counter
  - register set
  - stack space.

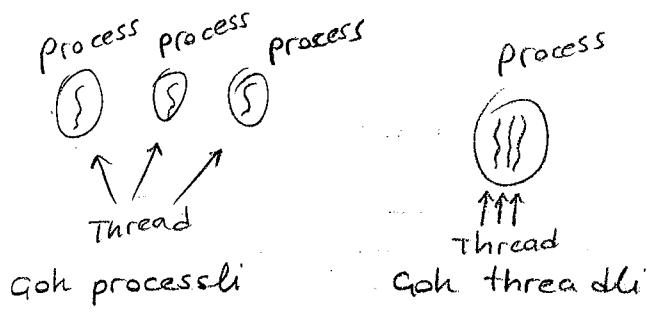
Kernel level TCB'si küçük process'e göre  
Kernel threadin olduguunu bile bilmez.

Bir adı thread taranı paylaşımlı eklentiler

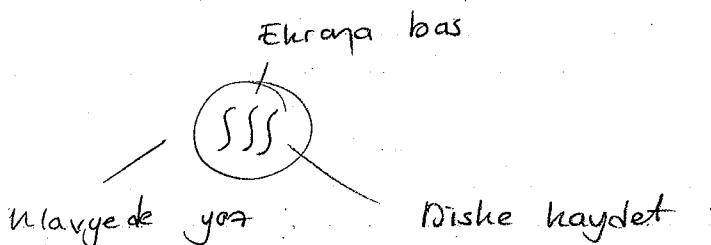
$\rightarrow$  Start

Chat programlarında ekranında direkt

çok threadli yarınca  
görebilirsin.



- \* Goh processli olanda sisteme büyük yük getirir.
- \* Kernel level threadin hemde state'i var.



\* Eğer böyle olması için ekran'a basması, diske kaydetmesi için ayrı ayrı enterlara basmalyız.

- dispatcher thread
- worker

\* Programın kaldırabileceği kadar worker thread olabilir.

### Popup Thread:

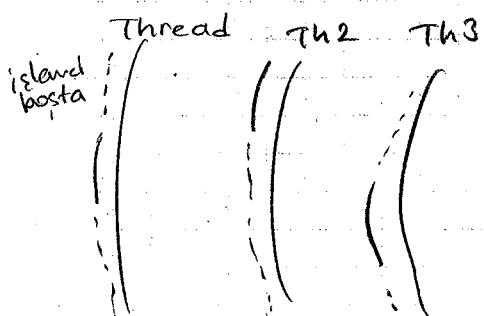
- mesaj almadan önce tek bir thread sonra sollu geçir.

Parallelism: farklı threadler farklı işlemelerde Goh threadli program yordigimda özel bir efor harcanır. za gerek yok.

Throughput: Bir işleminin birim zamanda gerçekleştirilen process sayısı ifade eder.

### Responsiveness:

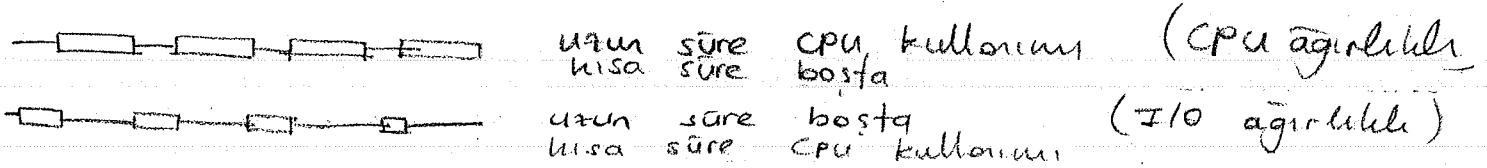
Bir thread kosaçak, biri oturşımıza gireceğe kullanır. İle kullanıcı için tepkisellik önemlidir.



Birişi durunca diğer haberleşebilir. CPU kullanım artar ve slurut.

## SCHEDULING: (Tarifeldendirme)

- Prosesleri seçip işlemci üzerinde hıyar - (scheduler)



I/O işlemlerinin önce başlaması ile CPU ile parallel Çalışmalar için scheduler I/O ağrılığının tarafını tutar. CPU kullanımını artırmak için.

### Process Seviyeleri:

- İşlemcide olabilir.
- Yarısı bellekte yarısı dıştadır.
- ikinci bir bellekte olabilir.
- Başlatılmamış bellekte bekleyen olabilir.

Fairness: Her bir process'e hâkemliği zamanı verir. (öncelikle)  
policy enforcement: sıra politikası

Balance: işlemciyi %100 mesgul etmez.

Throughput Birim zamanda yapılan işler miktarı.

Turnaround time: Bütün işlerin m[ın]

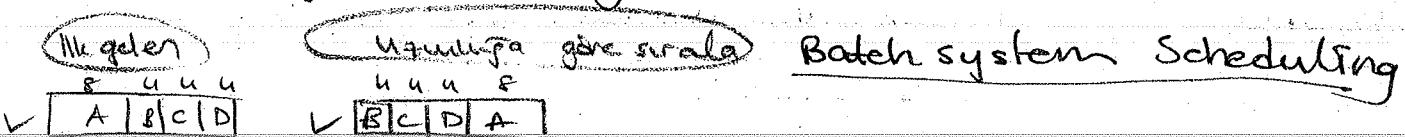
CPU utilization CPU kullanımı Max.

Response time: Hızlı cevap ver

Proportionality: Beklentilere cevap ver

Meeting deadline: Veri kaybı elmasın

Predictability: multimedya sistemlerinde bozulmaları önleme



Batch sistem.



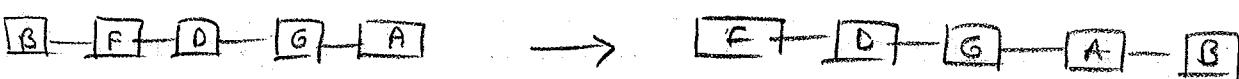
İşlerin tamamlanma zamanına göre sıralıdır.

İşin bitme zamanlarının sırayla topla

$$8 + 12 + 16 + 20 = 44 \quad 4 + 8 + 12 + 20 = 44$$

Interactive System  
Scheduling Algoritmaları

Round Robin Scheduling: Öncelik önceliği değil.



- Çalışma sürelerine göre sıralanır.
- İş biten sona atılır.
- 5 surec, her bitti 1 tur atması 10 ms.

- Tarifeleme algoritmaları sorulabilir.

$$\sum_{i=1}^n \frac{c_i}{p_i} \leq 1 \quad c_i = \text{zaman} \quad p_i = \text{periyod}$$

Real time systems.

$$\sum_{i=1}^2 \frac{1}{2} + \frac{2}{4} = 1 \leq 1$$

Real time'da başlatabilir.

- Long Term: Sıraya sokulup sokulmayacağı
- Medium " : Andottage atıncak mı
- Short " : işlemciyi elegeciyor mu
- IO : Hangi process I/O işlemi verilmeli
- Turn around Time: işin kabulu bitmesi
- Deadlines: Deadline uyulmalı.
- overhead: İşletim sisteminin getirdiği süre miktarı

Enforcing Priority: Kaynakların mesgul olmasın Dergili olsun.

### Önemli faktörler:

- Process interactif mi batch mı?
- Öncelikler
- IO mu CPU mu bağımlı
- Page fault: Prosesi sayfalara bölüyoruz. Bir sayfa bulunamıyorsa bu olur.
- Ne sıklıkla process kesilmiş?
- Proses'in aldığı kosma zamanı
- !! tanınmaması için gerekli kosma zamanı. (Bu bir tahmin)

### Scheduling Types:

→ Kosan process OS ile durdurulmuyor tanantlarına kadar kosuyarsa non-preemptive

→ İşlem OS tarafından kesilebilirse preemptive  
Günümüzdekiler böyle.

2. ise bir interrupting clock olmalı. Interrupt zamanı periyodikdir.  
interrupt zamanına quantum denir.

Quantum süresi ve sırası zamanıalsa interaktiflik olmaz  
Bir sürecin sürekli kosmasını öner. (habettiği kadar)

### FCFS! (First Come First Serve)

- Sıra sırası var. Bir bitince arkasındaki devam ediyor.
- Non preemptive
- Kısa ve IO bağımlı süreçleri cezalandırır.
- (,) kısa olurlar önce koşarsa turnaround time kötüleşir.

## Round Robin (RR)

Her biri s"reg kuantum süresi kadar doğan

Preemptive

zaman paylaşımlı, efektif

Sadece I/O bounded cerealoğrular, (Burdalar öncelik verilmeyen)

- FIFO türün kuantumu bitince 2. sonra 3. sonra yine 1.

### Quantum Size:

Eğer quantum size çok büyük seçilirse, RR = FCFS

" " " " " küçükse sürekli süreçler değişir.  
Kaynakların çoğu process switching için kullanılır

## VRR (Virtual Round Robin)

2. kuyruk var birisi hazır ready süreçler biri de AUX I/O yapans  
AUX önceligi farla -

I/O süreçleri kalın zaman kogar, 15 ms kuantumu varsa,  
10 ms koytu sonra AUX'a katıldı 5ms kogar.

Round Robin den daha iyi bir performans var. Ama daha  
fazla bilgi tutuluyor.

## Priority:

Her birine öncelik verilir. Her zaman öncelikli önce kosa-  
bilir. Bütünlerinin sürekli kosmasası için öncelikler  
zamana arattılır. \* I/O öncelikli

## Priority Classes:

Priority Class	000	Highest
PC3	00	
PC2	000	
PC1	0	↓ Lowest

- Her sınıfı içinde RR ile paylaşılır
- Üst sınıf kossa bir alta gelir

Öncelikle zamanla göre ayarlanmazsa alt sınıflar ölmeli

## Shortest Job First (SJF):

- En kısa zamanlı olan önce yapılır (batch sistemler)
- Non preemptive
- FIFO da bekleme zamanı az
- Ortalama turnaround time
- Her bir görevin ne kadar sürediği
- ( $\rightarrow$ ) Uzun süreli olanı sürekli hedef

## Shortest Remaining Time (SRT):

- Bir hedefin daha kısa sürede birisi gelirse hedef durdurulur o hedef.
- Quantum yok. Preemptiflik RR'den farklı.
- Kullanılan servis kaydedilmeli.

## Highest Response Ratio Next (HRRN)

- Hep ençelikli olanı gelirse ençelikli hedefler
- Non preemptive
- Priority =  $\frac{\text{Bekleme Zamanı} + \text{Service Zamanı}}{\text{Service Zamanı}}$
- Sürekli hedeflerin öncelikleri sürekli artar.
- Öncelik zamanı gerekçizdir.  
Beklediği sebece öncelik artar.
- Kısa işlem öncelikli

## Feedback Queues:

- FIFO mantıklı değil doğru giden bir hukuki.
- Eğer bir process'un quantumu bitmeden 110 yapmışsa nereden ayrıldığı yerde döner.
- CPU dateli ayrıldığı yerden bir alta intiyor.
- Bu işlem CPU bounded en alta nereye kadar serer.

Dispatching: - list serviyeler keşke CPU'ya gese,  
- Aşağıda therefore büyük quantum verilir.  
(Process switching atlayorsa)

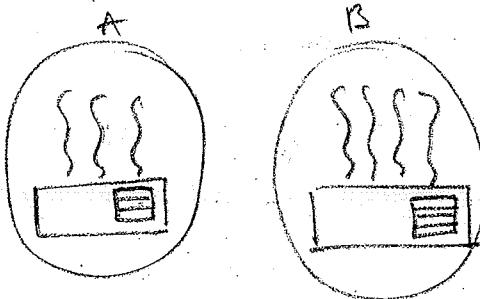
modification: - Bazen sisteminde 110 ayrılmaması bir üstüne de gelebilir.

\* CPU bounded wasıra input yaparsa hər tək qəzetlətilir.

## Scheduling Mekanizması:

- kısa process } bantların yanında olmalı.
- 10 bounded }
- işlerin kendi doğasına göre tari felcmeli 10 ms 11. ms de 10 olmalı. 10. s de hedefsen process switching artar. Ama bu her zaman mümkün değil.

## Thread Scheduling:



Bir process kesarken başka sürecin threadlerini tarife etmeyecek.

- A<sub>1</sub> A<sub>2</sub> A<sub>3</sub> A<sub>1</sub> A<sub>2</sub> A<sub>3</sub> → mümkün

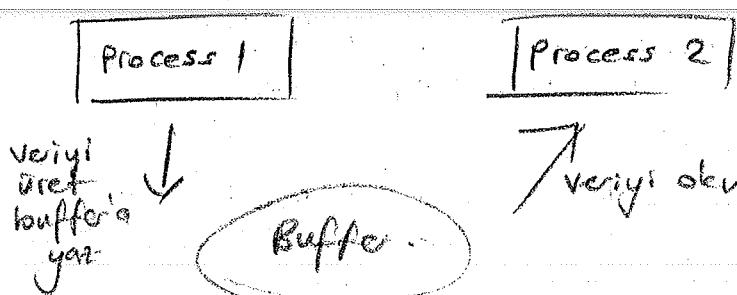
\* kernel tari felcme yapıyorsa A<sub>1</sub> B<sub>1</sub> A<sub>2</sub> B<sub>2</sub> A<sub>3</sub> B<sub>3</sub> Bu da mümkün

### Policy versus Mechanism:

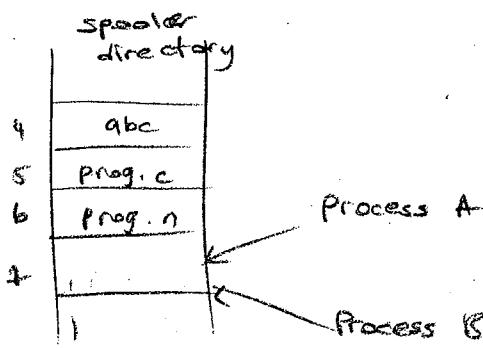
- ✓ Ne yapılmalı ne yapıldı bir ayrimi olmalı.
- ✓ Bir processin bir sürü child processi var. Parent process child processlerin önceliklerini ayarlayabılır.
- ✓ Parametreleri vererek tari felcmeyi kontrol edebiliriz. Sistemi yanlış hale getirip hale alabiliyoruz.
- ✓ Parametreler user tarafından girilebilir ve optimum tari fe

## INTERPROCESS COMMUNICATION:

Asıl sorun ne?



P<sub>1</sub> 1'i yazi P<sub>2</sub> kullanı.  
 P<sub>1</sub> 2yi yazi P<sub>2</sub> 7'i kullanıyor su anda, 3'un tamponuna girildi  
 P<sub>1</sub> bu sıradaki 3ü yazarız iken 2 yi bularak



A önce yaradı 7 ye 8 e geçti.  
sonra kesiildi. B 4 ye yaradı  
8 e geçti.

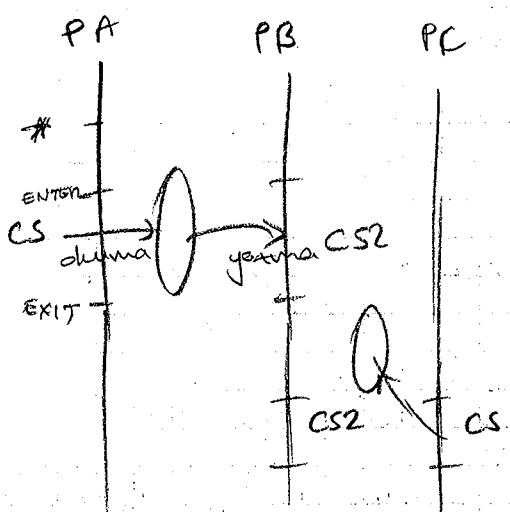
A yok oldu.

### Race Condition: 'Yukarıdaki.'

Bu durumda her process hizmetinden yarlış sonuc

#### Mutual Exclusion

(critical Region): Ortak paylaşım alanlarına erişim yapma konu bölgesi.



Biri critical section okuyan storiaunu dışlanır gerdiriyor.

✓ Aynı anda iki process CS de olmaya calı.

✓ CPU kilitlenmesi, sistem kaynaklarında yorum yapma. Bunlar değişebilir.

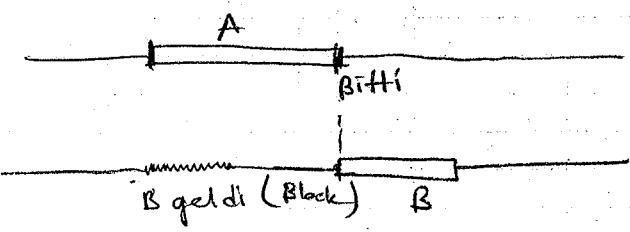
✓ \* diğerinin CS girmesini bloklaması.

✓ Fairness sağlanmalı yahsa starvation olur.

#### CS: Paylaşımı bölge.

Mutual Exclusion: sadece tek bir bölgeyi CS de olmasının sağlanması (Kazılıklı paylaşım).

#### Critical Region (2):



Neden processler haberleşir?

- içlerini senkronize etmek için (Veri alışverisi burada yok)
- Veri ve bilgi alışverisi

Bir nöktaya erisen diğerini bekliyor.

## Starvation: (Mutual Exclusion Problem) Indefinite Postponement

- Bir processin sürekli ertetmesi
- Nedeni: Tarifeleme algoritması bir taraflı desekler. çoklu yalanlama örnegi.

Deadlock: 2 veya fazla process hizlabilir zaman olmaya cah bir olaydır. Belirleme.

- Mesele A, B yapmasını bekliyor, B de A'yi bekliyor. Exclusion Nasıl Implement Edilir?

- Uygulama servisinde
- Hardware "
- OS

enter - critical - section ile başlar  
exit - critical - section ile kapar.

down (&empty); } Bu satırlar yer değiştirse.  
down (&mutex); }

önce mutex 0 oldu. Process bloklandı. Diğer process mutex'i ele geçirdi full'ü up yapar. Buffer dolunca down empty de bloklanacak. Program deadlock'a girer.

Semaphore Operations: return değeri yok her zaman gerekli

- Wait (s)  $> 0$  ise  $s - 1$  değilse blocklar. (down)
- Signal bloklanmış process'i kaldırıyor.  $s + 1$  (empty)

Dogru veri alışveriş ve senkronizasyonu sağlar.

Bu operasyonlar her zaman gerçekleştirilebilir. Bir sonuc olmazsa 0 se down, değilse bloklanır. Her zaman başarılıdır.

- Monitor: sadece bir process aktif baskısı bloğu geliştirse öncelik almamasını bekler. 0 aktif 0 bloğu kaldırılır.
- Fortran ve Java

producer boş alan yehsa terkedir (item üret buffera yar)  
consumer item yehsa " (item al tüketir)

signal (full)  $\rightarrow$  bloğu kaldır

\* Sleep and Wakeup: Buffer doluiken sleep -

Scheduler: 2. prompt etmeye çalışırsa çalışmasın.  
Bir daha wakeup işaretini olmasın.

Monitor(2): Bir process bloklandığında monitora väzifa edilir. monitor boşken wait (full) çalışmaz.

öncekiinde tüketici signal (full) gönderdiğinde kaybolduyor da bloklandığında " monitora uyarı gönderebilir.

\* Burada bloklanmayı garanti ediyor.  
Buffer boşken prompt edilir. then wait hizmində bloklandı. producer monitor içine girməsi consume içinde bloklandı. Sonra tüketici blocklər. (wait empty) sonra üretici tekrar tərtifləndi.

↳ Blocklənmis bir işaretə uyari gettiğin icin uyanabilir.

### Message Passing:

- Senkronizasyon + mesajlaşma
- Dörtlük yapılarında.

Buffer uanlugu 100 üretici item üretir buffer'a yoxdur. Tüketiciden boş mesajı alır. Üretici ürettiğim mesajı yollar. Tüketici önce m tanrı boş mesajı yollar. İlgili itemi alır tekrar boş mesaj söyler. Aldığı itemi tüketir.

mesaj alınca receive (buffer sayısı hədər boş mesaj yollar mesaj yoxsa bloklandı).

Send çağrısına bağlı bloklandırma olmaz.

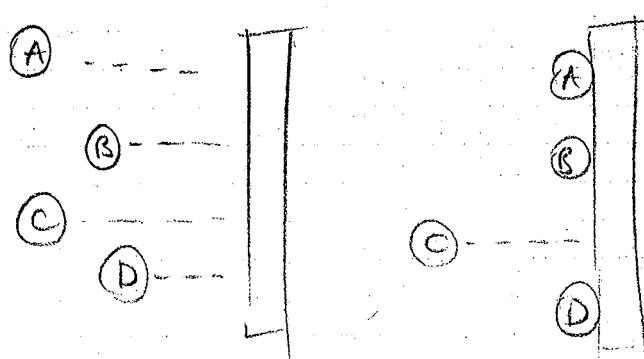
Tüketici m tanrı boş mesajı gönderdi. Extract ile gereklişləri alır. Hər aldığı icin buffer yer açılır boş mesaj göndərir.

send } Yer deşince performance düşer.  
consume }

### Barriers:

Bu şəhərdə procedure çağrıları yapılmabilir.

Barrier (Prod., ...). icinti oluşturur.



Digerleri C yi bekle C gelince biter.

A, B, C, D ortak bir veri üretti. Hepsi ulası (Senkronizasyon) təmə

Barrier yapısını mesaj geçme, semaphore ilə gerçəkləşdir.

// monitors (4) bunu yas //

- Ortak bir process oluşturular
- Hepsi ortak process'e mesaj yollar
- Barrierde getince processlər mesaj yollar ortaya -
- Tümü getince ortak process de bir mesaj yollar

## Dining Philosophers: $s_i$ nin başlangıç değeri 0 olur.

- kaynak kullanımını doğru bir şekilde olmalı. Max 2 kişi aynı anda yemek içmek 2 çatal olmalı.
- Deadlock herkes tek çatal alırsa - bu olmaya calır.
- Hepsi soldağını ele geçirildi sağlığını alamadı. Deadlock oluyor.
- Aynı anda iki çatal ele geçirilmeli. take-forks ( $i$ );  
Bunlar bir procedure.

take-forks önce mutex ele geçir. state = HUNGRY ; test, down.  
put-forks " " " " state = THINKING ; test, up.

test komşuya bak. yiyebilirse yemesini sağlıyor. yiyemeyeceğinde birakıyor.  
çatalı ele geçirirme sebebi up mutex, down -.

~~down (&  $s[i]$ )~~; başlangıç değeri ne olmalı?  $s[i]$  ler ne ise yarıyor?

çatalı ele geçirilemeyecek bloklanmak. down (&  $s[i]$ ) burada.  
 $s[i]$  başlangıç değeri 0 olmalı.

EATING konumuna geçince up (&  $s[i]$ )  $s[i]$  1 yapılıyor.  
up ile block kalıdırılıp yemesini sağlayabiliyor.  
Alt procedure kullanılıyor.

test (LEFT) } çatalları burada serbest bırakıyor.  
test (RIGHT) }

## The Readers and Writers problem - Banka oturum sistemleri - Bilet rezervasyon sistemleri

Writer exclusive olur, reader paralel olabilir.  
(tek tek olmalı)

rc : reader counter  
down (& db) okuyucular ve yazıcılar buraya erişmek için yanışır  
önyükle mutex up  
yazıcı up db

## The Sleeping Barber Problem:

Bekleme hizmetçileri, boş değilse up mutex yapıp filkeye mesgulise block.  
" " " " boşsa berber, berber

up (&barbers)  $\rightarrow$  berber uyanır.  
up (&mutex)  $\rightarrow$  mutex serbest bırakır.

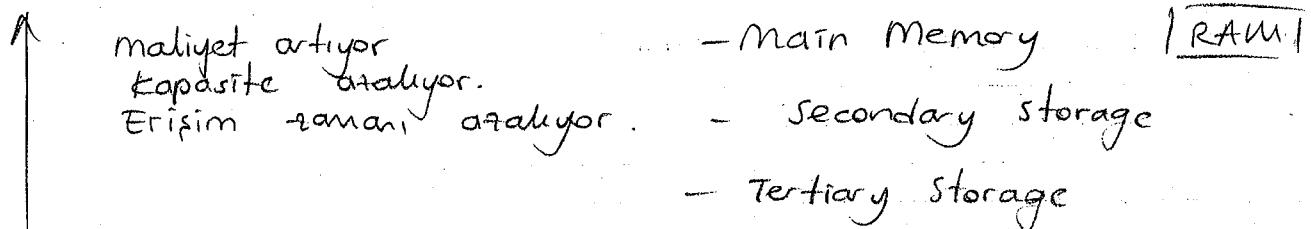
müşteri bitince berber uyuyor.  
up (&customer) berber uyandırır.

Hem müşteri hem berber bloklararak bloklu.

Berber sürekli çalışır.

→ process A, B, C de / burlar senkronizasyonu ifade ediyor.  
→ ise senkronizasyon yoktur. Diğerini beklemeden solanır

## memory Management: (CHAPTER 6)



Contiguous: Programları pepspe yükleriz.  
Non-contiguous: " "      "      değişik boşluklu yükleriz.

Real memory: Gerçek fiziksel bellek (Bellek erişimleri burada)

Virtual memory: RAM + harddisk burların mantıksal haflesimi.  
(Programlar burada yaşar) Adreslemeler burada.

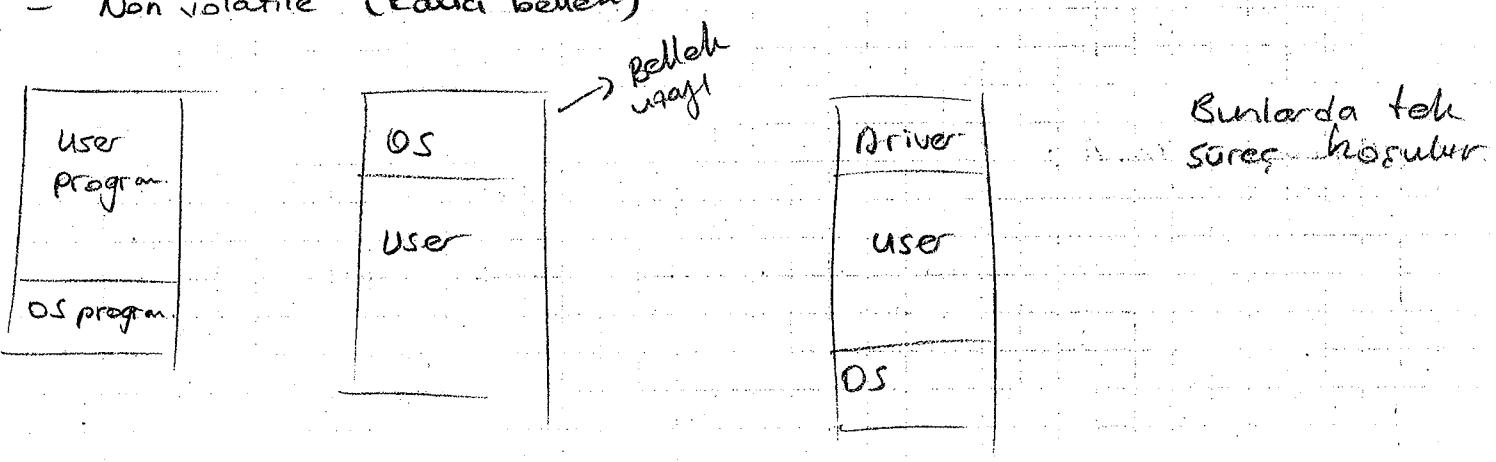
Block: Hard diske toplu eklenen bilgi 256 byte

principle of locality: Program utarının belki belki alanlarında erişme

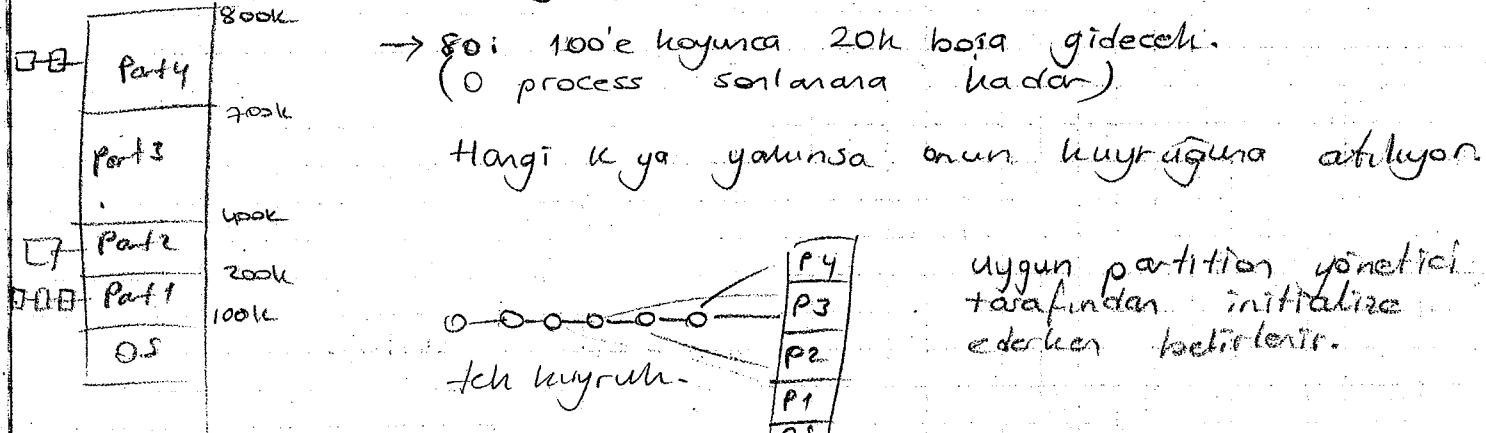
Fragmentation: Belleğin verinin kucuk parçalarla bölünmesi  
(İsletim sist. açısından büyük bir problem)

Programı ne istiyor?

- Large
- Fast
- Non volatile (kalıcı bellek)



Sbt Partition: Sadece 1 process yüklenebilir. Fragmentationdan dolayı  
boşa gidiyor. Kuyruk olursa daha iyisi.



No. kadar. fakta surec həşərəsə CPU kullanımını artar.  
8 tane həşərə CPU kullanımını  $\approx 100\%$  e sıtıyor.

$$\text{CPU Utilisation: } 1 - \frac{P^n}{(kullanım)} = 1 - \frac{0.8^3}{(0.8)^3} = 0.488$$

CPU kullanımını  $\approx 20$  olan processor 3 tane kullanımında  $\approx 50\%$  e sıtıyor.

Processorların CPU kullanımına fəhlə rəs

$$1 - p_1 \times p_2 \times p_3 \dots p_n$$

Bunlar I/O kullanımını arası -

I/O kullanımını çox olurlardan çox həşərəsi, CPU " " " " " 1-2 tane hələ  $\approx 100\%$  yaxşıla sıtabilir.

	1	2	3	4
CPU Kul.	.80	.64	.51	.41
Process. Kul.	.20	.36	.49	.59
CPU/Process.	.20	.18	.16	.15

$$\begin{array}{l} \text{5in } \approx 0.49 \\ \text{2nin } \approx 0.59 \end{array}$$

$$0.24$$

$$1.2$$

$$0.8 \times 0.8 = 0.64$$

$$1 - 0.64 = 0.36$$

$$0.36 / 2 = 0.18$$

Contiguous - m. A.: [OS] User Prog.]



600 nöqtə digəri yəhənərək ikinci parçaya yüksəldəndən sonra bellekde böyük programlar (Overlay)

Fixed partitions: [OS] Process 1 | Frag | P1 | F1 | P2 | F2 |

Dinamik partitior

OS bütün sırasıyla P1, P2, P3 sonu 608

OS P1 boşluq P3 (P4 sığınır)

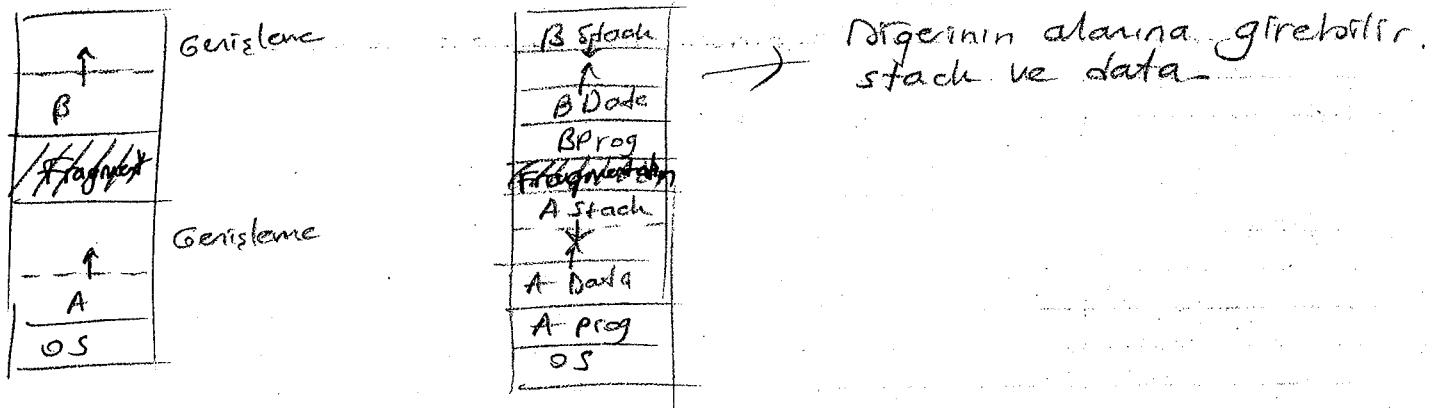
OS P1 P3 P4 relocation (çox pahalı yer degistirildi).

~~soft~~ memory allocation: 10 ns  
Bellek erişim  $1 \cdot 10^6$  kez  
m21 1GB

} Relocation yaparsak ne kadar zəmanət alır. Təməntə kəydiur.

- Relocation dan sonra boş bellek alanları, program seviyesinde tahrif edilmeli. Base ve limit belirlenmelidir, güvenilirlik için adres değişimi yapılmalıdır.

### Swapping:



Unused Memory: Renteli olan yerler kullanılmıyor, sadece boş.

Günümüzde işi biten Bitmap 1 kullanılır 0 kullanılmıyor. Başkaşının alanına girmeyen aralığı bulmak için taranır.

Linked Lists: 6 dan başlayarak 3 tanesi



### Placement Algorithms:

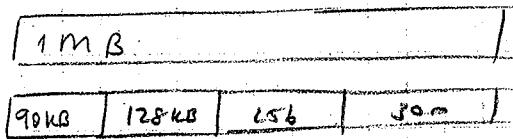
First fit: Yerleşebileceği ilk aralığa. (En baştan) - En hızlı ve hızlı performansı.

Next fit: Kaldığı yerden itibaren tarar. - First fit'in sonradır.

Best fit: Tüm aralık tarar ve en uygun alana (Performans kötü, küçük miktardaki fragmentler)

### Performance Issues:

#### Buddy System:

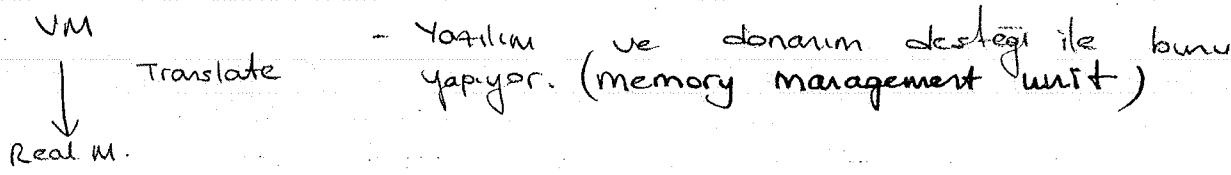


- Hizlı tahrif yapar
- Fragmentasyon var. - (Kullanılmayan bellek)
- Bazi parçalar his kullanılmıyor. Bular, bellekte yahut emzirme çöküğüne neden olabilirler. (Kullanılmayan parçalar)
- Proses size, fiziksel bellekte bağlıdır.

Bunlara cevap olarak virtual memory kullanırız.

## VIRTUAL memory: Kavramsal

- Sınırsız programın belleğine göre
- Programın gordüğü sanal bellek. Gerçek belleğe itdeşim yapıyor.



## Paging:

### Virtual Address Space

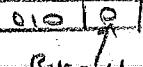
- Sırasıyla adreslene
- 64K 4Klik bloklar

### Physical A.S

- Rastgele adreslene,
- 32K 4Klik bloklar.

Bellek dolunca birini silmeye yerine başkasını yollatır.

### Page Table: 1b k adreslenir

- Böyük programlar kogalın ana neden bellekler kullanılır. Program hızlı kogalılması için parçaları bölüyoruz.
- Bir sayfa gerekunce   
fiziksel adres
- Sayfa değişimini olunca bu adresi bir bölgeye yazar.

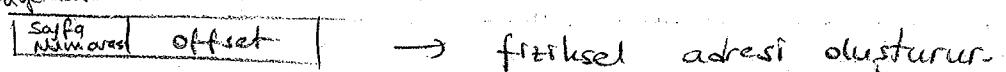
\* virtual adres offset, table adreslene olmak kılınır.

Program bozuklukta geldi Page fault oldu. Bellek doldu. Page replace algoritması. Değişim olur belleğe yazar ordan kosalır yine yeri yerleştirilir.

- En fazla fiziksel bellek kadar yani 8 sayfa yahşetir.

- ✓ Programda biraz bozuk olabilir. (son sayfanın yarısı boş)
- ✓ V.M. hangi sayfaya istek varsa ettir, yükleyir
- ✓ Process size'ı fiziksel bellekten yükseltir dekabılır.

page table setle -



- main larası + isteğe göre diğer sayfları yahşetir

## Interpretation:

4K 12 bit  
1b sayfa 4bit

Page Frame Numarası

Present / Absent bit

Page Protection

Reference bit

Modification bit

Baska sayfalara gitmesini öner.

0 sayfaya gitme yapılmış. Sayfa değişimi istenince swap area'ya yda öyle depişti.

### Adres Mapping :

Page fault olunca

Bu neden olur?

sayfan yok.

Interrupt - trap

8. sayfaya geldi

İstan. Sayfa hizmetler alır.

OS işlemci durdurur

8. sayfa

32 bitlik adreslane.

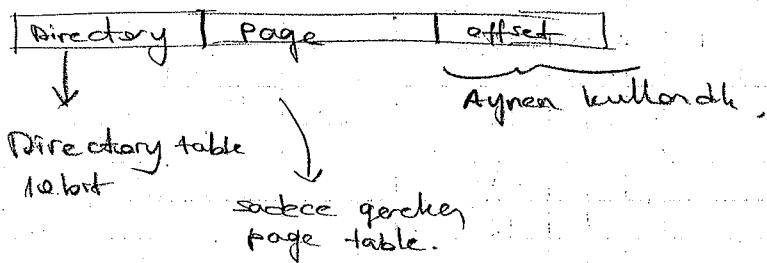
12 bitlik sayfa

20 bit 1MB

0 0  
1 1  
2 2  
3 3  
4 4  
5 5  
6 6  
7 7  
8 8  
9 9  
A A  
B B  
C C  
D D  
E E  
F F

Page table lain 4MB ayırmak gerekiyor.  
 $(2^{22})$

### Two Level Lookup



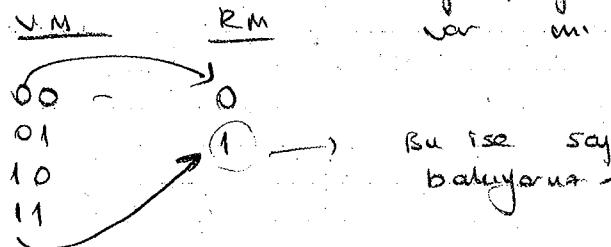
- Dersantay geçilme  
- Daha etkin bellek kullanımı.

### Inverted Page Tables:

V. Adres utayı 64bit page size 4K direct mapping page table tablo  
muh. iain ne haađar olañ lozum?

$2^{54}$  byte.

Fiziksel bellek lain bir page table oluştur. V. Adress kulan offset haric  
diğer bilgi Hash table balyq fiziksel bellekte  
var mi bulur yerini de bulur.



### Translation Lookaside Buffer:

Page # → TLB → offset ile birleştir. gerçeli bellek.  
(Hizli bellek)

Page table  
yolu

- TLB dalmışsa sayfayı atıyorum içeriğin adresleneleri bellek

TLB de adresler de belleğe kaydedilmesi.

- Process son sayfasının yarısı kaybedilir.
- Sadece istekli bölgeler.
- Process size sınırlı.
- Process sayfası sıralı yakalama konuda zorlu.

### Segmentation:

- Segmentler değişken boyutlu.
- Programlar farklı parçalardan oluşturulur. main, data, stack.

Segment #	Offset
-----------	--------

- ✓ Bir process'in birden çok segment olabilir. Her bir segment rasi segment boyutu, numarası, Baslı Adresi burda olmalıdır.
- ✓ Symbol table çok büyükse olguncu sorun (source text'e) her birin farklı way olması daha iyi.

- Programı segmentli yapının faklunda önceden gereklilik ana pure ise bilmeli.
- Segment sayısı kadar虚 address olmalıdır.
- \* Procedure ve datalar ayrı edilebilir farklı segmentlerde olabilir. Bu segmentationda var.
- Tablolardan boyutu değişim kolore edilebilir segmentation.
- \* Neden üretildi? mantıksal olarak bagimsiz alıcılar, korumalı ve aynı bölgeler bilgi paylaşımı.

Pure segmentation versi struktur bellek yönetimiinde sonunda karşılaşıyor.

### Segmentation with Paging:

İkişinin birleşimi ile gerçekte adres elde edilir.

Segment #	Page #	Offset
-----------	--------	--------

VM RM

(page table lar)

- Her bir segmenti parçalara böleriz.
- Page frame numarasına ulaşırız.

- 9 bit sayfa numarası.
- Pure Segmentation
- Sayfa

### MULTICS

Page Number	Offset
6	10

Segment numarası ile descriptor  $\rightarrow$  Page frame  $\rightarrow$  Adres  
page number page

TLB hizli erisimde kullanan bellek. Bilgi geçerli olmalı. Segment num + Virt. page  $\rightarrow$  Page frame.

Pentium: Selector var  
Index      |    |  
              1    2

- Local  
- Global

Segmentin base ve limiti var.  
- soyfaya göre  
- byte'a göre

10      10      12  
Dir      Page      offset

- 0  $\rightarrow$  Kernel
- 1  $\rightarrow$  Sistem accisi
- 2  $\rightarrow$  Pay. kitapçığı
- 3  $\rightarrow$  User program

Ker degru  
sistem acci  
ile emirlerin

## OS Policies for Virtual Memory:

Fetch Policy: Sayfaları swap array  $\rightarrow$  bellekten yoktur.  
Hangi sayfaları yükleneneceğin karar verir.

Demand! Sadece istiyas duyulan sayfalar yüklenir. Diğerleri atılır.  
(page fault ols.) (Rastgele sayfaları yokuş)

Prepaging: Mesela 10 sayfa yüklenir. Hersei kullanılmayabilir. Peşpeşe  
sayfalar hizli yoktur (Mazinden sorakiler sıralı hizli)  
- Fazla sayfalar, hiz adrestenmesse verimli olur.

Placement Policy: sayfanın adının RAM'de nereye yerleştirileceğiz

Uniprocessor: Pure segmentation, sıralı foli. Yerine yerlesmeli  
sayfalar, "rastgele" yerlesebilir

NUMA: Bellek cihazları farklı eşit değil. Dağıtık yapıda olsun.  
Process. hangi işlemicide kullanıysa, o işlemcinin lokal belleğin  
de nehangi hiz yere yerleselerdir.

Replacement Policy: Bellek dolu page fault oluştu. Yer hiz  
sayfa getirilmeli. Nereye getirilecek - Bir  
sayfa atacak yeri sayfa onun yerine. Hangisi kullanılabile

- Page fault forces choice - 1+ fazla sayfa kullanılabilir.
- modified ekranları swap array'e kaydetmeli, olmayanların üstünde yap.
- sık kullanılan sayfaları saklama. time goes getebilir. tekrar istenir.

## Replacement Algoritmaları:

- Girilecek sayfa, en az referanslaracağ sayfa olur.
- Locality den dolayı geçmiş ve gelecekte kullanılmıştır. Bugün çok kullanılmış olan gelecekte de kullanılacak.
- Geçmişteki sayfa kullanılmıştırna bakılır.

Optimal!: Referans alg. (Gereklid degil). Bir algoritma performansı buna bakarır.

- NRU San taraflarda kullanılmayan
- LRU " en az kullanılan
- FIFO sık kullanılmayan

Local scope: Page fault yapın replace

Global scope: Herhangi bir processin seçip replace

→ Bu frameler kilitlenir. Kernel, buffer sayfaları sıkarsa sorun olur.

Optimal Replacement Algorithm: LRU kullanıması

- Gelecekte kullan süre sonra referanslanacağı kullanırız kullanırız
- Gerçekleşmesi mümkün değil
- OS gelecek halimda ideal haliyle isahip olm. gereklidir.  
sayfa referans dizisi:  
V.M. 3 sayfa, R.M 5 sayfa.

\* 1 in referanslarındaki yer yek 1'i çıkar. 2 daha sonra referanslanacak onu çıkar.

→ Bos olaları timan page fault. (2,3,2) → 3 tanesi  
çıkardıklarımız 3 tanesi page fault

NRU:

Referans ve modified kullanırız

R bit → Okuma (yazma) setle

M bit → sayfa değişmesi yoxsa, R de setlerir.

Hardware: R, M her belde ref. olunca - setle

Software: R, M " page fault old "

Prosesor başlayınca R, M Olsun.

Belli clock interrupt " "

④ R bit her clock interrupt ile resetlenir

Page fault olunca boş oluyor ona akt sıfırlan bir tanesi eklə

Class 0 ref - mod -

Class 1 ref - mod +

Class 2 ref + mod -

Class 3 ref + mod +

(reference, modified)

→ kötü bir algoritma

FIFO: circular buffer ile en yaxlı sayfa çıkar.

→ sistem sıkçı geri aldı system thrashing, Geraklensə basır

R, M bitlerine bakarak çıkarmaya karar veririz:

(yaxlı ama sık oluyanları çıkarması için.)

gen 5 tane page fault 2 en yaxlı çıkar. (6 tane page fault)

Second Chance P.R.A:  
R.M bitlerine baktırır  
sek oluyor.

En çok kullandıkları ikinci bir sans ve  
tümünde. Setlerde de daha yoksuk.

clock P.R.A: Sact kimi gösterirse  
 $R=0$  hold.  
 $R=1$  diğerle geçer.

LRU: En azin rastgele kullanımları, implementation sorunlu  
performansı optimuma en yakın.  
→ Her biri için zaman etiketi olmalı.

Hardware Imp. of LRU:  
- Kullandırma göre liste yap.  
- Her bir erişimde listede güncellenecektir.

Approximate:

- Her bir emirle counter +1
- counter değerini page table'da tutarız. Page fault olsun.  
ca counter değerini en az olanı çıkar.

Software Imp. of LRU (NFU)

- R bit değerini page fault olusunda alırda page table elle.
- min. counter değerine sahip sayfa değişir.
- Yaklaşık tam deger değil.

0 tüm satırı 1'e, sütunu 0'lıyoruz. Hangisi referanslanmışsa  
onu tam yapırız.  
→ Hangisini page replace yaparız en küçük değer olanı  
degistiriz.

\*\* En azde referansları 1 degeri 0 olduğundan

0 ise referanslanmadı  
1 ise referanslandı

101011 ekle sola kaydırır

En küçük counter değerine sahip olan 0010 0000 cihazdır.  
8 clock interrupt gerince sayfalar aynı oldular.

) Bu yöntem yedekleme yapar. Kullandığın sayfalar cihazları

ÖRNEK

Burada yedekleme yok.  
Başlı counter 0, R biti olur.

Son rastlarda yükselen ana counter küçük olmaktadır.

modified NFU → LRU yedekleme var sola kaydırma

LRU, NFU dan dahan syi'iyi orijinalde daha yahin... (aging)

**NOT!**

Optimal	Gereklilikmet.
NRU	Kötü bir algoritma
FIFO	Önemli sayfaları atabılır
Second Chance	FIFO'ya avantaj sağlar. R m baka
Clock	Uygulanabilir
LRU	İyi ancak zor
NFU	Kullanılması zor. (Geliştirilmeleri zor)
Aging	LRU + Aging

Betady's Anomaly: Virtual S (0-4)

- a durumunda FIFO varsa g Page Fault verilmis Page fault olmasin, R bitine balsaydik olabilir.

## Stack Algorithms      virtual (0 - 7) → 8

5647  
book bina

tic dala since yaklermenis istenirse o kayara astma  
Bellekte olanin encebi yesi 3an yesi 4 ya 4 yonelik  
yolculukta 4 yolculuk astance.

Aittali bolge fiziksel kellek degit Aseyi inise page falt ok

Distance string'e bakarab process'e soyfa veriyor. Page fault atalar.

Son referansları en üstte alınırlar.

Distance 4 ten büyükse Page Fault olur.  
En alttaki istenirse 8 derdir.

$$F_M = \sum_{k=n+1}^{\infty} C_k + C_{\infty} \quad \text{Process'se 1. first used blocks versen}$$

↓

Page fault says:

$$C_2 + C_3 + \dots + C_{\infty} = 20 = F_1$$

$c_1 = 4$     $c_2 = 2$     $\dots$     $c_\infty = 8$    böyle sayılarımıza bakın.

Bu proses'e kaç sayfa verirsek optimum olarak koyar bu şekilde

## Resident set Management

Prosesin burların Anabellikte (RIM) de olan seyfaların sayısı ve tahlis politikaları.

- Resident set kütüklere P.F. silme, artır.  
- " " " fasta process icat etmek, Daha fasta  
hastır process bulunur.

Mesela  $f$  den daha büyük segmentin bir enlemi yok.

## Allocation VS Scope:

- Her bir process'e svt sayfa ver. → fixed allocation
- " " ihtiyaca göre " " → variable allocation

Fixed: P.F. olmussa replacement PF olana yapilir. (local) sayıfa sayfa

Variable: " " "

(Global) → replacement tüm process sayıflarıın seçilir.

\* Process Resident Seti PF'a göre değişir. PF olana sayıfları yüklerken teforansformasyon silinir.

Global fixed mümkün değil!

cümlü A'yi B'ye aktarırsak A atanır B artar svt katma

- \* Ne kadar art process kasarsa CPU kullanımı azalır  
swapping → Prosesler yer degistiriyor. Tekrar yüklemezi zaten alt kucuk processlerin artması PF artırr.

## Local vs Global Allocation Policies:

1b sayıfa var.  
Anın en yaglisı A<sub>5</sub> → <sup>yagli 3</sup>A<sub>6</sub>  
tümünün en yaglisı B<sub>3</sub> → <sup>yagli 2</sup>A<sub>6</sub>

Local → cirosuya istenilen prosesler PF art Prosesde daralma yok  
Global → zanırda B yede olabilir.

- Global olanda en yagli A<sub>5</sub>, C<sub>1</sub> 3 olmuş olsa yukarıdan sağlı bakan
- Local " hangisi page fault olmussa ona bakan

## Working Set 20 sn- süreli main var.

- Main program ile baskan.
- 4,5,6,7 ye en az 20sn eristi. 25sn toplanda. Burası bellek altısa, P.F. atanır. Birińi çıkarırsan thrashing. Sürekli geri yükler.

4,5,6,7 çok kullanır.

Burada 4. yek mesela o processi kosma. Beltek kesip onu yükleyene kadar bekle.

n clock tick aralığı belirleriz. (Dereysel olarak belirleriz)  
Sürekli referansları sayıfları dahil edebiliriz - sayıflar svtler.  
Resident setteki sayıflar sürekli değişir.

## Cleaning Policy:

modified sayıfların ne zaman bellekte yerləndir?

mi olmuş sayıfları çıkarmalı yeri sayıfları yüklenen extra tanır.

Page Buffering: - Buffer dolmadığı sürece sade daha etkin oluyor.

→ Clearing işlemi batchler şeklinde toplu şekilde. Bos frame aralıkları bu işlem yapılır. circular list'e bakarak işlemi yapıyor. Hep aynı sıradır değil.

(+ik) Load control: Koşabilir süreç sayısını belidir. Swap'a bekleye ter sahipmeyiz.

- Göz process varsa Resident set döndür. Yokusuk PF, trash'ın blur.
- Prosesler interaktif haldeki şekilde yapılmalı. (Orta seviye)

~~SPM~~ 0.2 sn etkileşim limite 10 ms kuantum verilirse. Bu şekilde hangi process yüklenir?

( ) 0.2sn → 200 ms.  $\frac{200}{10} = 20$  process

N Tüm tasarımlar iyi olsa lbb trashing olabilir.  
O Bellek ihtiyacı çok dabit.  
T BO'lari listede bekleyebilir. (swap) Koş - bekle.  
Gök bellek isteyenleri osalt.

### Implementation Issues:

- OS'in Paging ile ilgiliğine.
- Program size belirle page table oluştur. (Process üretimi)
- Process koşarkey MMU reset  
Page fault olduğunda yarını belirle hangi sayfa okunacak senturma tüm page tabloları kosalt.

P.F olurca,

Hardware trap ve kernel jüreye girer.

Registers kaydet

Hangi virtual bellek gecidi

Page frame bulunulsun. Belirle

Modified disk yap.

Yeni sayfayı diskten getir.

Page Table update. (cikarılacak 591)

Fault olurca instruction başlangıç dururus.

Fault olur tekrar tarif edilendir.

Registers restore

P.F. olur tekrar kollar.

### Instruction Backup

1000	MOVE	opcode
1002	6	first operand
1004	2	second "

→ Sayfa move a hedef kosulda sonus hatalı.  
PF aralarda olursa emir baştan tekrar alınır.  
kaldırılsın yerden deşil

## Page Size Uzun Olması:

### Avantaj

- Internal fragmentation azalır.
- Kod blokları ve data structure nyunu kolay olur. Sadece kullanan yerler bellekte yerlesir.
- Kullanılmayan kısımların yıklanmaması

Detavantaj: Page table boyutu sistem çok soyadır olusur.  
Bellek boyutu boyutu boyut.

- Bellekten okuma hızı düşer. 32 kB olsun. 2 kB lik eriş 16 kB lik veri. 16 kat hız düşer.

Optimum page size ne olmalı?

$$\text{overhead} = \frac{s.e}{p^2} + \frac{p}{2}$$

process size  
 → s.e  
 → soyadın  
 yarısı kadar  
 varsa P olur  
 Mayip  
 → page  
 size

internal fragmentation  
 ile olur mayip

Page size 2'nin katları  
 okuma yatma döşen  
 olsun oluya

page table → p ye göre tarey alıncaya)

$$-\frac{s.e}{p^2} + \frac{1}{2} = 0 \quad p^2 = \sqrt{2s.e}$$

## Locking Pages in Memory:

Bir process I/O için beklerken diğer başlayabılır.  
Sistem processları (sistem programları) (kernel) bellekte saklanır.

- \* page replacementta bu processler dahil edilemez
- \* Buffer bellek, aborted, mesnul like kosalılmıyor.

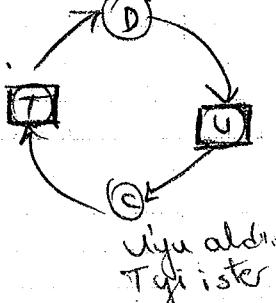
## DEADLOCKS (Chapter 9)

Birede süreçlerin elindeki kaynakları bırakmadan önceki süreçlerin kaynaklarını istemek - Kaynakları geri alamazsın deadlock olur.

Process D: Hiç olmuyacak bir sayi process beklerse  
System D: 1+ process deadlock olmuşsa

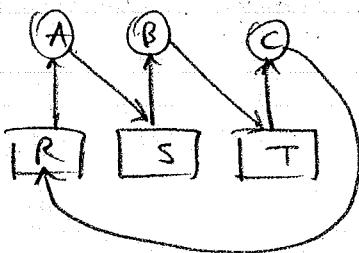
Tüplü  
Uyuşmaz

D ve C birbirlerini  
beklerler.

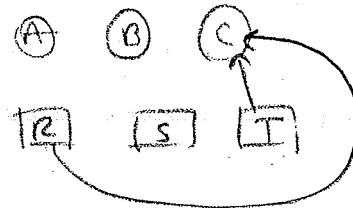


## Deadlock önleme:

- Tahsisî düzgün yaparsak,
- mutual exclusion garantisi et, teli kaynak kullanma,
- önceden parkedereli



İstek sırasını değiştirecek, B'nin isteklerini geçiktirerek önlenebilirdi.  
Ancak, bu çok pahalı bir işdir.



Bunlardan en az biri önlese deadlock oluşumu önleir.

mutual Exclusion: Her kaynacı aynı anda 1 sırada kullanır.

Hold & Wait: Elinde tutuyor, istiyor (pinning filozof yöntemi)

No Preemption: Bir sırada verilen kaynakla zorla alınamaz. Kendisi bırakın.

Circular Wait: İki veya fazla sıraç zincir şeklinde kendisine önceki sıradan kaynacı bekler. (kaynaklar, orta - sırada processlerdir).

m.E!: Printer, spooling yaparak kaynak paylaşımını önlebilir.  
her say "yapılmas"

- Gerekmedikçe kaynak tahrîsi yapma
- Çok az sayıda kaynak istegeç yapılır.

HW: Proses'in istediklerinin hepini ve yada hiç vermemesi.  
Başkasının elde etmesine verilmey.

- Yüntem calısıyor ve kodlaması, kolay.
- Kaynaklar boşa kullanılıyor. (sırası gelmediği için ele geçirildiği starvation olabiliyor. İstediği halde kaynakları elde etmeye fırsat yok elde etmeyebilir. Bu kaynak isteyenlerin elde etme ihtiyali az.)
- Kaynak ihtiyacı biter belirlemeyebilir. 1, 2, 3, 4 istedi 1, 2, 3 ele geçirildi 4 kaldı 1, 2, 3 birbirinden torundan.

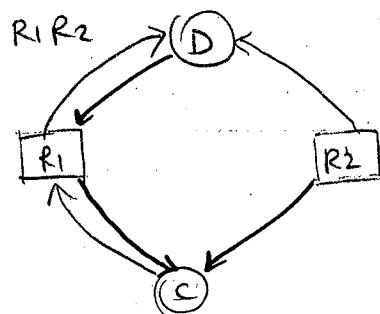
- No Preemption: 1 kaynak istegeçinde diğerlerini bırakmak zorunda olmasını istiraden verilebilir.
- Kaynak kullanımı staka ryi, erdiriyor.
  - Pahalı bir yöntem.
  - Diğer kaynakları ele geçiremediği için starvation

Kaynakların geri alınma durumu printer için uygunlaşır.  
Arka sayfa başkasının saçması.

circ. Wait: Kaynaklar numaralandırılır. Prosesler sadece kaynakları orta sayıda istegeç bilir.

- Yüntem calısır. Baslı os lar da kullanılmış.

- Problemler:
- İhtiyaçla göre isteyenler.
  - Her eklemde yeniden numaraları分配.
  - Kodlanması zor.



Eğer D başlarsa R1 istedi alır. Békler. C de R2 istemeli istiyor. olsa sırasa göre R1 istemeli zaten de békler D R2yi istedine alır. işi bitince C ye geçti.

\* UNIX'te deadlock'a neden olan process'ları nasıl kurtulur?

### Deadlocktan Schüttma: (Deadlock'a neden verebiliş)

Durumu kontrol et deadlock olacak mı istemeyle söyle mi?

- kaynak tahsisini doğru yap.

Preemptable: kaynaklar sorsa geri alınabilir. (Belki)

Nonpreemptable: " " olursa sorun olabilir. İlkhakını ve kaynakları ele geçirilemezse paketlenebilir. (Yaslı)

### Banker Problem:

(Borsa)		(İstek)	
Max Need	Loat	İstek	İstek
800	410	390	390
600	210		

$$\text{Anapara} \quad 110 + 210 \\ 1000 - (620) = 380$$

→ İşin ek para verenin 1 firsat.

### Dijkstra:

P: Tahsis edilmiş

E: Tam kaynaklar. önceden elmine düşenler.

E ( )

P (5 3 3 2)

A (10 1 0)

P (4 2 2 1)

A (2 1 2 1)

\* En fazla hangisi kaynakları geri dönecek olan koşar (A) max sayıda process kozmak.

D-A - ?

Tahsis 1.

İstek 2.

## Dijkstra's Banker Algorithm:

MAX NEED < toplam kaynaklar  
kaynaklar sorted birde anında getirilebilir.

- Proseslerden biri tamamlayıp yacalısa kaynakları vermezlik.
- SAFE state da kalmışsa kaynak isteklerine cevap ver.  
(deadlock olmaması)
- \* Galler, da sayıda kaynak verdiğinizin (SAFE de kaldı.) ekrana yazdırır.
- \* Stat sayıda kaynak gerçekte kaynaklardan toplam kaynakları sistemdeki olur.
- \* Proses kaynarkeyen max ihtiyacını değiştirdiğe deadlock olur.

Ostrich Algorithm: Develusu gibi haberlerde olma.

- Deadlock nadiren oluyorsa problem yok. Ondan sonra ilgileniriz.
- Windows ve UNIX bunu kullanıyor.
- Uygulanabilirlik  $\begin{cases} \text{Purlo arası seçim yapar.} \\ \text{Degrubuk trade off} \end{cases}$

Deadlock Detection: (Tespiti) 18.11.2014

Circle programla algılanmak çok zor. Maliyeti yüksektir.  
Uygulanabilirliği az.

Deadlock Recovery: Deadlock'a sahip olan process kaldırılır. Tüm verileri silinir.

Processten kaynaktan alımları kesip, kaynak nonpreempt ise, kaynaktan ilk verildiği duruma getirilir. Bu rollback, checkpointer oluşturular. process halledi bir yerde devam eder.

- Checkpoint nasıl programda eklenir?

\* Önemi düşük processlerden başlayarak sonlandırma yapılır.  
Sistem processleri kullanılamaz.

Safe:  
Yes / No

A	3	9
B	2	4
C	2	2

A	3	9
B	4	4
C	2	4

A	3	9
B	0	-
C	2	7

A	3	9
B	0	-
C	7	7

Free: 3

Free: 1

Free: 5

10 kaynak  
B ye 2 tane  
verildi

B bitti  
B'in kaynakları  
free

C ye verildi

Sonra A ye verilir.

Unsafe: kaynaklar B ye veriliyor tamanlanır. C ya verilirse  
2. 4 → 9 ihtiyaci var & tanı old. için tamanlananız.

### Problems:

- Sistem processleri sonlanırsa, sistem sonlanır
- Boxen yararla process deadlock'a girer. Belli bir sureye göre olursa deadlock'ı kalmamaya çalışır. veya re-boot
- \* Deadlock'in türlerini nasıl görürüz?  
Bloklanmış process CPU'yu tanımı tüketmez.  
Task manager'daki durumuna bakılır.

## CHAPTER 10 FILE SYSTEMS:

Algoritmalar ve veri yapılarını kullanarak mantıksal dosyalanma ile -  
miktari fiziksel hale çevirir.

- Veri depolama ve üzerinde işlem yapma
- Hataları minimize eder
- Performans optimize eder.
- Veri kayipları minimize eder
- Farklı I/O device destekler
- Colu kullanımı destekler.

### Long Term Information Storage:

Diskte tutulan bilginin ömrü 5-10 yıl

### Kullanıcı ne beklər?

Sembolik isimle erişim

Dosyaları kontrol etme, silme, oluşturma, yeniden yapma, kopyala, yedekleme, değiştirmeye.

### Files: isim + içeriği.

LINUX tipi çok önemli değil

\* Byte, record, tree şeklinde tutulabilir.

→ Extensions bunlar önemlidir.

Erişim sırası ve rastgele erişim.

Sequential: Tüm kayıtları baştan oku, attıra, gel sırma. Her sırma ypk. sıralı okuma ortamında olur.

\* Random: Rastgele okunabilir. Database sistemlerinde

File marker bir noktaya getir sırada okuyabiltirsin.

- Çok zoruz alır.

File attributes: Read, write, execute, archive, hidden, system.  
creation, last access, last modification.

file operations: append, seek, get attributes, set attributes  
eklene file marker numarasi erisilebilir agresi satılık değiştirme

→ File attributes farklı OS'larda farklıdır.

\* An example program using file system calls - örnekini kendin yapmağa çalış.

(-) Bir dosya içeriğini başka bir dosyaya kopyalar.  
Bunu stem cagruları ile yapar.

Memory Mapped: Dosya bellekte tutulur. Parallel olarak P/W

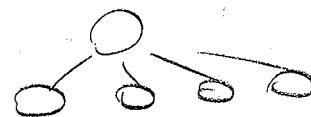
Directories: Attributes burada saklanır.

Directory entry - DOS windows  
farklı bir veri yapısı - UNIX

\* path names.

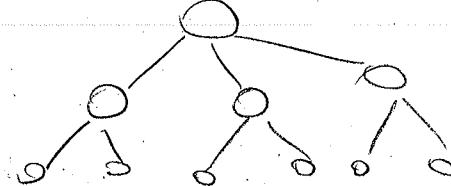
\* Operasyonlar - link, unlink.

Single - Level Directories



Two Level

"



Windows  
Ent.)

NTFS

Hierarchical: Dosyalarının oluşturmasa kopya, onları.  
BUT sadece kopya erişebilir.

UNIX'te kullanılan directory klasörleri yapılabilir.

(mutlak) Absolute path fr / di / dr / fr

Relative path fr : fr

Physical Disk Space Management:

- Diskler tracklardan enaz sector / block lardan oluşur.
- Sector en az 512 byte.
- Disk başı farklı tracklarda okunmak için yer değiştirebilir.

Bu konu yerini sonrala silicon diskler oluşturur.

Disklere dosyalar nasıl yerleştir?

- contiguous
- Bağlı listeler
- Bağlı listeler + index kullanır. (DOS) file systems (UNIX)

Super block: File sistem halikendali tüm parametreler bura da tutulur.

Contiguous: Rastgele erişim yapamazsin. (Tüm blokları oku sonra eriş) Farklıdan 1 block istesek bir sürü relocation formu. Aradan silsek fragmentation.

FAT : name, start block, length.

Dosya tamamen birin buları birbirer yeter.

### Linked List Allocation:

\* Fragmentation bizi alan yoksas linkler ile halleştirilebilir.  
eliminated

- Random access sorulur.



Dishit rastgele dağılmış varsa R/W çok yavaş olur.

Bir bloğu okumadan önce öncedenki blokla, da okunmalı gerisi

Linkler FAT (File Allocation Table) içinde.

Veri dosya tutarlılığı var mı, bunda bakılır.

A) 4 7 2 10 12 -1

B) 6 3 11 10 -1 File allocation table  
EOF

FAT için ne kadar alanı ihtiyac var?

Pointerlar tablo içinde farklıdır.

$$16 \text{ bit} \rightarrow (65536 + 2) * 2 = 131076 \text{ byte}$$

$$32 \text{ bit} \rightarrow (x + 4) * 4$$

Block size bir blokta hangi boyutlu dosya tutulabilir?

\* 8KB - 16KB - 32KB FAT 32 2, 4, 8 TB olmaz.

Block size büyükse çok fragmentation olabilir.  
65536 1KB 64MB

Dosyamız lokdan büyükse Single, Double, Triple

Blocklardan diğer block numarasını öğren, ordan 3.ye  
ordan dosya döktür. numarası.

iNodes: Attributeler yolu → Dosya ile ilgili bilgiler tutulur.

Tutabileceğimiz en büyük dosya size nedir?

Block size: 1KB

Block numarası 4 byte 32 bit

Bir block içine 4 byte ka tanır block. 256

Triple 256. 256. 256

2KB olunca, triple: 512. 512. 512 + x ... böyle lügüm kalmaz  
artısları oluyor.

## CP/M File System:

User Code - Dosya - Uzanti - Extet - Block - Disk, block  
Adı and numbers

## DOS Directory Structures:

File Name - Extension - Attributes - reserved - time of creation - date of creation - Path to file - data block

Windows 98: Directory S.

Benzeri yani, adaların boyutları var.

Bugünden: Dosya adı 260 karakter.

Linux Path Name Look up: inode tablosunda dosyaya erişebiliriz.

245 oraya git directory içindeki dosyalara git.  
6 inode git 132. block astı. variyorum.  
mailbox 60

working path  
verja root

## Shared files:

f2 paylaşımlı dr, dr de görünür. Burda aynı inode'i gösterir.  
link count 2 olur. 0 oluncadır dosya silinir.

Data rate → yükseltilebilir  
Disk space utilization → Boş容量 kullanma  
(Disk kullanımı)

## Alttaiki block size

Tüm dosyalar 2KB, 1KB olunca 2KB kayıp olur %50  
Normalde son sayfa yani kadar kayıp.

## Linked List of Disk Blocks:

Disk bossa tüm bloklar, bağlı listede tut. Tabii ki olsaksa listeden sileriz.

## Bitmap:

Her bir disk bloğu için bir adı. Tabii ki en 1 blok esneği 1 blok esneği 0

## File system Reliability: Dosyalarda sist. sagılıklı çalışır mı?

Directoryleri tara kullanıcıları kontrol et, koglu listede var mı kontrol eder.  
Tutarlılık var mı kontrol eder.

b de 2. blok boş bloklar içerisinde geçiliyor kayıp olabilir.  
Altta 0, 1 yapın düzeltiriz.

c Bitmekte 2 olur

Bağlı liste 2 kez eklenir olabilir 1 yapın

d Hangisine 2 dahil olmamalı, hash bilgisi yoksa; manual yorumla

## File system Performance:

Cache üzerinde bilgi tutarak performance artabilir.

Block okunması, cache de olursa okunması FIFO LRU

- Sıkılık açısından olmalı.

## File Position: 16/32 bit

Bir sırada dosyada okunacak veya yazılacak olsun  
- inode → Her dosyada 1 tanesi  
- process table

\* file position aynı dosyaya farklı bir process erişiyorsa farklı  
\* Parent child aynı process table'da olmalı.

### Solution:

process table

parent
child

file position table

position

inode of file

C directory owner C count 1

B, C owner C count 2

B owner C count 1

Dosya linkleri silinmiş

C silindi  
B de silinmiş

B linki nasıl silinir?

- Paylaşımı做的lar bir yerde tutulur. Bu adresden silinir.  
- Tüm sistem tara.

- Disk başlangıcına

- Diskin ortasına en çok erişen yere

- Diskte grup başlangıcına

} disk başlangıcı, haraketi minimize edilecek şekilde disk bilgileri yenilenmelidir.

# CHAPTER 11 ( INPUT OUTPUT SYSTEMS )

Data readler ve yazılmış ana bantlar değişebilir. Genel olarak bilinçli.

## Block Devices:

- I/O blokları halinde gerçekleştir.
- Blok boyutu 128 - 1024 byte arası değişir.
- Harddisk, floppy, CDROM lar, tape'ler.

## Character devices:

- I/O karakter halinde gerçekleştir.
- Terminal, printer, mouse, joystick.

## Device Controllers

PCIe iain electronic card  
mainframe " unit } Ekuma yorumlanan boy  
cihazlara analog işaretler  
geliş.

- Emrin gerçek icrasını gerçekleştirir.
- Disk kafası hareketi ile.
- Farklı cihazlar iain farklı device controller'lar var.

Mechanic components: Seketler  
Electronic " , chipler

\* Bir cihaz controller'le erisim kontrol etmek zorunda değil.

Paralel → seri → Hata Kontrol S → Dizeltme → Alınan bilgilerin ana bellekten alınması veya sraya yazılması.

## I/O Techniques:

### ① Programmed I/O:

İşlemci process adına isteği yollar. Process busy waits gora kaldığı yerden devam eder.

ÖRNEK: Printer mesajı degilse, hârisa yarılabilir.  
Hâris ise 1 karakter yazılır.

### ② Interrupt driven I/O:

İşlemci yine process adına isteği yollar. Process suspend edilir bloklarının yanına atılır. Başka process secalebilir işlemci interrupt ile I/O hattının onları. Blokların sona eren hâris processlerin yanına atılır.

ÖRNEK: Kopyalama başlatır. (User-kernel seviyeye).  
Interrupt enable - tanım gelmesi

. Printer status kontrol eder hâris mi?

Io yapmamış isteyen blokla.

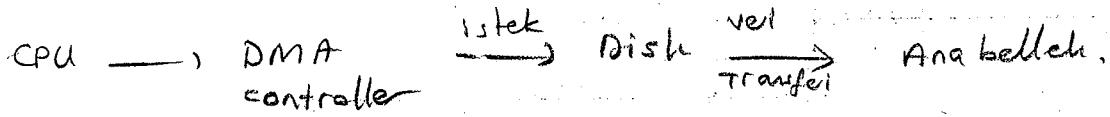
Interrupt servis rutini çalışır.

kernel seviyesinde data printer'a aktarılır count fo  
count = 0 olunca data bitti user block kaldır process  
interruptlar onaylanır (Tekrar interrupt gelmeyece hâda)

i → Global bir değişken

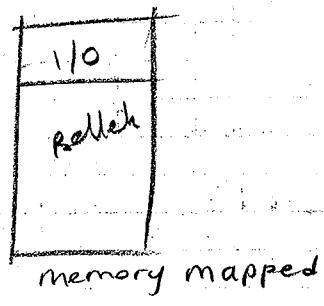
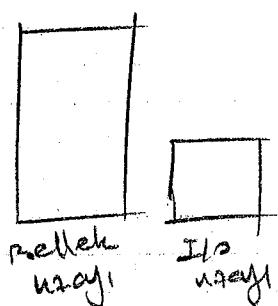
### ③ DMA I/O:

- DMA ana bellek ve I/O cihazları arası veri transferi ile işlenir.
- Veriyi blocklar halinde , DMA ile belleğe veri gönderilmesi alınması yapılır. Kesmeliye benzeyir.
- DMA cihazları I/O işlemleri ni doğrudan yapabilir.
- cycle stealing: DMA , CPU'nun bus'u kullanmasını engeller. CPU aşağı alır. (suspend → I/O bitere haddi) katıldığı yerden devam edebilir.
- Genelde I/O işlemi burst (patlama) şeklinde gerçekleşir.

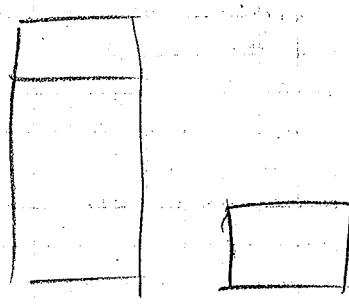


OPENGL: user dan kopya  
DMA controller setle  
CPU blockla scheduler.  
acknowledge Interrupt  
unlock.

### Memory mapped I/O:



motorola (6802)

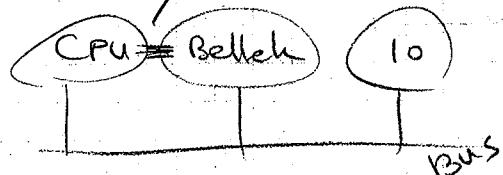
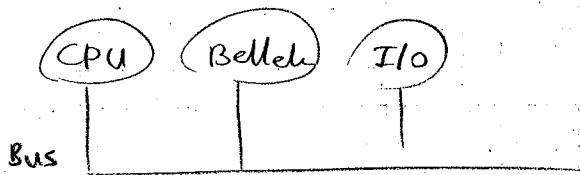


Hitit sistemler

- Avantaj yok
- 1 pikeci, yarınca icin ekranın fırınına yaramasın

- Dogrulan istedigin pihali yaratırsın

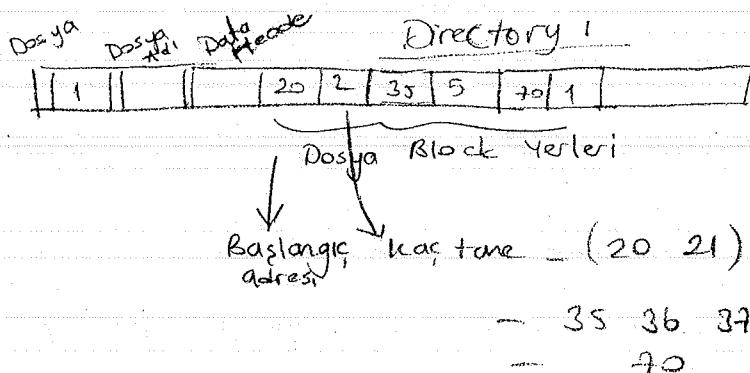
Hüttäge etiket  
bir lors



Gokku bus

- Bellek
- Cache
- Normal bus

PCdeki buslar



## Structuring I/O software:

- User software → I/O system calls
- Device independent software → Naming, protection, blocking, buffering, allocation
- Device driver → Setup device registers
- Interrupt Handler → Wakeup driver when I/O completed
- Hardware → Perform I/O operation.

- User software: sistem çağrıları ile işlemeler yapılıp,  
~~komutlarla~~

- Device Independent: uniform, altınlı.

- Çok kullanıcılı sistemlerde komutlar entegre edilir.
- Cihazların fiziksel block size'ı farklı, mantıksal block size 1KB fiziksel ise ayılar farklılık təkrar təkrar okunur.
- Hata raporlama: Bad block Başarılı okunamasa hata mesajı.

Standart olmayan interface → M → SCL heresi farklı standart interface → SCL SCL heresi aynı.

\* Bir yazma biri okuma yapan 8ki buffer dəbilir (parallelizm)

Network de routerlar verileri okur, depolar, iletir.

## Device driver:

- I/O emirlerini icra eder, I/O cihaz durumlarına bakar.
- İstekleri kuyruğa ekler.

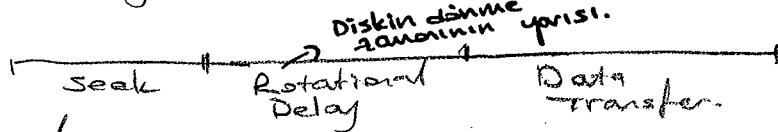
## Interrupt handler:

I/O yapıldığında process ashiya alınır. Bitinec Interrupt handler → device driver. Vənələr yapılırlar sonra istəfədək

- Registerler kaydedilir
- Interrupt service procedureleri ran stack.
- Açıq interrupt portatollar.
- Registerler kopyalanır.
- Servis procedurerler işlenir.
- Məni setle

## Diskler:

- \* Günümüz disklerinin özelliklerine bakımlı yapıda diske daha fazla veri kaydedilebilir.



Disk kafasını track üzerine getirmek  
Ne kadar fazla ise disk kafası o kadar hizlentir.  
ve performansı düşer.

## Yaklaşık Disk Performansı:

$$\text{Seek time } Ts = m * n + s$$

sbit → Geçilecek track sayısı. ↓ Startup time (Rastleyip harekete geçmeden arasımda geçen zaman)

$$\text{Rotational Delay } Tr = \frac{1}{2r}$$

Disk saniyede 100 tur 1 tur 10ms  
rotational delay 5ms

r: rotation speed

$$\text{Transfer time } Tt = \frac{b}{r * N}$$

b: Transfer olacak bit sayısı  
N: Tracklere bit sayısı

$$\text{Ort. Access Time: } Ta = Ts + Tr + Tt$$

8 response track okunmasında 1 tracka göre yap diger 7 tracka 220ms,

$$\begin{aligned} &\text{Kürtgele} \\ &1 \text{ sector okunma } 2 \text{ anı. } 28.8 \\ &256 \text{ " } \rightarrow 256 \cdot 28.8 = 7.37 \text{ saniye.} \end{aligned}$$

- 16.7.2 ≈ 32 1 tur atılıncı diske bu kadar sector okunur

→ Seek time neden esit kabul edildi?  
Track sayısı esit old. için.

## Disk Scheduling:

- ① FIFO: ilk gelen isteği hizlara soran sıralı.

- starvation yok. (Bir istekin cevaplanaması diğer bir istek yet.

$$\begin{array}{ccccccc} 11 & 1 & 36 & 16 & 34 & 9 & 12 \\ \text{Başlangıç} & & & & & & \end{array} \quad \frac{111}{6} = 18.5$$

$10 + 35 + 29 + 18 + 25 + 3 = 111$

## ② Shortest Seek First (SSF)

En yakin olana önce gitilir. Disk hafası hanehedi. 10.2  
Aynı söyle bir sorun var. istekler bastır gelmeye devam ederse starvation olatabilir. 36 adı bekleyebilir.

## ③ SCAN! (Asansör) → 10.0

starvation sorunu var. En iç ve en dışta data gel gitgit. ortadaki isteği bilmeyen. Gitmeye başlanan yönde cevaplanmayan istek halmeye neden devam eder. sevgeridir. Ortadan başlayıp. sonra sıra başa.

## ④ CSCAN! Disk hafası sona erdiğinde cevap vermiyor. \* Performansını düşürtür. (Tek yönü asansör)

## ⑤ FCAN! En iç ve en dışta disk hafası yapımı gibi aynı tracktan istek geldiğinde tıkırır. ~~ve~~ kuyrukları. → Bir scan. istekleri yapmak → istekleri almak.

→ Yeni istekler eski isteklerin tümü cevaplanan hedefler beklerir.

### Common Disk Errors:

- Olmayan bir sector için istek gelmesse.
- Bu durunda isteği sonlandır hata mesajı ver.
- CİD toplamısa birkaç kez olunur. Yine yanlışsa Bad CRC gelirse gerçinse disk formatta.
- Fiziksel olarak disk zarar görmesse. bad block işaretle orada girme.
- Controller hatası, controller degistirmeliyiz.
- Seek hatası. 6. silindire yelash anlaştı gitti. Disk otomatik olarak başa döner.
- Recalibrate: Reset yaparak silindiri 0'a gönderir (disk). Disk onar ya da değiştirilir.

## RAID: (Redundant Array of Inexpensive Disks)

- Güvenlik } Buları artırmak için.
- performans }
- 0-5 seviyeler
- 0, 1, 3, 5 anlatılacak

### 0 Level:

- Blokler doğrudan okulu okuma yapma.
- $0 \rightarrow 1$  - Hiz artar. (2 kat hızlı)
- $1 \rightarrow 2$  - Yedeklenebilir.
- $3 \rightarrow 4$  - Fault tolerans yok.

hard 10: Raid 1 + Raid 0 - Yukarıdan bah. raid 0

- Fault tolerans

- Hızlı okuma yapma.

- Disklerden biri di lense. Sorun yok.

## RAID 1 (Mirroring):

- 2. disk 1. nin ayna kopyası.
- Aynı anda 2 okuma yapma. Okuma performansı iyi yavas.
- Birisi giderse diğerinde devam fault tolerance

## RAID 3:

- Dosya birden fazla diske degit + eki disk.
- 1, 2 ve 3 parity sahibi
  - Birisi kaybolursa parity ile telkari elde edilebilir.
  - Hatta 1'in hamming code kullanılmali.
  - Yavas. okuma hızı.
  - CAD / CAM sinyal işleme için iyi.

## RAID 5:

- RAID 3'e benzer. ama parity tüm disklere ayrılmış.
- Hala okuma yavaş
- Online bakım yok
- N disk için toplam disk kapasitesi:  $N - 1$

## Disk Formatting

Preamble	Data	ECC
----------	------	-----

↓  
sentrize ol ↓ verileri oku      Düzeltme yap.

## Power Management:

- Display cihazların güç tüketimini en yüksek, ana anakart.
  - CPU güç tüketimini en çok yükseltti.
  - Hard disk dökü.
  - Ses sistemleri ortatabilir
  - Bellek - ortatabilir.
  - ✓ Renkliden siyah beyaz cihaza geçer.
  - ✓ Speech recognition hizmete sayısı azalt.
  - ✓ Image resolution düşür grafik kartı enerji tüketimini azaltır.
- Program daha az enerji ile çalışması kon

## (CHAPTER 12) SECURITY AND PROTECTION:

Confidentiality: Sadece izni olanlar erişebilir.  
(Açık bulmaya çalışır)

Data integrity: Verinin degistirilmesi, bütünlüğün bozulması

System availability: sistemin her zaman erişilebilir olması  
Denial of service (DoS) atakları

Intruder: Saldırgan bir kişi

- Ticari ve askeri casusluk } bu yuzden
- Para haramname

Veri kaybı sebepleri:

- Yangın, sel, savaş (doğal afet)
- Hardware ve software hataları (%100 test edilmiş programlar da)
- Veri giriş, yanlış disk takma (insan hataları)

Cryptography:

Plaintext in → Encryption algorithm → ciphertext  
Decryption algorithm → Plaintext out.

\* Algoritmayı gizli tutmaksızın güvenli olmayı söylemek.

Secret key: Alfabe harflerinin yer değişimi

Public key: Birçok kişiye sunulsa bilgi ortaya çıkar

private key: Public keys private bilimlerin olmasız -

One-way function:

$y = f(x)$  yolerde  $x$ 'i elde etmek mümkün değil

Digital signatures:

Dosya orijinalini signature block D(hash) formunda taşır.

User Authentication: (Tullanıcı giriş)

Kullanıcı bildiği  
" Sahip olduğum  
" olduğum şey

magnetic cards: (smart card)

Sistem bir soru sorar ve her tane bir farklı bir sayı sorar  
password de girilir. Kredi kartı.

Biyometrik okuma

Eksik bir hatman  
Parmak izi

### Önlem:

- Sadece belli zamanlarda login olsun (mesai saatlerinde)
- Otomatik geri çağrı (mesajla şifre gönderme vs)
- Login denemeleri sınırlı
- Tüm login işlemleri database de tutulur.
- Basit loginname ve trap oluşturarak hiziyi sonral bir sisteme aktarabiliyor yaptığını babbabiliriz.

Trojan: Free programları Alt programları deşifrelebilir.

Logic bombs: istenatilan personel sisteme zarar verir.

Tramp doors: Bir kaynak kodu içerisinde sisteme girişmesi.

Buffer overflow: Bırular, kullanıcıları başka yerlere erişebiliriz.

- Daha fazla bellek isteği } { Bırular, siberhizlilikleri etmemek için deki veri etc geçerdir.
- Disk adları " "
- illegal çağrı (parametre hatalı çağrıda olmalar)
- Del, rubout, Break klavyede basınca uygulama durdurulur.
- logini kurarsan sisteme girin.
- OS yapısını modify etmeye önde.
- DON'T listesini yaparak
- Trap door içerisinde
- sistem yöneticisini arkadaşına yardım laddi cittinda hizip

TENEX: Page fault olunca sistem yavaşlar.

ugraşarak ilk karakteri sayfa sonuna.

Karakter doğrultusunda page fault diğerine geçer.

- Tek tek kontrol hale getir. Tümdeki kontrol ediyor.

### Design Principles:

- Sistem dizayni public olmasi.
- Default erişimler
- Oturumlu kontrol etmek. (Objede erişimince)
- Her bir process'e en az privilege verin. Sadece o servisyede bir eylem yapsın.
- Koruma basit, uniform. Sistemin en alt tabakası.
- Psikolojik olarak hatalı edilebilir basitleştir.

### Network Security:

Virus: Disardan gelir. içeriye girer.

virus hızlı yayılır, zo algılsızın, zo haldürsün kendini çoğaltılabilir.

Aktif kalıcılaştırır.

Black mail: E-mail eklerileri alma.

Dos atışı: Virus ierde aktif oldukçe sistem servisi kullanamaz.

Intra-corporate (İşbirliği) Birden fazla erkekte bilgi edin

- Virus Assemblyde yaratılır ve hedef chlene dropper

- Virus algılama size'a göre yapılmış. Programı sıkıştırılmış, parmak izine göre, "Kendisin"

Collusion olmasın diye diğer sayfalarla yerleştirmeli

Integrity Checkers: Eleme cihazma var mı?

Behavioral checkers: Sürekli başka prog. erişmeye çalışır  
belteh nifor vs,

Virus avoidance: Sık sık yükleme, antivirus kullan, iyi OS kullan  
email eklenmeli - asma -

Bilgisayar hapat güvenli otomatik bağlat.  
sistemin olduğu hismi hapat.

Internet Worm:

- Kendini yahleyen worm
- worm'un kendisi

Mobile Code:

- Browser trusted değilse sandbox içinde kurulur içinde  
gelmeye çalışırsa hapatır.
- imzaya bakarak güvenli miolar.

$H_1 = \text{hash (Applet)}$   $H_1 = H_2$  ise hoso.  
 $H_2 = \text{decrypt (signature)}$

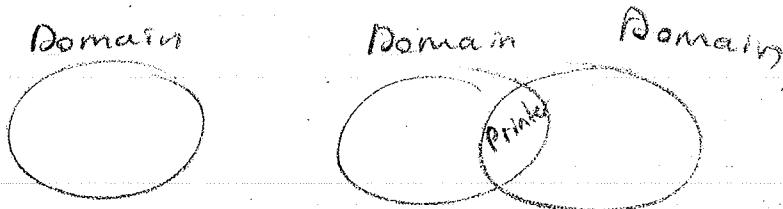
Java Security:

- Derleyici değişkenlerin nötrye kullanmasını öner.

- private eric interi denetler.

- Değişken döngülerinin illegal döngümlerini öner.

Protection Domain:



Protection matrix: Domain ve objeler var.  
Verilen yetenekleri yapabılır.

(Eklentili: Domain 1 den Domain 2 ye gecitlendir,  
Domain 2 den 1 e gecitlendir.)

Access control lists: Her bir objenin erisim kontrol listesi var

Capability: Her bir processin size na göre yetecek listeleri vardır. (kernel servisiye - sistem sagları ile erisir).

Multilevel security: Askeri sisteminde üst yarasanın ana istekini okuyan  
1. seviye process objeyi R-W  
2. seviye " " R-W üst seviyeyi W

Covert channels: Bilgiyi server'a kaydeder. İstikrarlı bir  
sureç bizi izleyen veri gikanır.

## MULTIMEDIA (CHAPTER 13)

Dagitim agı → distributif kentler → ADSL ... turkiyede gecerli  
kabloslu TV çok gecerli değil.

Sifirlamamış HDTV 648 Mbps → 288 GB/hr  
(Uncompressed)  $1 \text{ Mbps} = 10^6 \text{ bits/sec}$

Video dosyasında bir çok bloklesen varır.

### Audio Encoding:

Örneklemme hatası  
Kuantalama hatası (8 seviyede kodlanmış 3 bit)

Bit sayısı arttıkça kalite artar.

PC 8 bit

Ses 16 bit

örnekleri soñu değerlerde dönüştürmeden dalyı hata oluşturur.

PCM → CD lerde sifirlamalı durumu.

### Video Encoding:

NTSC video 640 x 480 Pixel

Her pixel 24 bit.

Y: Parlaklık Aynen görüler, Değiştirilemez.

I, Q: Renk fark işaretleri, Yareya dayanırlar. Gen. aynı şebeke

8x8lik bloklara bölünür.

Blokları olduğu gibi ad. Discrete cosine transform ad.  
sonra quantala (yüksek frekans azaltır)

DCT'yi quantization table'a bası quantized coefficient  
edde et. (kayıtlı kuantalama). Tersine discrete  
cosine adı verilir.

- MPEG tablosunda 35 tane 0 byte ile kodlanır.
  - Günümüzdeki en başarılı sıkıştırma tekniklerinden.
  - JPEG sıkıştırılmış resim.
  - son frame ile üst frame ile farklılar görüntülenir.
  - Ana resim arada bir yerlerir.
  - Burun dışında farklı yerler kesimler gönderilir sadece

## Multimedia Process Scheduling:

- Deadline sağlanamazsa video bozulur.
  - Realtime kontrol sistemlerinde sorunlar olabilir.
  - Prosesörler diğer proseslere bağlı değil. Real time de olabilir.
  - Bir aralıktan bir aralıka prosesler aynı CPU zamanına sahip olur.
  - Periyodik döngülerde deadline yok. Döngülerin deadline'ları farklılıkla tarih bekletilebilir.
  - Süreç halede olabilir

Pull system: istek, blok ardarda gelir, Her blok rəm ayrışdırır.

**Push system:** Bir istek blok ardında gelir başka isteğe gerek yok.

video on Demand: Tüm frameeler hafiflerde değil geri varma  
problemleri稳定性.

\* en çok kullanılanlar dikkate en çokta yerleştirilir.  
(Popüler olanlar)

## File Caching :

- File Caching:  
En popüler olur RAM disk ortası konuları. (Önceli)  
Her bir frame içinde yerleştirir  
Her bir filmin ilk birkaç dəlikası, disklər geri halan,  
yarası beləkən.

Static scheduling: Tek tek frameler hâlinde

Dynamic scheduling: En yarın başlayacak görev.  
seek scan, first job.

## (CHAPTER 14) MULTIPLE PROCESS SYSTEMS :

Yarı iletken teknolojisinde (fiziksel bellekdeki) teknolojik gelişme olmasa clock rate artmaz.

- hah getürdecli sistemler kullanırız

Shared memory; or fah kelte. (Günümüzde) - Kolay programleur  
yalançlığı. Hafızada büyük boyutlu verilerin yer almaması.

Shared memory, interconnect: Her birimde herhangi bellek var. deftik mesaj

Internet: complete system. Magnetic alloy system.

ötek problem:hangisi erişebilir sun belirtme -

## Multiprocessors:

2 veya fazla CPU 1tanı bellek paylaşır.

- CPUlar bus üzerinden bellege yarar. (Biri erişirken diğer bekler)
- CPU + bus + cache : Birinci yarınca diğerinin yaradığı gecisiz
- Her bir işlemci özel bellek, cache var. ortak bellek
- İşlerinden haberleşiyor (dıştıktı + paylaşımı bellek)
- Genel amaç veri tutarlığını sağlanan için bunu yapalar

## Crossbar switch: 8 blok, 8 işlemci var.

- işlemciler aynı anda aynı blok'a erişemez.
- Bir erişimde paylaşır. blokları bekletir.
- Mesaj geçmede paylaşır. file
- Paylaşımında ortak bellege yostur. diğer ortak kullanır.
- Olası birleşme geciklemesi var.

Omega switch: Çok ağanlı switchler kollararak bağıl olabiliyor. tasarımı.

- Mesaj iletilir, bağlantısı kurulur.

## UMA: Uniform Memory Access

tüm bellek ucayına erişim süresi aynı

## NUMA: tüm bellek ucayına erişim farklı farklıdır.

Tüm bir adres ucayı tüm CPU'lara kollarır.  
Uzak bellek ucayına erişim daha uzun. LOAD, STORE ile

Node: Paralel sistem.

256 Node varsa en az 256 işlemci var.

Her bir node da 2 işlemci 8-12 "

Node	Blok	Offset
8	18	6

Directory ortak bellek nerede tutuluyor? Ona göre istek yararla interconnect network ile elde eder.

## OS Types:

Her bir işlemci OS var. Ortak bellek ucayı.

## Master-slave:

(master) OS var.

Programları kopya etmek için diğer OS var. (slave)

## Symmetric:

ortak programları ve OS var. he birinde

## Time sharing:

- 15 tane CPU var.
- Önce A tarifesi 4. isterdiye (önce hangisine yüklenirse oraya yükleneşi olmazsa herhangiye)
- B tarifesi 12 ye
- Hepsi doldu nadir.
- Nedeni mevcutta durdurmaya gerek yok.

## Space sharing:

- nulleri sistemde (CPU) space ister lisi bitince serbest bırakır.
- komşulukları da silikate oluşturur.
- 2 tane boşta var 4 tane istenirse 2 tane lokaler.
- Paralel sistemlerde bu nullanır.

## Gang Scheduling:

Bir grubu aynı veya yakın aralarda tarifelemeye calistır.

- 0. de A grubu
- 1. de B " boşta kalan olursa C
- 2. de D
- 3. de E
- 4. de yine A

- Paralel karma her grupta farklı işlemci Gruplar, aynı anda tarifelenebilir.

## Multi computers:

cluster

cluster of workstation (cows) → en ucuz doğrudan paralel hesaplama yapabilme.

GPU: Yüksek kapasitede vectoriel sistemler.

- Yıldız, torus, hypercube, cube, grid kolları. }
- Daire adı kullanılmış.

Her bir düğümde bir CPU var.

Bu link tabanlıdır haberleşebilir.

- Grid, Torus, Hypercube en çok kollarla paralel hesap.
- Yıldız bilgisayar yapılarında

Store and forward: Bağlantı rehberine verilen kaydedip aktar.

RPC: Uzaktan process programı. Sıfırın (client) RPC isteği yollar. server'a yüklenir lokaller, sonra servisler gelir. Doğru programlama.

- Network geçişmesi çok olmamalı. Heterogen sistemlerde.

(virtual)

Distributed shared memory: Her bir tekni özel bellekt var.

Hardware } sistem

OS

Application → programı yapar.

Remote sistemlerde hardware neredi yapar sonucu getirir.

Distributed shared memory de birden çok CPU aynı blok üzerinde güncellenebilir. Ama bir CPU veri tutarlılığı için diğer CPU'larla同步操作 eder.



consistency: Birinci güncellendiğinde diğer de güncellenebilir.

Aynı sayfa

Load balancing: (Eğer çok değerini, mesela masa szesi)

bir process sayısına göre yüklenmesi  
sonra birası on birası çok yüklenmiş C yeri olur.

Receiver-initiated: Açı yahut old. sayılır. Sunu alabilenin dize söylenir.

\* Sistem tasarımcısı uygun gordüğü algoritma uygular.  
Hibrit sistem olabilir.

Middle ware hizbetsizliği sağlar. Local  $\leftrightarrow$  Global döntüştürme  
farklı sistemler arasında yapmak.

- Connection oriented
- Document Based      Middle ware

## CHAPTER 15 (Operating S. Design):

### Naming:

Name  $\rightarrow$  Directory'i bul  $\rightarrow$  node ile bağlantı.

### Hiding the Hardware:

CPU                          } Bulunca bantır.  
Word length                }

Space Time Trade off  
Bytelari scymale icin program veya macro kullanır.

Her bir pixel icin 24 bit  
" "                8 bit ve palet      } Aynı değer

o gordürün yerine ne yazacaksın.

Plan  $\rightarrow$  code  $\rightarrow$  test modules  $\rightarrow$  Test system  $\rightarrow$  Deployment  
Piyasaya sunulmalı.

- \* Kodu yazan ve test eden farklı kişi
- \* En az 1 yıl destekli

Main program

