1906003052015

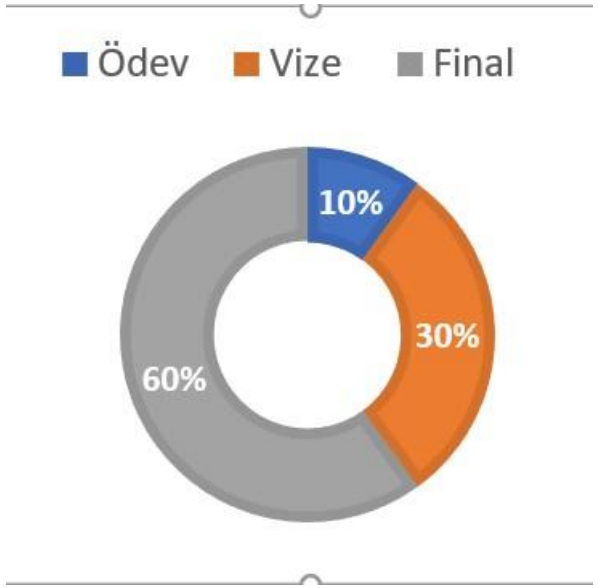
İşletim Sistemleri

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

Giriş

Ders Günü ve Saati:
Çarşamba: 13:00-16:00

- Uygulama Unix (Linux) İşletim sistemi
- Devam zorunluluğu %70
- Uygulamalar C programlama dili üzerinde gerçekleştirilecektir. Öğrencilerden programlama bilgisi beklenmektedir.



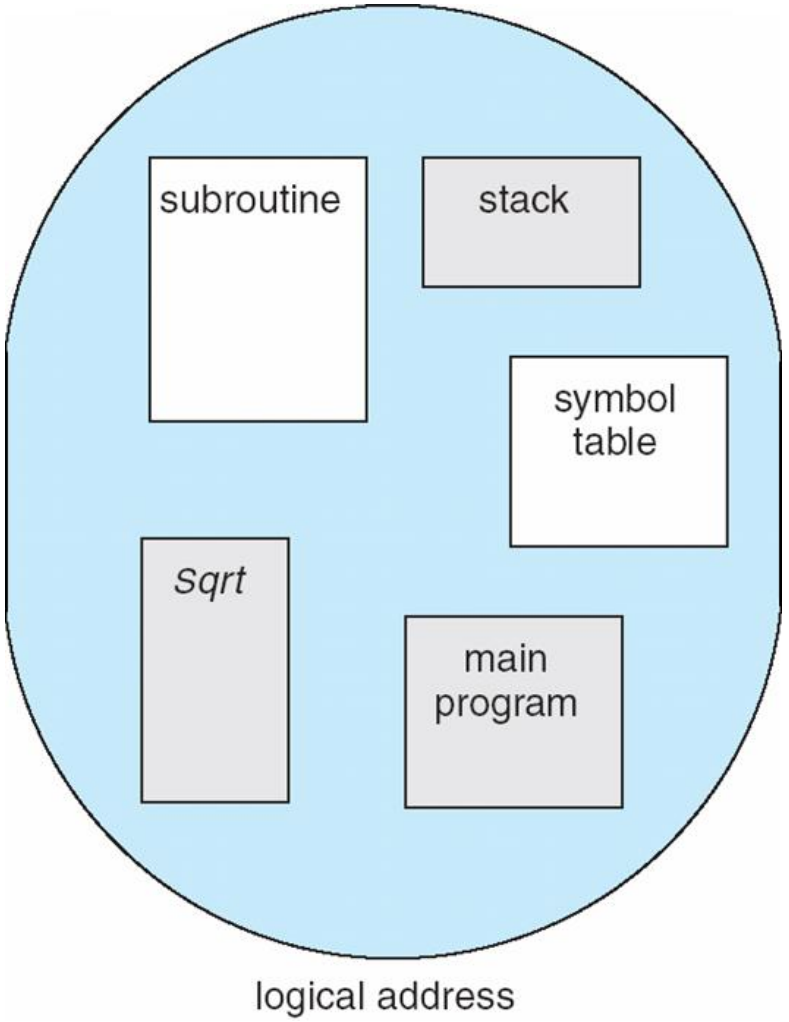
HAFTA	KONULAR
Hafta 1	: İşletim sistemlerine giriş, İşletim sistemi stratejileri
Hafta 2	: Sistem çağrıları
Hafta 3	: Görev, görev yönetimi
Hafta 4	: İplikler
Hafta 5	: İş sıralama algoritmaları
Hafta 6	: Görevler arası iletişim ve senkronizasyon
Hafta 7	: Semaforlar, Monitörler ve uygulamaları
Hafta 8	: Vize
Hafta 9	: Kritik Bölge Problemleri
Hafta 10	: Kilitlenme Problemleri
Hafta 11	: Bellek Yönetimi
Hafta 12	: Sayfalama, Segmentasyon
Hafta 13	: Sanal Bellek
Hafta 14	: Dosya sistemi, erişim ve koruma mekanizmaları, Disk planlaması ve Yönetimi
Hafta 15	: Final

SEGMANTASYON

- Bir kullanıcı programı, programın ve ilişkili verilerinin birkaç bölüme ayrıldığı bölümleme kullanılarak alt bölümlere ayrılabilir.
- Segmentasyon, her işin, ilgili işlevleri yerine getiren parçaları içeren her modül için bir tane olmak üzere, farklı boyutlarda birkaç bölüme ayrıldığı bir bellek yönetimi tekniğidir. Her segment aslında programın farklı bir mantıksal adres alanıdır.
- Sayfalamada olduğu gibi, segmentasyon kullanan bir mantıksal adres, bu durumda bir segment numarası ve bir ofset olmak üzere iki bölümden oluşur.
- Maksimum bölüm uzunluğu olmasına rağmen, tüm programların tüm bölümlerinin aynı uzunlukta olması gerekli değildir.
- Eşit olmayan boyutlu segmentlerin kullanılması nedeniyle, segmentasyon dinamik bölümlemeye benzer.

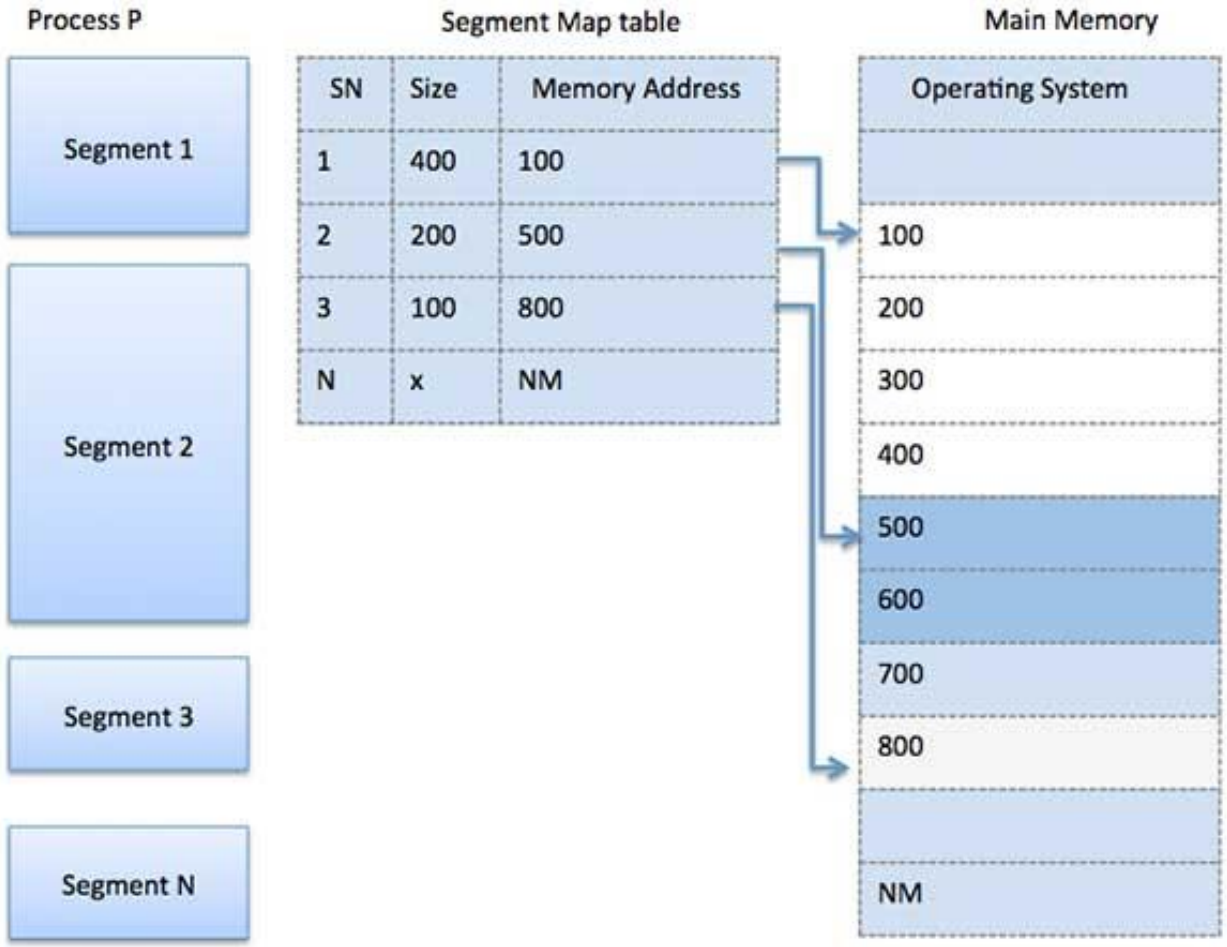
SEGMANTASYON

- Belleğin kullanıcı görünümünü destekleyen bellek yönetimi şemasıdır. Sayfalama programcı için görünmezken, bölümleme genellikle görünürdür ve programları ve verileri düzenlemek için bir kolaylık olarak sağlanır.
- Bir program, segmentlerin bir koleksiyonudur.
- Segment, aşağıdaki gibi mantıksal bir birimdir:
  - ana program
  - prosedür
  - işlev yöntem
  - nesne
  - yerel değişkenler,
  - global değişkenler
  - ortak blok
  - Yığın sembol tablosu
  - diziler



SEGMANTASYON

- Bir program bölümü, programın ana işlevini, yardımcı işlevleri, veri yapılarını vb. içerir.
- İşletim sistemi , her işlem için bir segment haritası tablosu ve ana bellekteki segment numaraları, boyutları ve karşılık gelen bellek konumlarıyla birlikte boş bellek bloklarının bir listesini tutar.
- Her segment için tablo, segmentin başlangıç adresini ve segmentin uzunluğunu saklar.
- Bir bellek konumuna yapılan bir referans, bir segmenti ve bir ofseti tanımlayan bir değeri içerir.



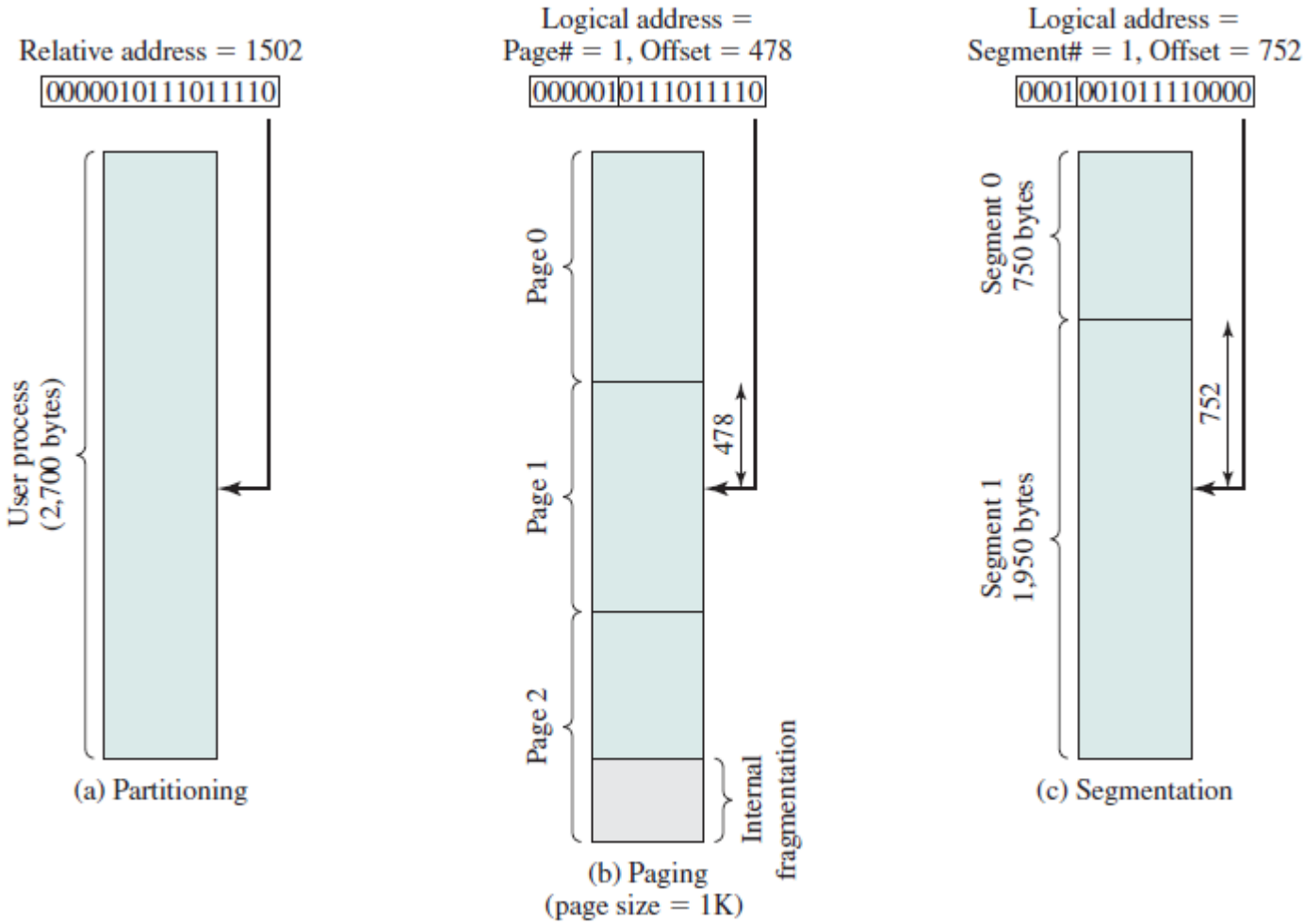
SEGMANTASYON

- Eşit olmayan büyüklükteki segmentlerin bir başka sonucu da, mantıksal adresler ile fiziksel adresler arasında basit bir ilişki olmamasıdır.
- Sayfalamaya benzer şekilde, basit bir bölümleme şeması, her işlem için bir bölüm tablosunu ve ana bellekteki boş blokların bir listesini kullanır.
- Her bölüm tablosu girişi, karşılık gelen bölümün ana belleğinde başlangıç adresini vermek zorunda kalacaktır.
- Geçersiz adreslerin kullanılmadığından emin olmak için giriş ayrıca segmentin uzunluğunu da sağlamalıdır.
- Bir işlem Çalışıyor durumuna girdiğinde, segment tablosunun adresi, bellek yönetim donanımı tarafından kullanılan özel bir kayıt defterine yüklenir

SEGMANTASYON

- CPU, iki bölümden oluşan bir mantıksal adres oluşturur:
  - Segment numarası
  - Ofset
- En soldaki $\boldsymbol{n}$ bitin segment numarası ve en sağdaki $\boldsymbol{m}$ bitin ofset olduğu $\boldsymbol{n} + \boldsymbol{m}$ bitlik bir adres düşünün.
- Örneğimizde, $n = 4$ ve $m = 12$. Böylece maksimum segment boyutu

$2^{12} = 4096$ 'dır.

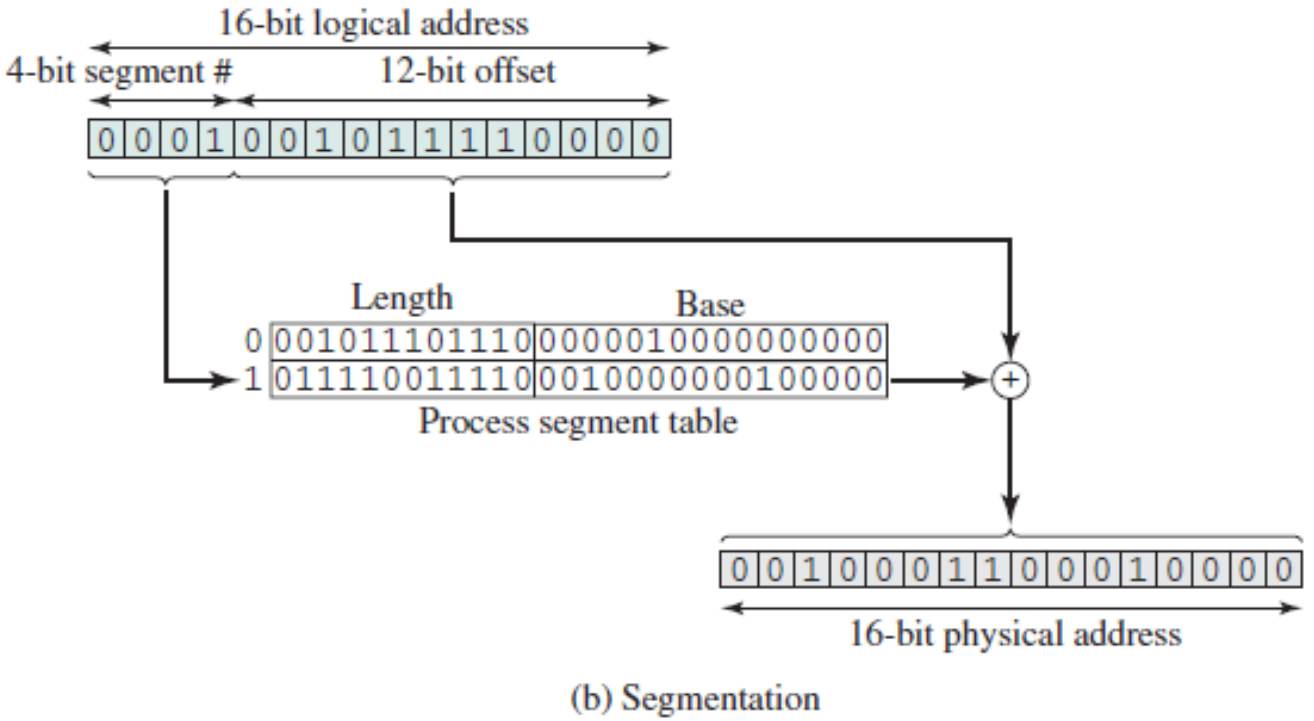


SEGMANTASYON

- Adres çevirisi için aşağıdaki adımlar gereklidir:
  - Mantıksal adresin en soldaki n biti olarak segment numarasını çıkarın.
  - Segmentin başlangıç fiziksel adresini bulmak için işlem segment tablosunda bir dizin olarak segment numarasını kullanın.
  - En sağdaki m bitlerde ifade edilen ofseti segmentin uzunluğuyla karşılaştırın. Ofset, uzunluktan büyük veya eşitse, adres geçersizdir.
  - İstenen fiziksel adres, segmentin başlangıç fiziksel adresi ile ofsetin toplamıdır.

SEGMANTASYON

- Segment numarası 1, ofset 752 olan mantıksal 000100101111000 adresimiz var. Bu segmentin 0010000000100000 fiziksel adresinden başlayarak ana bellekte bulunduğunu varsayalım. O zaman fiziksel adres
- 0010000000100000 + 001011110000 = 0010001100010000

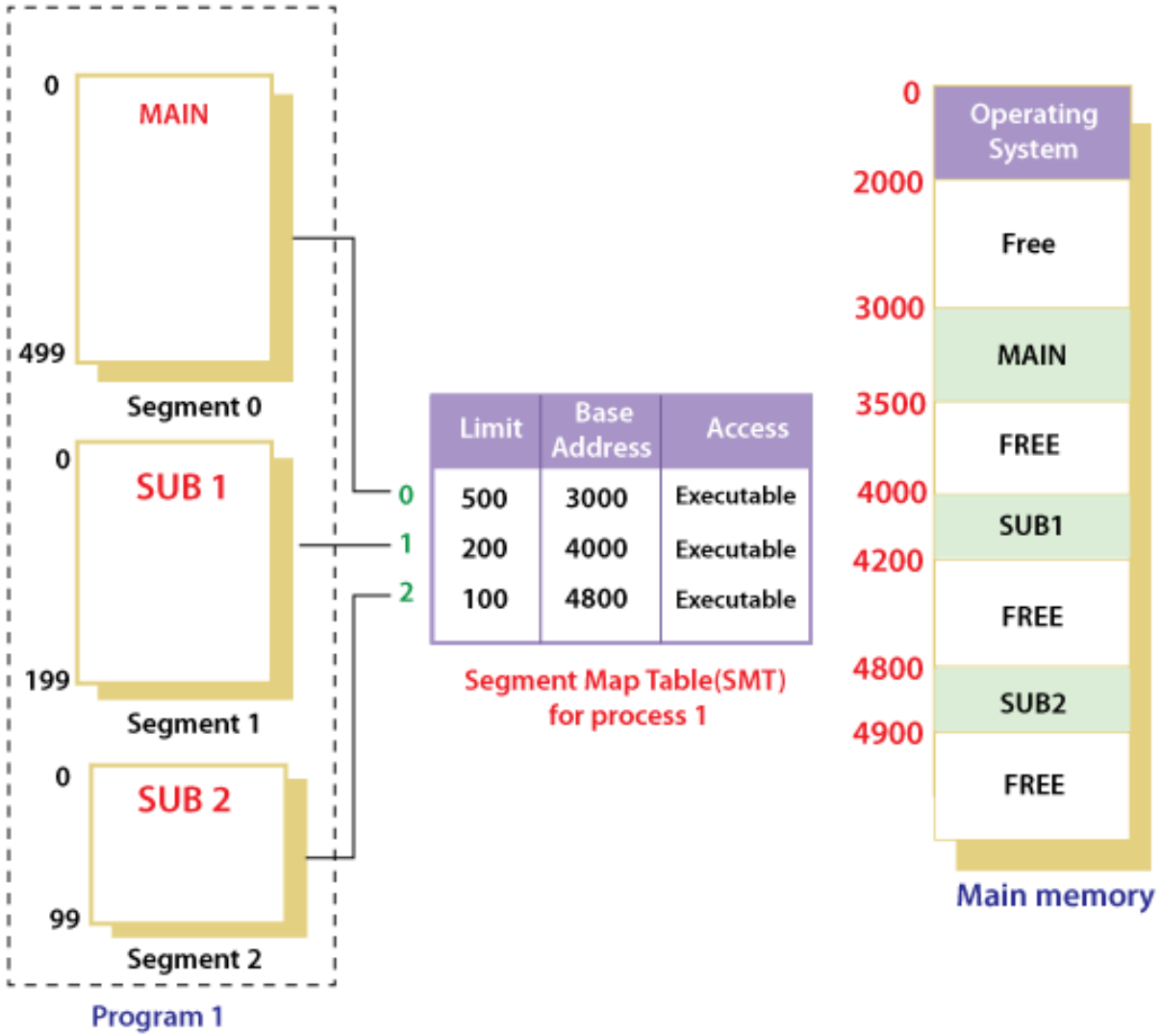


(b) Segmentation

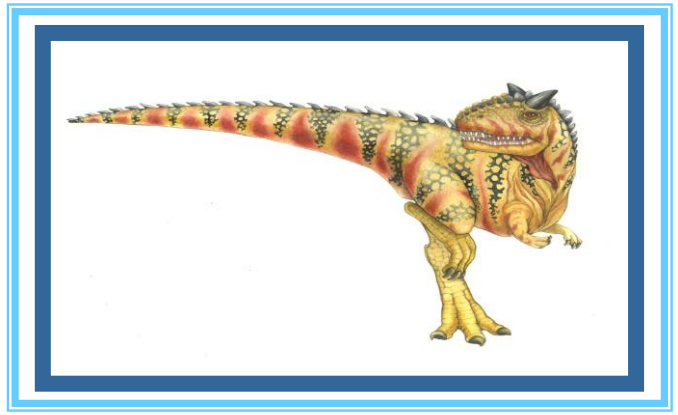
SEGMANTASYON

- Bölüm numarası için 4 bit ve bölüm ofseti için 12 bit ile 16 bitlik bir adresin kullanıldığını varsayalım, bu nedenle maksimum bölüm boyutu 4096 ve hakemlik yapılabilecek maksimum bölüm sayısı 16'dır.
- Bir program belleğe yüklendiğinde, bölümleme sistemi işlemin ilk bölümünü tutacak kadar büyük bir alanı bulmaya çalışır, bellek yöneticisi tarafından tutulan boş listeden alan bilgisi elde edilir. Daha sonra diğer segmentler için yer bulmaya çalışır. Tüm bölümler için yeterli alan bulunduğunda, bunları kendi alanlarına yükler.
- İşletim sistemi ayrıca her program için bir segment haritası tablosu oluşturur.
- Segment harita tabloları ve donanım yardımı ile işletim sistemi, bir programın yürütülmesi sırasında mantıksal bir adresi kolayca fiziksel adrese çevirebilir.

SEGMANTASYON

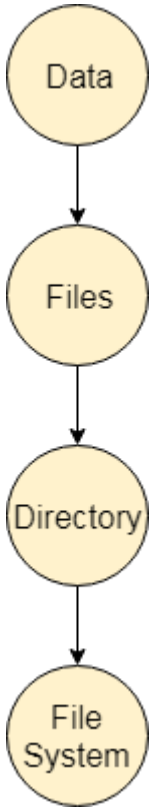


DOSYA SİSTEMİ



Dosya Sistemi

Kullanıcı açısından, bir işletim sisteminin en önemli parçalarından biri dosya sistemidir. Dosya sistemi, tipik olarak ikincil depolama ile ilişkili kaynak soyutlamalarını sağlar. Dosya sistemi, kullanıcıların aşağıdakiler gibi istenen özelliklere sahip dosyalar adı verilen veri koleksiyonları oluşturmasına izin verir:



- **Uzun süreli mevcudiyet**: Dosyalar diskte veya başka bir ikincil depoda depolanır ve bir kullanıcı oturumu kapattığında kaybolmazlar.
- **İşlemler arasında paylaşılabilir**: Dosyaların adları vardır ve kontrollü paylaşıma izin veren ilişkili erişim izinlerine sahip olabilir.
- **Yapı**: Dosya sistemine bağlı olarak, bir dosya belirli uygulamalar için uygun bir dahili yapıya sahip olabilir. Ek olarak, dosyalar arasındaki ilişkileri yansıtmak için dosyalar hiyerarşik veya daha karmaşık yapıda düzenlenebilir.

Dosyanın Nitelikleri

1. İsim

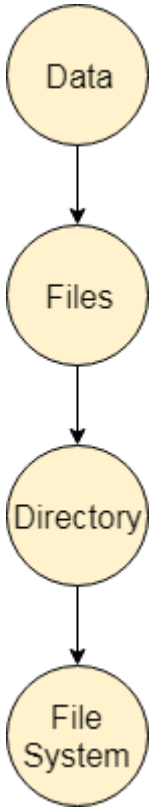
Her dosya, dosya sisteminde dosyanın tanındığı bir ad taşır. Bir dizin aynı ada sahip iki dosyaya sahip olamaz.

2. Tanımlayıcı

Her Dosyanın adıyla birlikte, dosyanın türünü tanımlayan kendi uzantısı vardır. Örneğin, bir metin dosyası .txt uzantısına sahiptir , Bir video dosyası .mp4 uzantısına sahip olabilir

3.Tip

Bir Dosya Sisteminde, Dosyalar video dosyaları, ses dosyaları, metin dosyaları, yürütülebilir dosyalar vb. gibi farklı türlerde sınıflandırılır.



Dosyanın Nitelikleri

4.Konum

Dosya Sisteminde, dosyaların saklanabileceği birkaç konum vardır. Her dosya, konumunu özniteliği olarak taşır.

5.Boyut

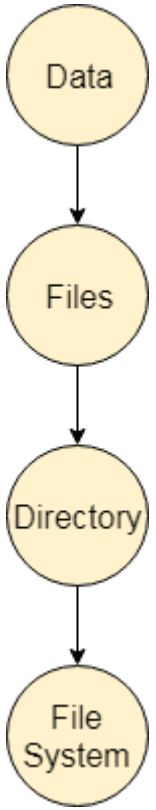
Dosyanın Boyutu en önemli özelliklerinden biridir. Dosyanın boyutu ile, dosyanın bellekte edindiği bayt sayısını kastediyoruz.

6. Koruma

Bilgisayarın Yöneticisi, farklı dosyalar için farklı korumalar isteyebilir. Bu nedenle, her dosya farklı Kullanıcı gruplarına kendi izinlerini taşır.

7.Saat ve Tarih

Her dosya, dosyanın en son değiştirildiği saat ve tarihi içeren bir zaman damgası taşır



File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Structure

- Dosya türleri, dosyanın iç yapısını belirtmek için de kullanılabilir. Kaynak ve nesne dosyaları, onları okuyan programların beklentilerine uygun yapılara sahiptir.
- Ayrıca, belirli dosyaların işletim sistemi tarafından anlaşılan gerekli bir yapıya uyması gerekir. Örneğin, işletim sistemi, yürütülebilir bir dosyanın, dosyanın bellekte nereye yükleneceğini ve ilk talimatın konumunun ne olduğunu belirleyebilmesi için belirli bir yapıya sahip olmasını gerektirir.
- Bazı işletim sistemleri, bu fikri, dosyaları bu yapılarla işlemek için özel işlem kümeleriyle birlikte, sistem tarafından desteklenen bir dizi dosya yapısına genişletir.

File Structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
  - Operating system
  - Program

Dosya Erişim Yöntemleri

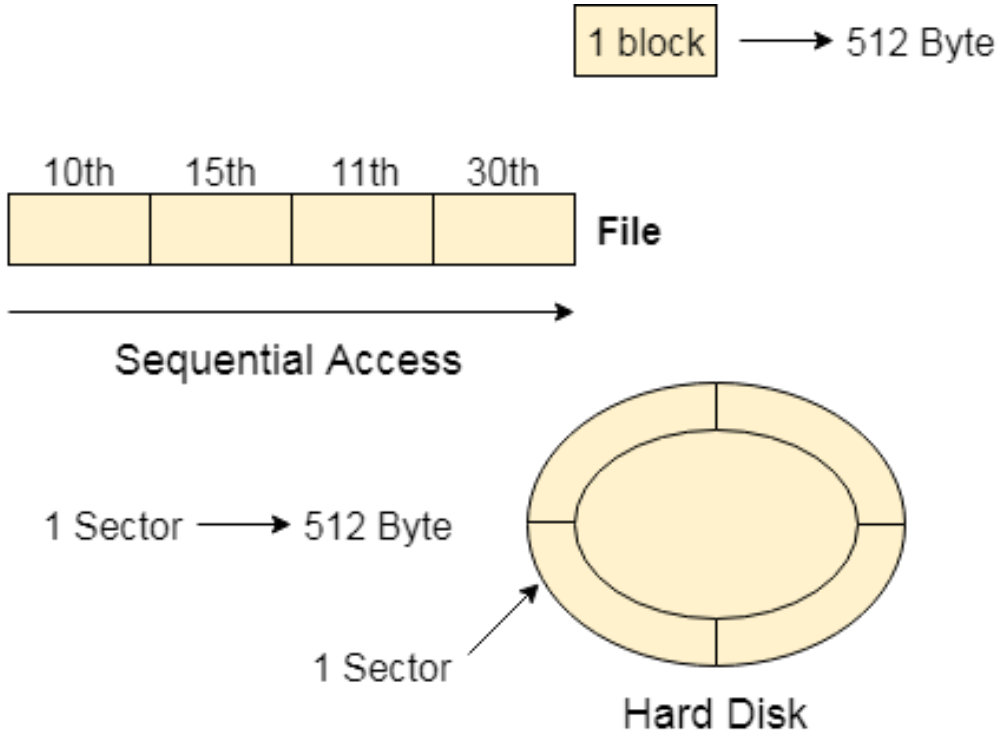
Dosya erişim mekanizması, bir dosyanın kayıtlarına nasıl erişilebileceğini ifade eder. Dosyalara erişmenin birkaç yolu vardır -

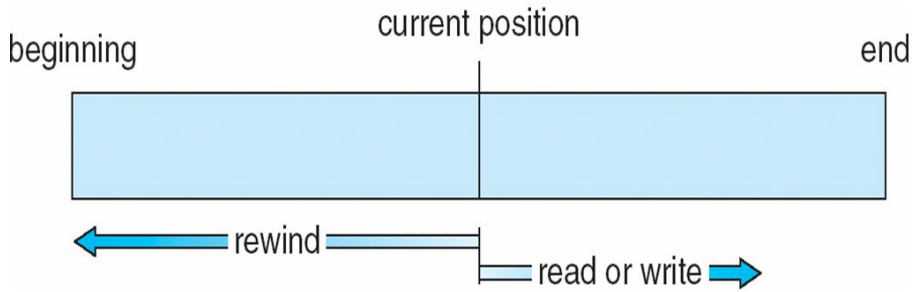
- sıralı erişim
- Doğrudan/Rastgele erişim
- Dizine alınmış sıralı erişim

Sıralı Erişim

En basit erişim yöntemi sıralı erişimdir. Dosyadaki bilgiler sırayla, birbiri ardına kayıt işlenir. Bu erişim modu açık ara en yaygın olanıdır; örneğin, editörler ve derleyiciler dosyalara genellikle bu şekilde erişir.

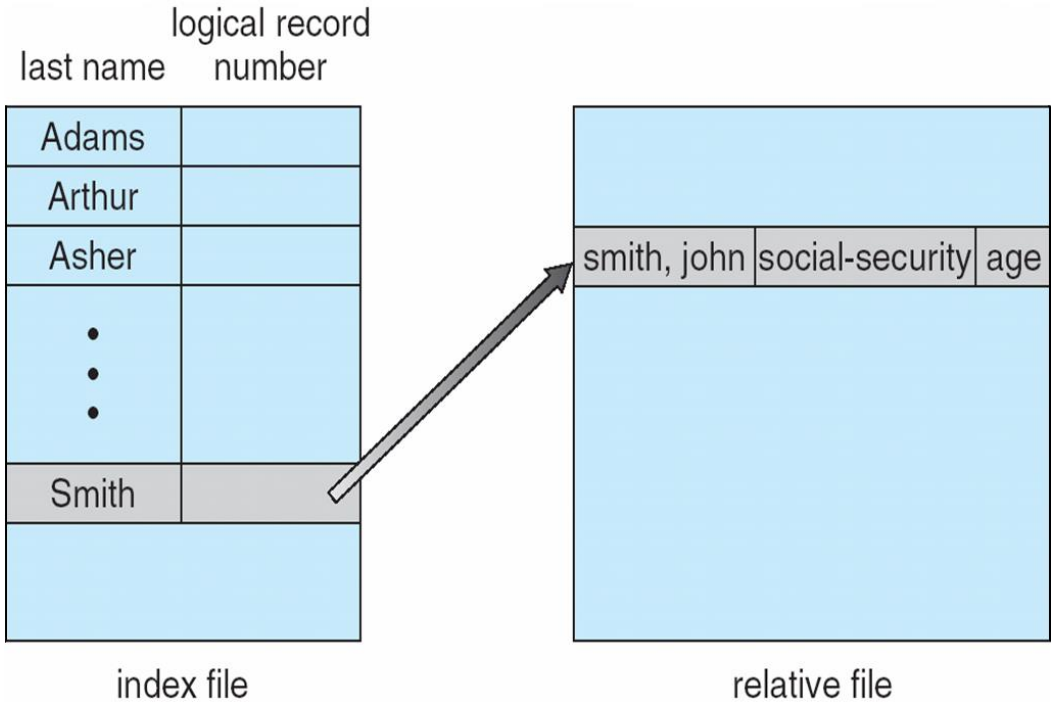
Sıralı erişimde, işletim sistemi dosyayı kelime kelime okur. Başlangıçta dosyanın temel adresini gösteren bir işaretçi korunur. Kullanıcı dosyanın ilk kelimesini okumak isterse, işaretçi o kelimeyi kullanıcıya sağlar ve değerini 1 kelime arttırır. Bu işlem dosyanın sonuna kadar devam ede





Doğrudan Erişim

Diğer bir yöntem ise doğrudan erişimdir (veya göreli erişim). Burada bir dosya, programların kayıtları belirli bir sıra olmaksızın hızlı bir şekilde okumasına ve yazmasına izin veren sabit uzunluklu mantıksal kayıtlardan oluşur. Diskler herhangi bir dosya bloğuna rastgele erişime izin verdiğinden, doğrudan erişim yöntemi bir dosyanın disk modeline dayanır. Doğrudan erişim için dosya, numaralandırılmış bloklar veya kayıtlar dizisi olarak görülür.



Dosya Sistemi Operasyonları

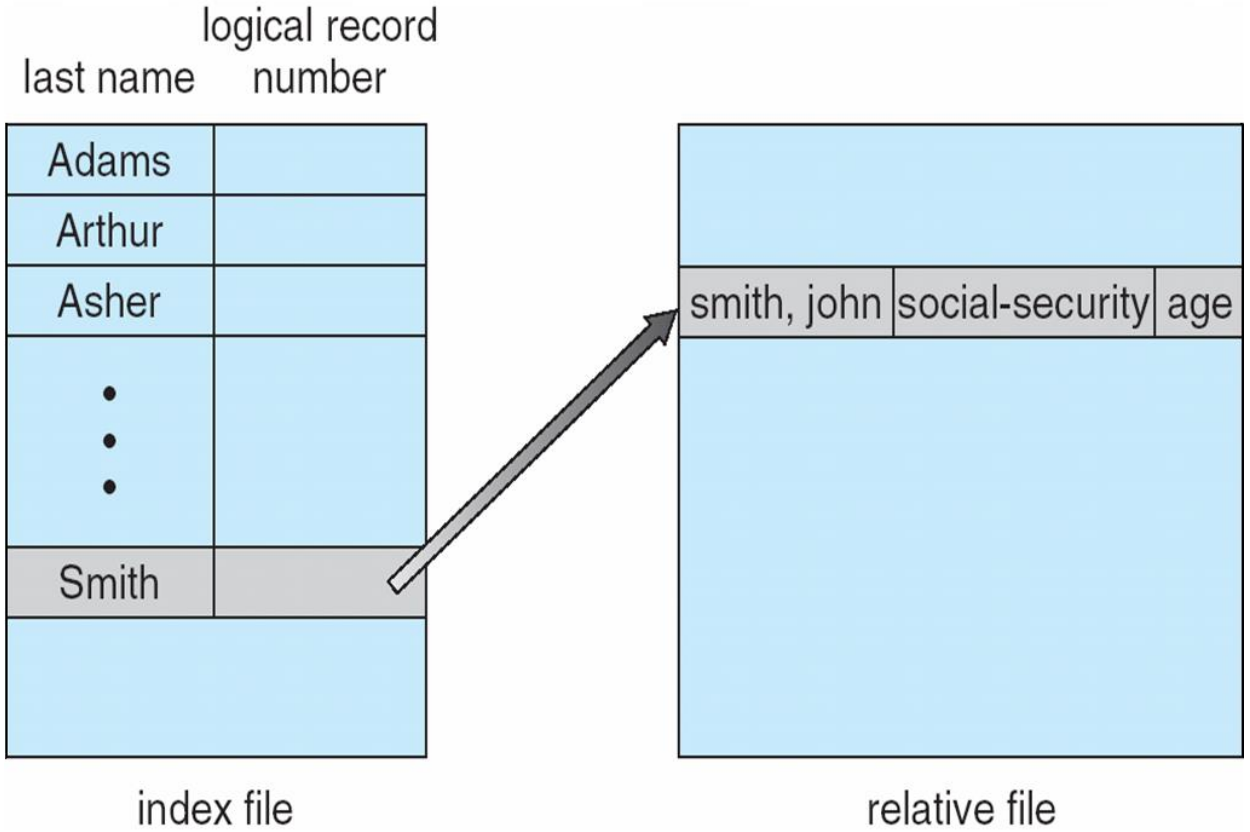
Herhangi bir dosya sistemi, yalnızca dosya olarak düzenlenen verileri depolamak için bir araç değil, aynı zamanda dosyalar üzerinde gerçekleştirilebilecek bir işlevler koleksiyonu sağlar. Tipik işlemler:

- Create: A new file is defined and positioned within the structure of files.
- Delete: A file is removed from the file structure and destroyed.
- Open: An existing file is declared to be “opened” by a process, allowing the process to perform functions on the file.
- Close: The file is closed with respect to a process, so that the process no longer may perform functions on the file, until the process opens the file again.
- Read: A process reads all or a portion of the data in a file.
- Write: A process updates a file, either by adding new data that expands the size of the file or by changing the values of existing data items in the file

Other Access Methods

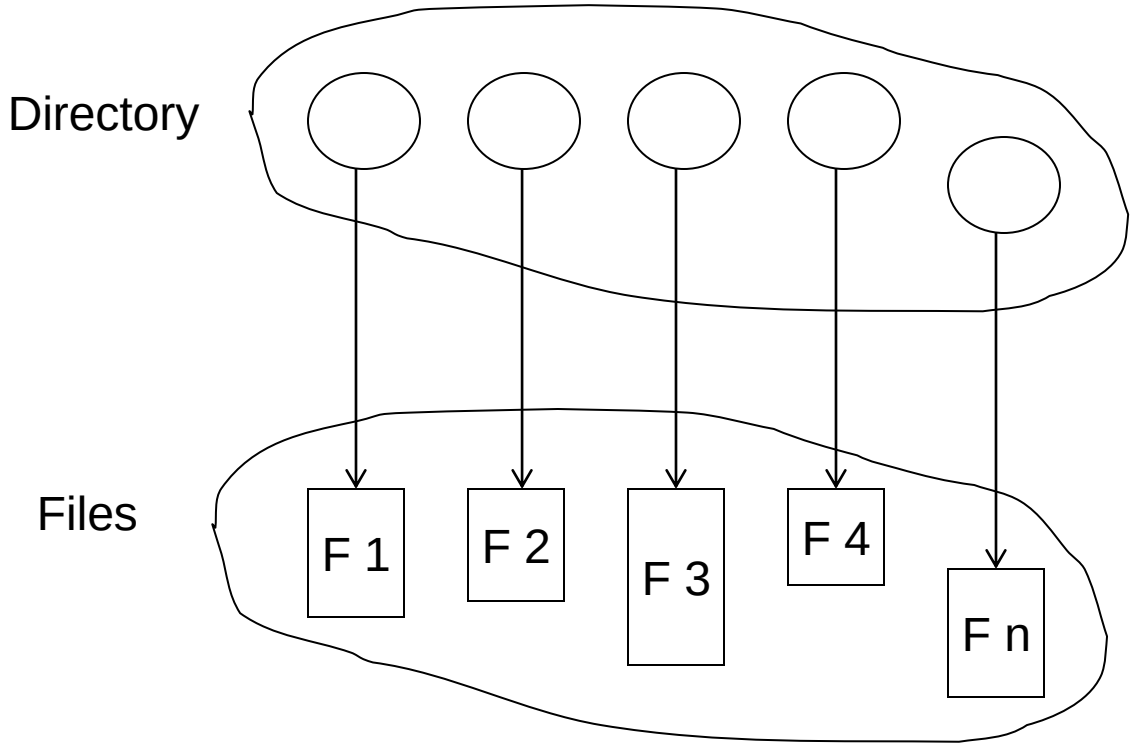
- Can be built on top of base methods
- General involve creation of an index for the file
- Keep index in memory for fast determination of location of data to be operated on (consider UPC code plus record of data about that item)
- If too large, index (in memory) of the index (on disk)
- IBM indexed sequential-access method (ISAM)
  - Small master index, points to disk blocks of secondary index
  - File kept sorted on a defined key
  - All done by the OS
- VMS operating system provides index and relative files as another example (see next slide)

Example of Index and Relative Files



Directory Structure

- A collection of nodes containing information about all files

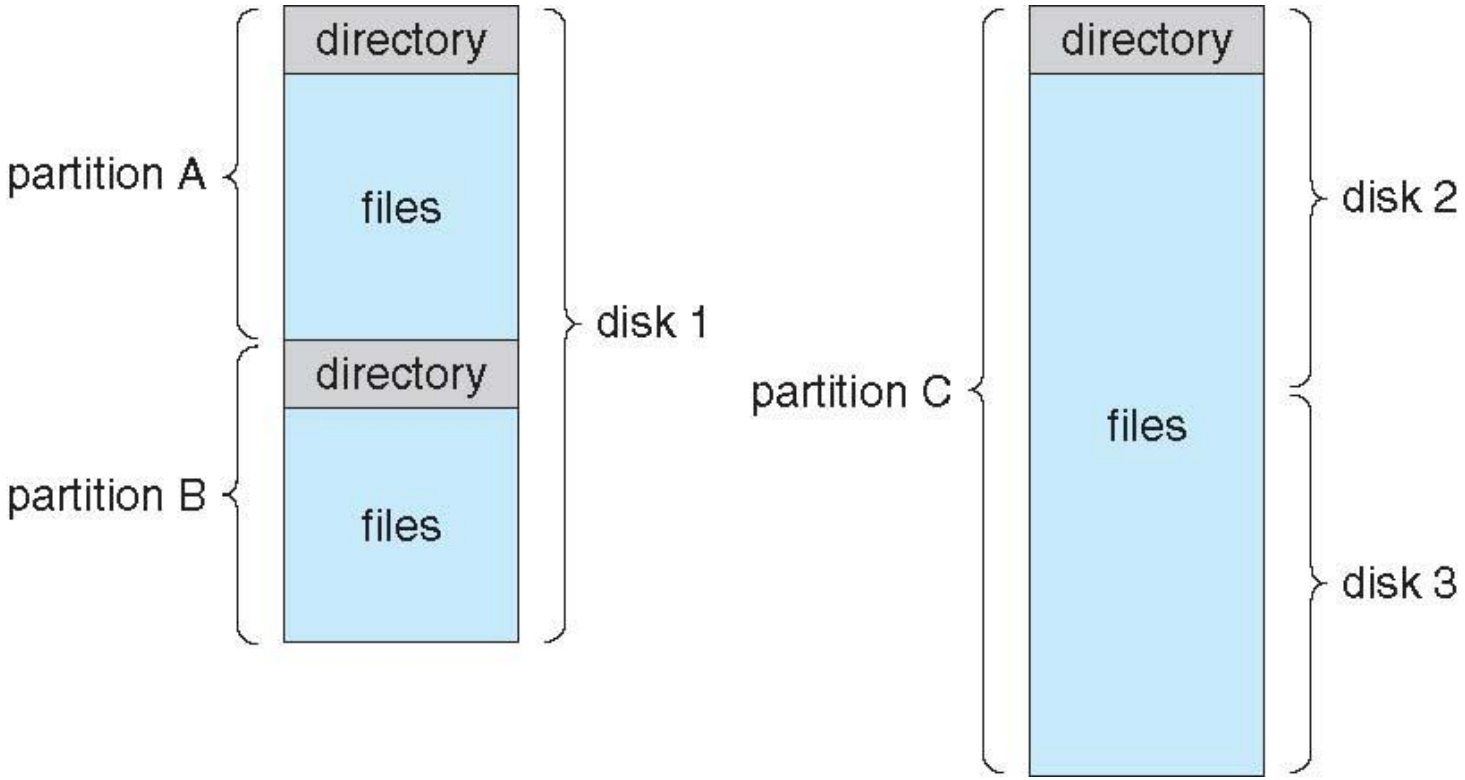


Both the directory structure and the files reside on disk

Disk Structure

- Disk can be subdivided into **partitions**
- Disks or partitions can be **RAID** protected against failure
- Disk or partition can be used **raw** – without a file system, or **formatted** with a file system
- Partitions also known as minidisks, slices
- Entity containing file system known as a **volume**
- Each volume containing file system also tracks that file system's info in **device directory** or **volume table of contents**
- As well as **general-purpose file systems** there are many **special-purpose file systems**, frequently all within the same operating system or computer

A Typical File-system Organization



Types of File Systems

- We mostly talk of general-purpose file systems
- But systems frequently have may file systems, some general- and some special-purpose
- Consider Solaris has
  - tmpfs – memory-based volatile FS for fast, temporary I/O
  - objfs – interface into kernel memory to get kernel symbols for debugging
  - ctfs – contract file system for managing daemons
  - lofs – loopback file system allows one FS to be accessed in place of another
  - procfs – kernel interface to process structures
  - ufs, zfs – general purpose file systems

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

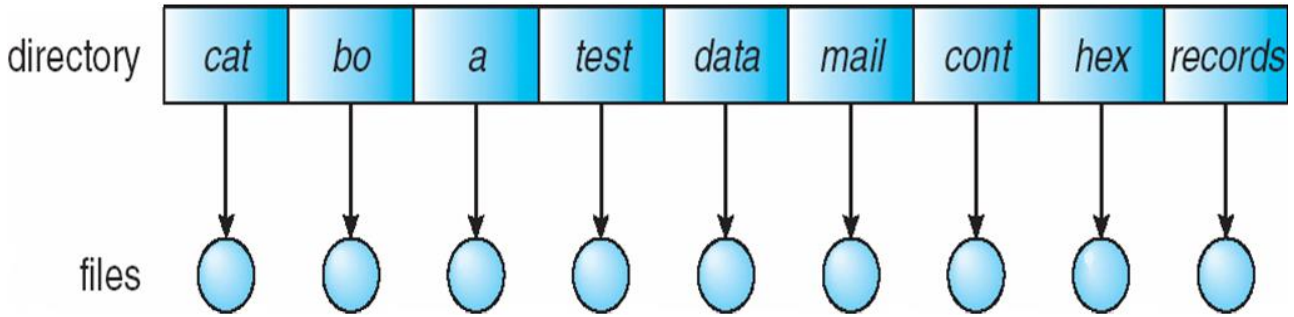
Directory Organization

The directory is organized logically to obtain

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Single-Level Directory

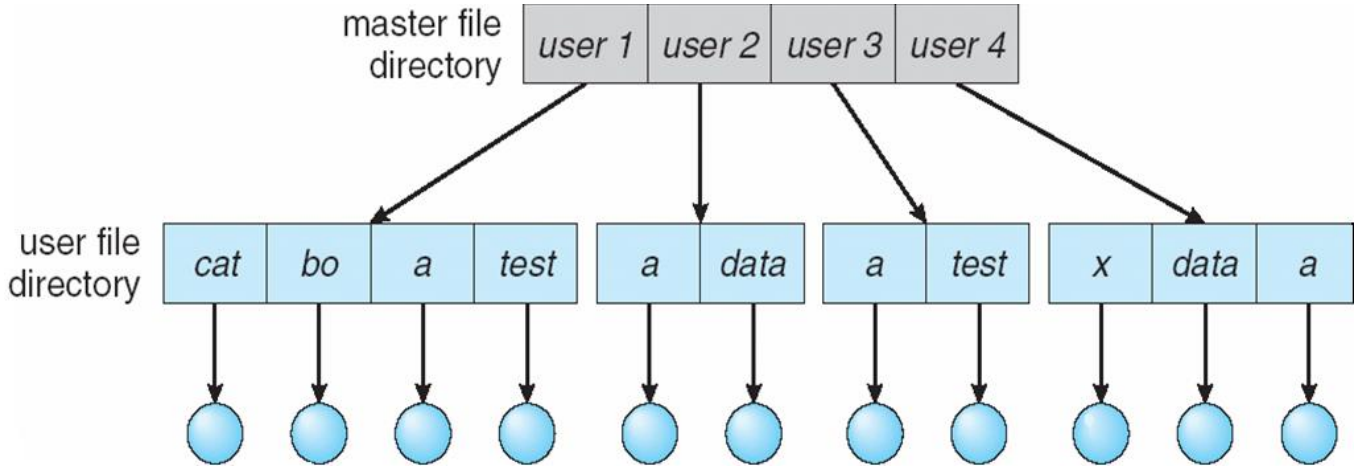
- A single directory for all users



- Naming problem
- Grouping problem

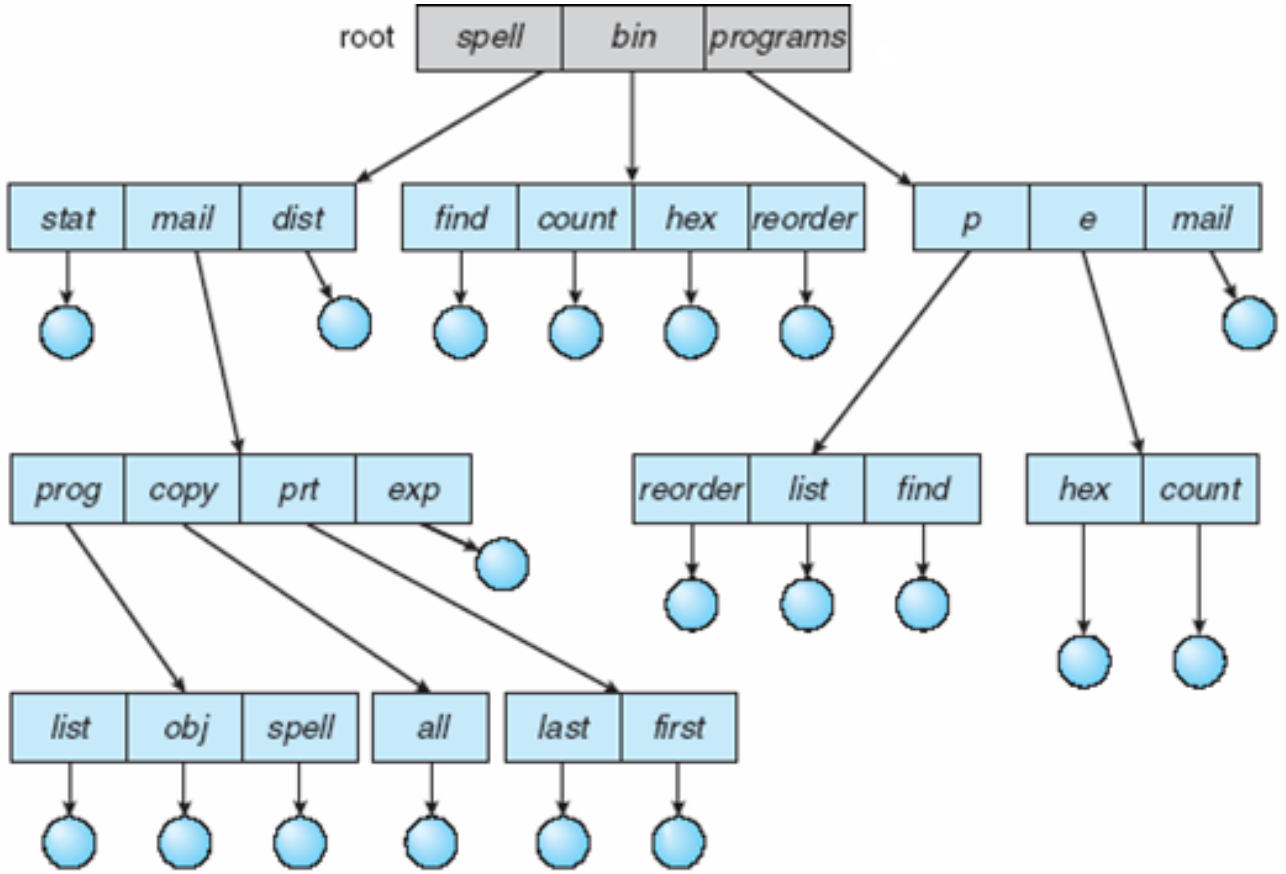
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- Efficient searching
- Grouping Capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
  - `type list`

Tree-Structured Directories (Cont)

- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

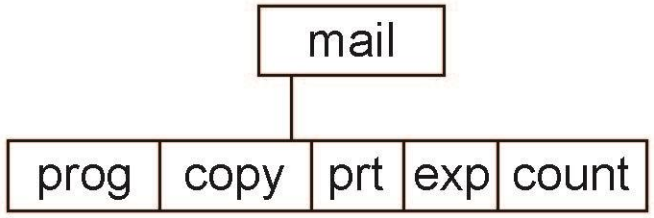
```
rm <file-name>
```

- Creating a new subdirectory is done in current directory

```
mkdir <dir-name>
```

Example: if in current directory **`/mail`**

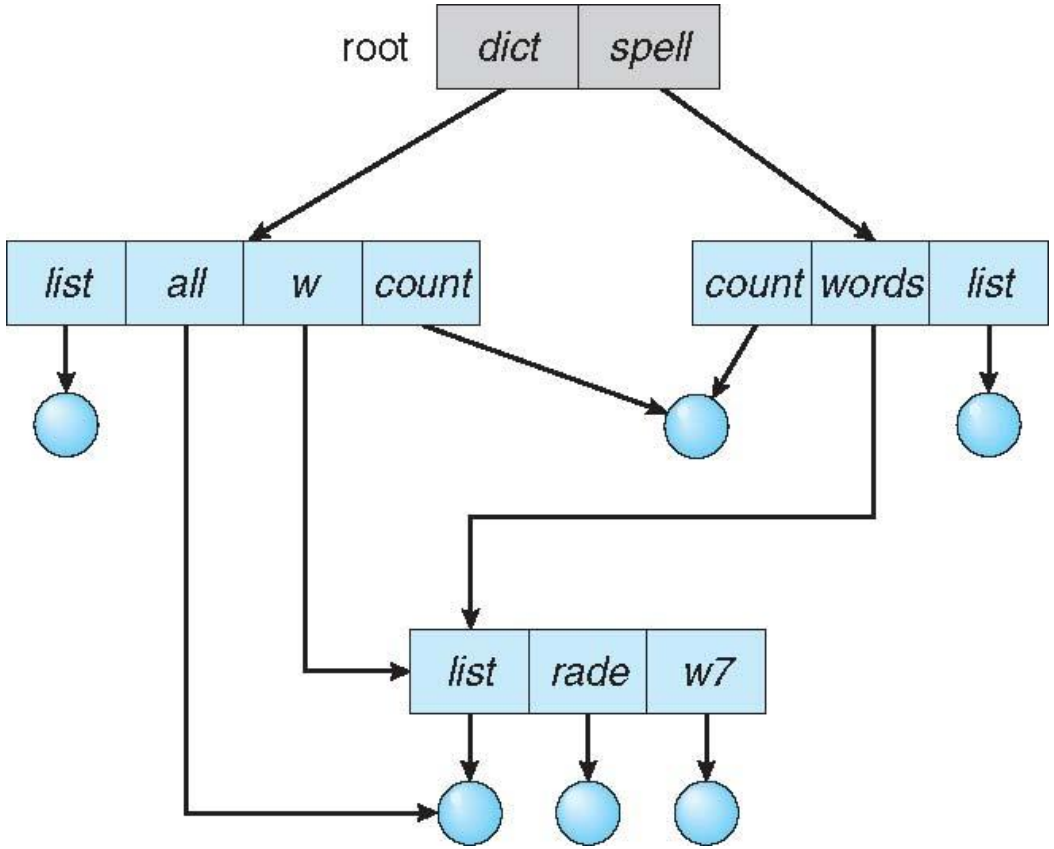
```
mkdir count
```



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

- Have shared subdirectories and files

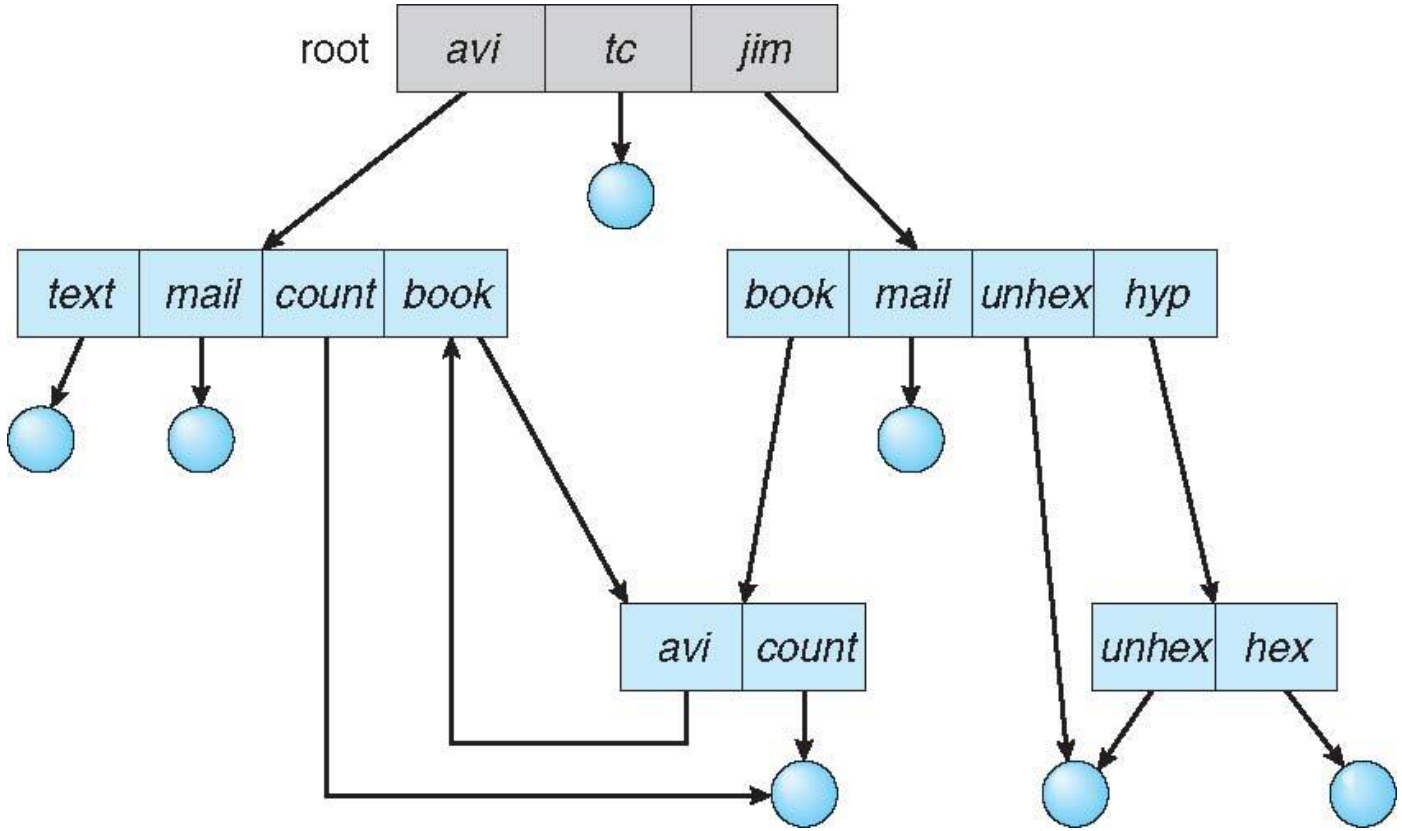


Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If ***dict*** deletes ***list*** ⇒ dangling pointer

  Solutions:
  - Backpointers, so we can delete all pointers
    Variable size records a problem
  - Backpointers using a daisy chain organization
  - Entry-hold-count solution
- New directory entry type
  - **Link** – another name (pointer) to an existing file
  - **Resolve the link** – follow pointer to locate the file

General Graph Directory

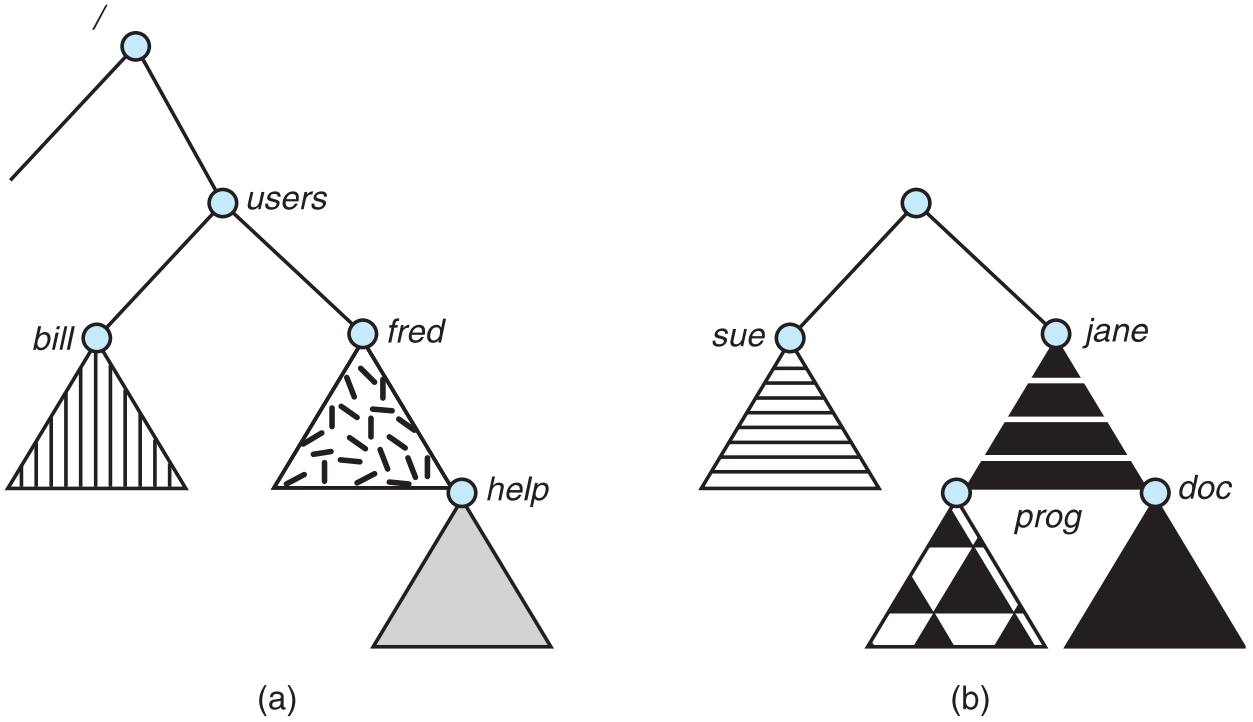


General Graph Directory (Cont.)

- How do we guarantee no cycles?
  - Allow only links to file not subdirectories
  - **Garbage collection**
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system (i.e., Fig. 11-11(b)) is mounted at a **mount point**



Mount Point