Chapter 11: File System Implementation

Chapter 11: File System Implementation

- File-System Structure
- File-System Implementation
- Directory Implementation
- Allocation Methods
- Free-Space Management
- Efficiency and Performance
- Recovery
- Log-Structured File Systems
- NFS
- Example: WAFL File System

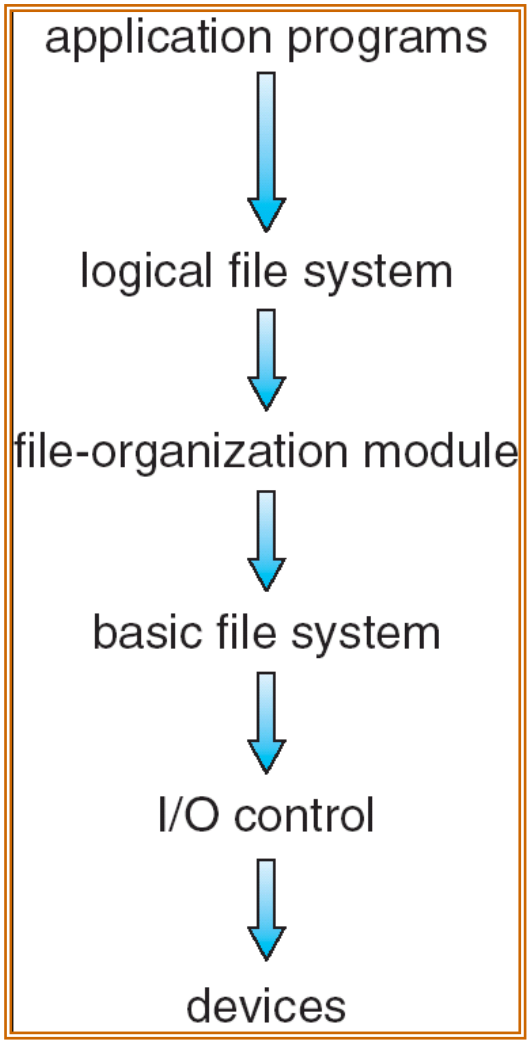
Objectives

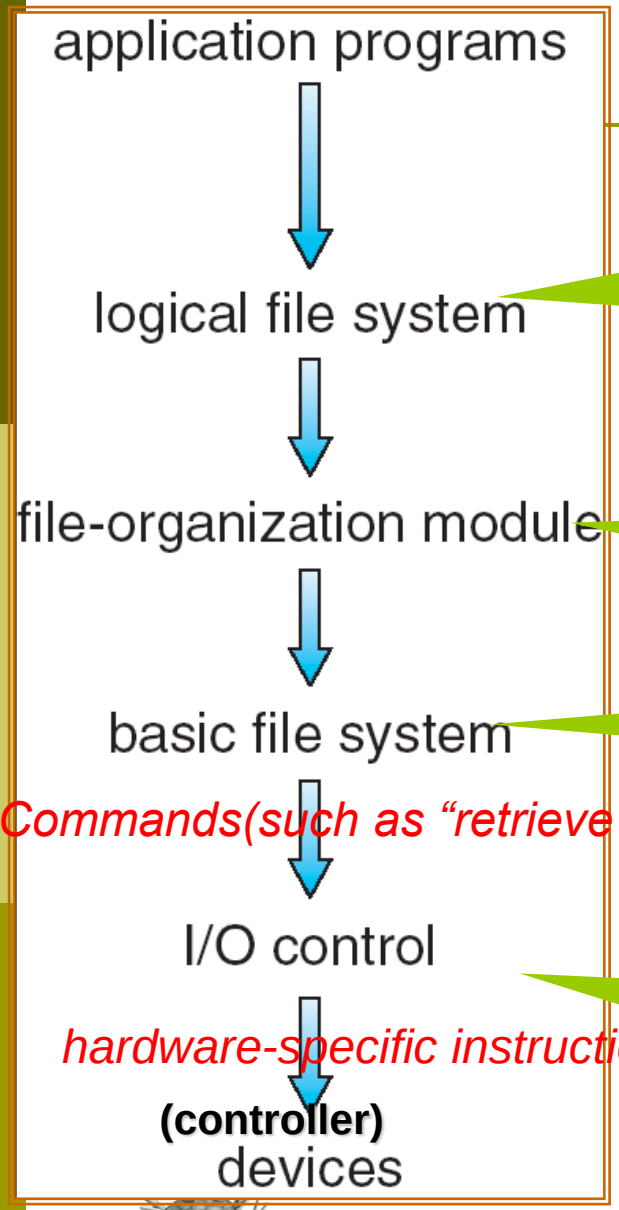
- To describe the details of implementing local file systems and directory structures
- To describe the implementation of remote file systems
- To discuss block allocation and free-block algorithms and trade-offs

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
- File system organized into layers
- **File control block** – storage structure consisting of information about a file

Layered File System





Manages metadata information. 1.manages the directory structure; 2.maintains file structure via **file control block**(**FCB**); 3.protection and security.

1.Translate logical block address to physical block address; 2.free space manager.

Issue generic commands to the appropriate device driver

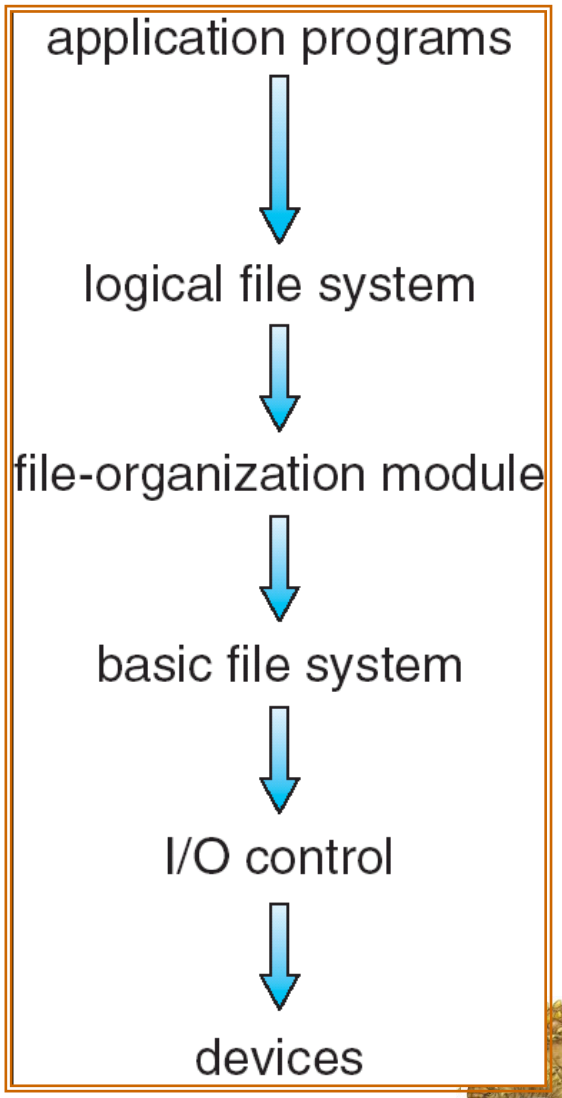
device drivers and ***interrupt handlers*** to transfer information between the main memory and the disk system.

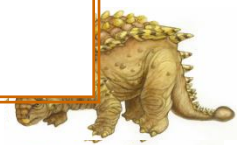
Layered File System

Layered File System

- **设备层**
  - 硬件（磁盘、磁盘控制器等）
  - 处理来自 I/O 控制层的指令

    例如：
    - 寻找特定的磁道
    - 读扇区
    - 写扇区



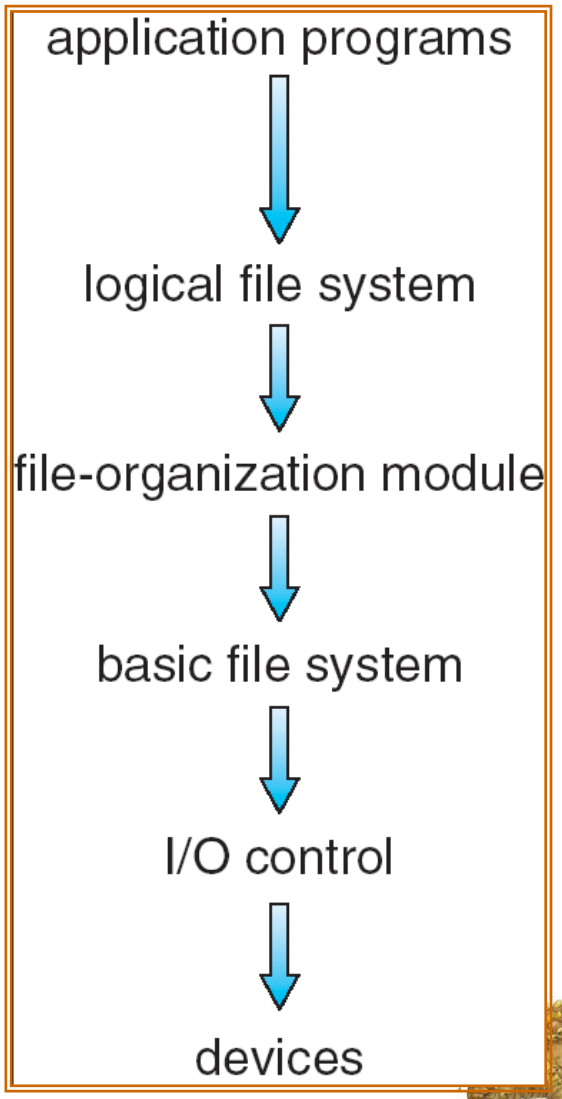


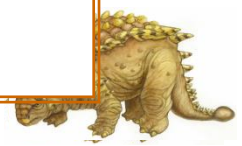
Layered File System

- **I/O 控制层**
  - 设备驱动程序;
  - 告诉设备控制器采取何动作;
  - 翻译来自基本文件系统层的命令以使硬件能够理解;

    in: 从设备 x 读物理块 722

    out: 寻找磁道 8，从磁头 6 读扇区 13

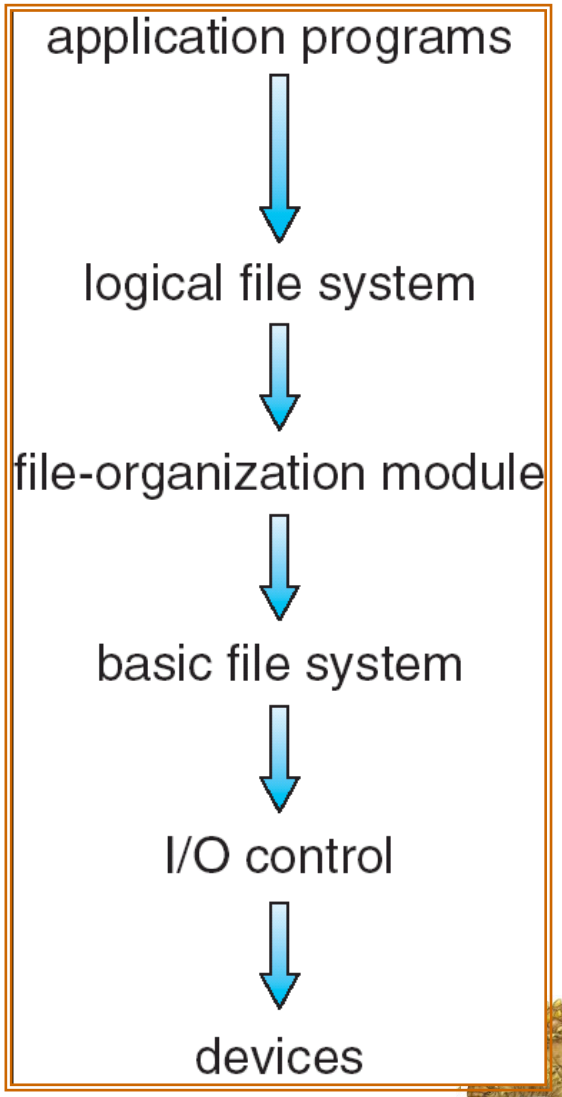


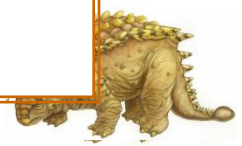


Layered File System

- **基本文件系统层**

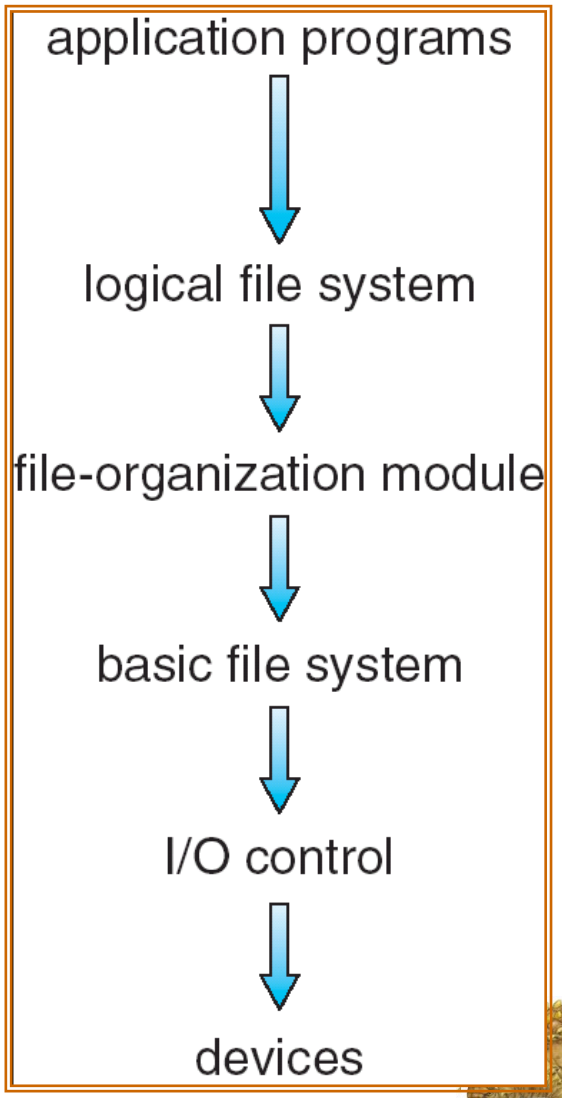
  向合适的设备驱动程序发送一般命令就可对磁盘上的物理块进行读写。

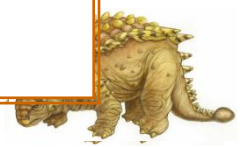




Layered File System

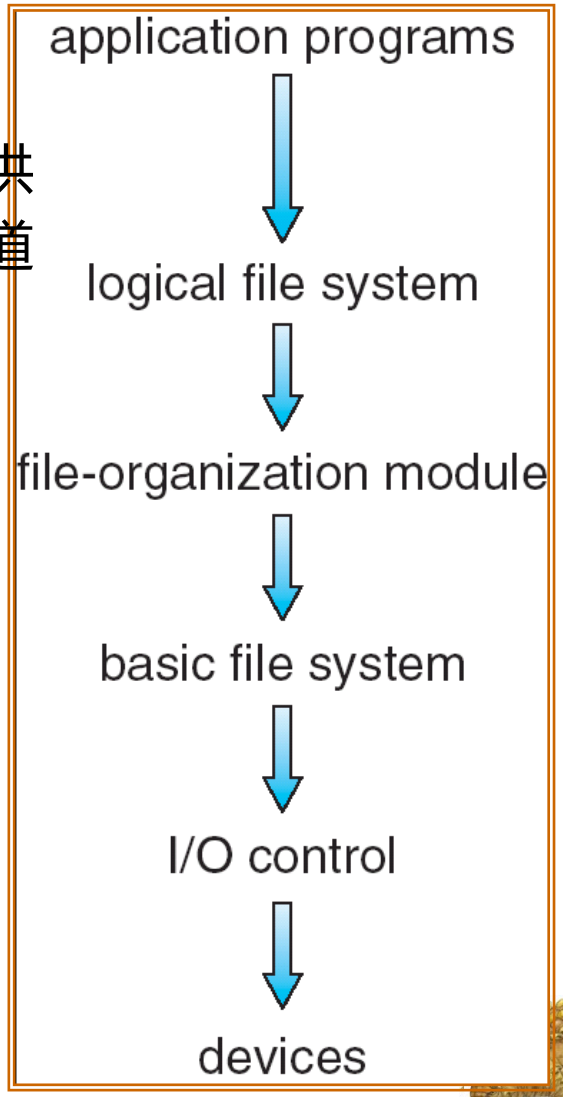
- **文件组织模块层**
  - 知道文件及其逻辑块和物理块
  - 每个文件存放在一系列的磁盘块上，每块有一个逻辑块号(从 0 到 *N*)
  - 这个模块把逻辑块号翻译成物理块地址
  - 维护磁盘上的空闲空间。

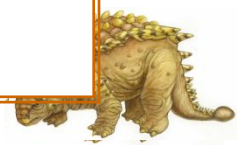




Layered File System

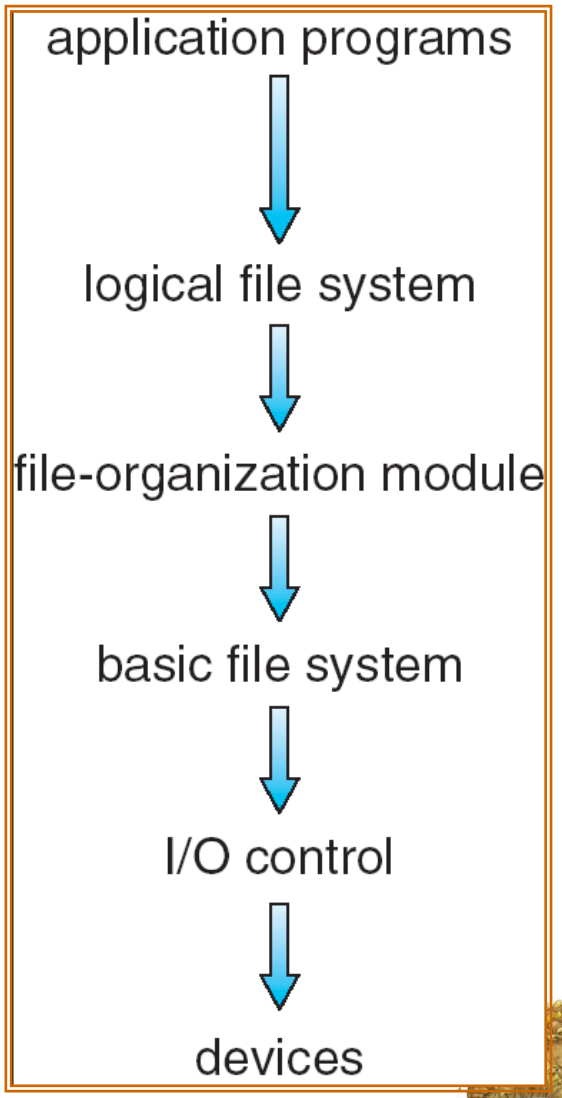
- **逻辑文件系统**
  - 根据给定的文件名，向文件组织模块提供所需要的信息。例如、文件组织模块知道文件的逻辑块。
  - 维护目录结构
  - 处理文件保护与安全
  - (MS Windows 与 Macintosh OS 一样，用户可以看见这个模块的一部分)

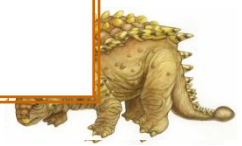




Layered File System

- **当应用层创建一个新文件时**
  - 逻辑文件系统向文件组织模块层提供有关该文件的信息。
  - 文件组织模块层 把文件的逻辑块号翻译成物理块地址。
  - 基本文件系统 …
  - I/O 控制 …





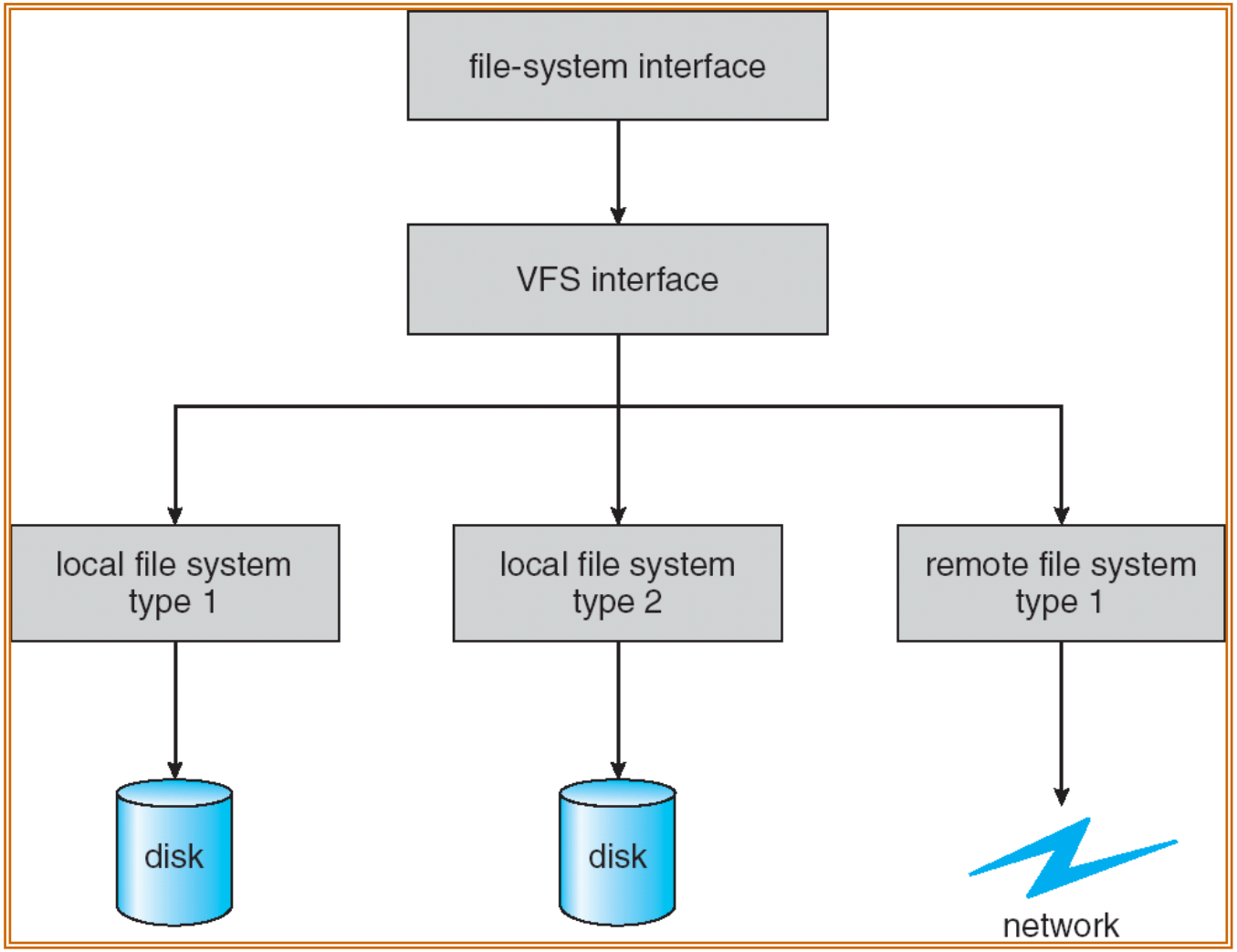
A Typical File Control Block

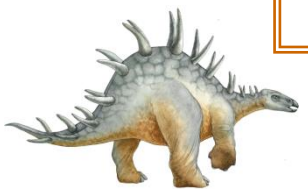
file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

Schematic View of Virtual File System





Directory Implementation

- **Linear list** of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute

- **Hash Table** – linear list with hash data structure.
  - decreases directory search time
  - **collisions** – situations where two file names hash to the same location
  - fixed size

Allocation Methods

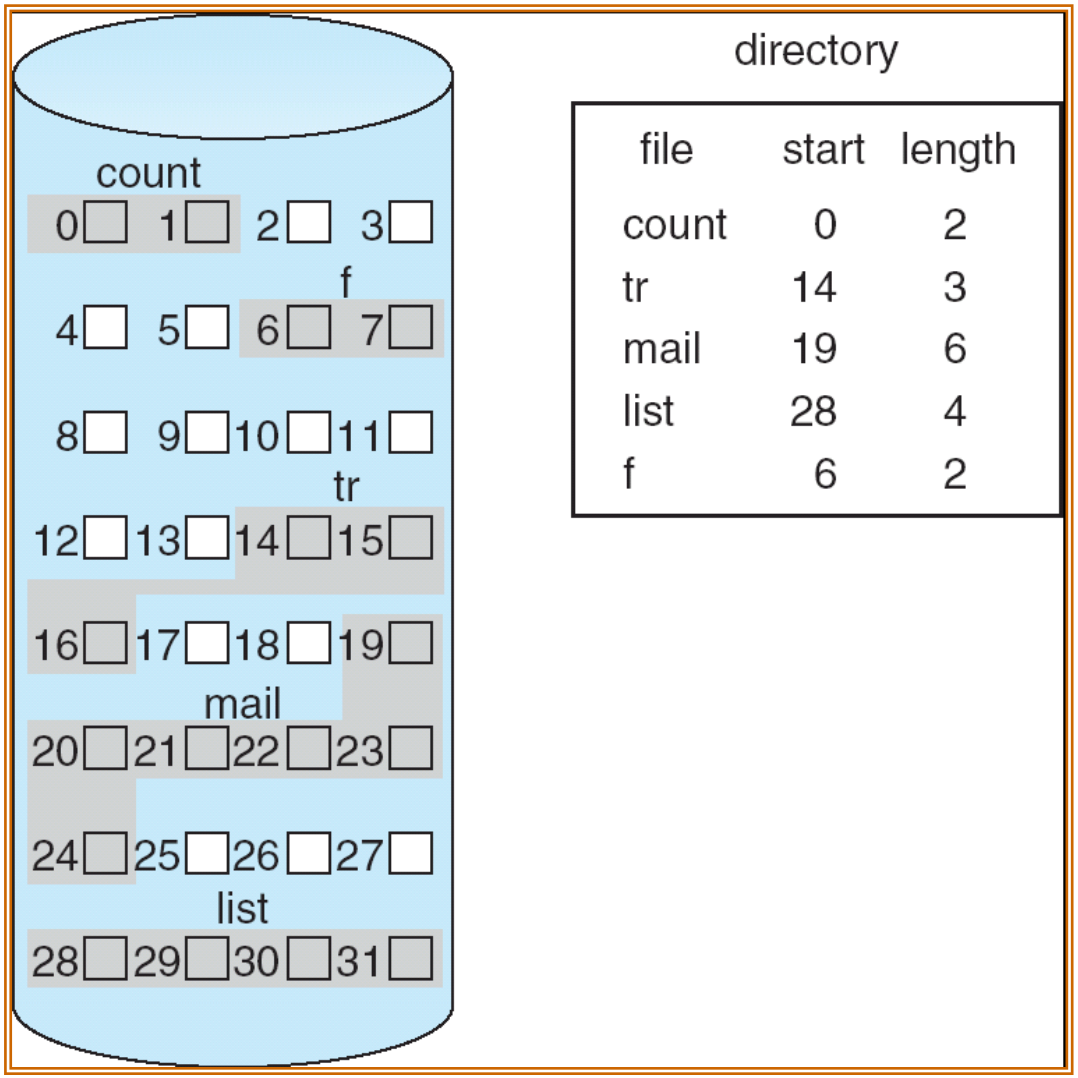
An allocation method refers to how disk blocks are allocated for files:

- **Contiguous allocation**
- **Linked allocation**
- **Indexed allocation**

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow

Contiguous Allocation of Disk Space

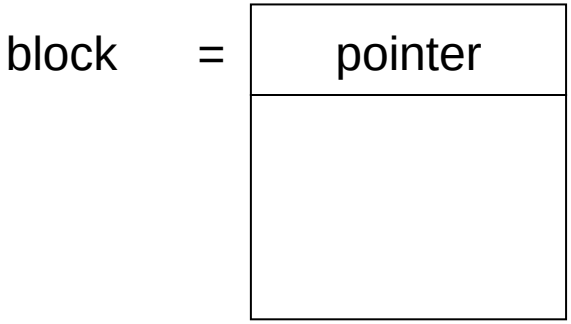


Extent-Based Systems

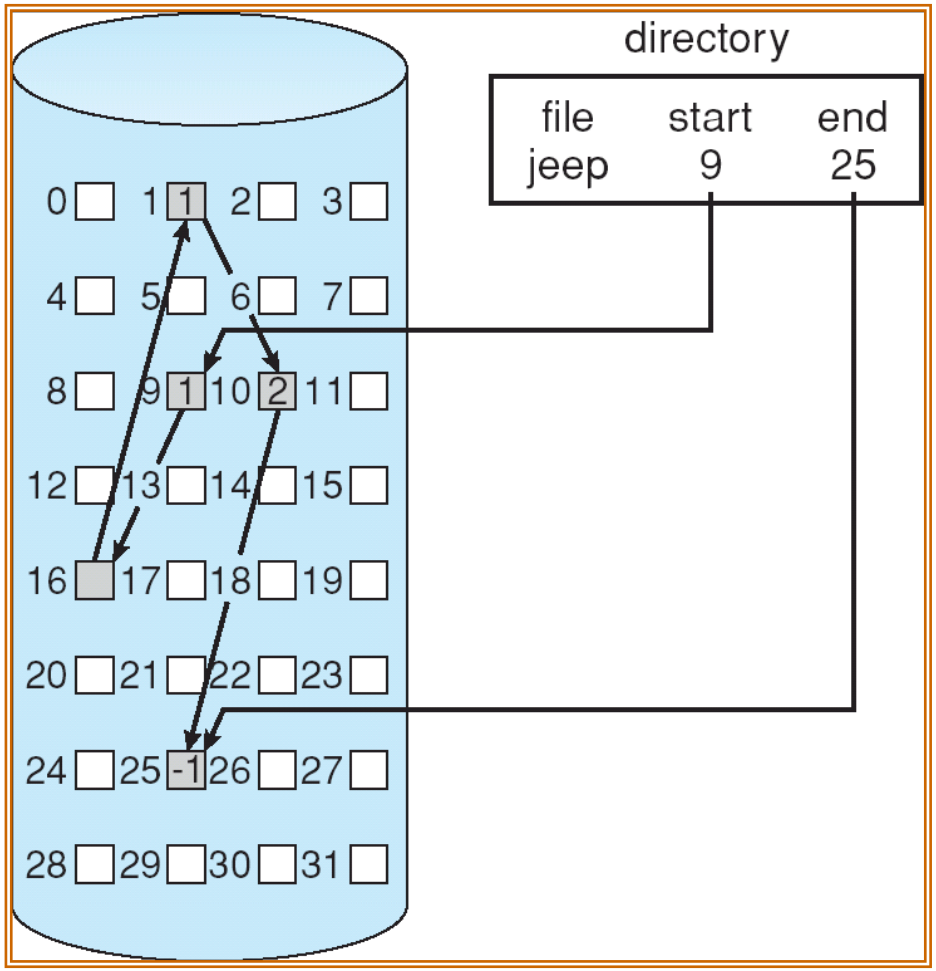
- Many newer file systems (I.e. Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in **extents**
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents.

Linked Allocation

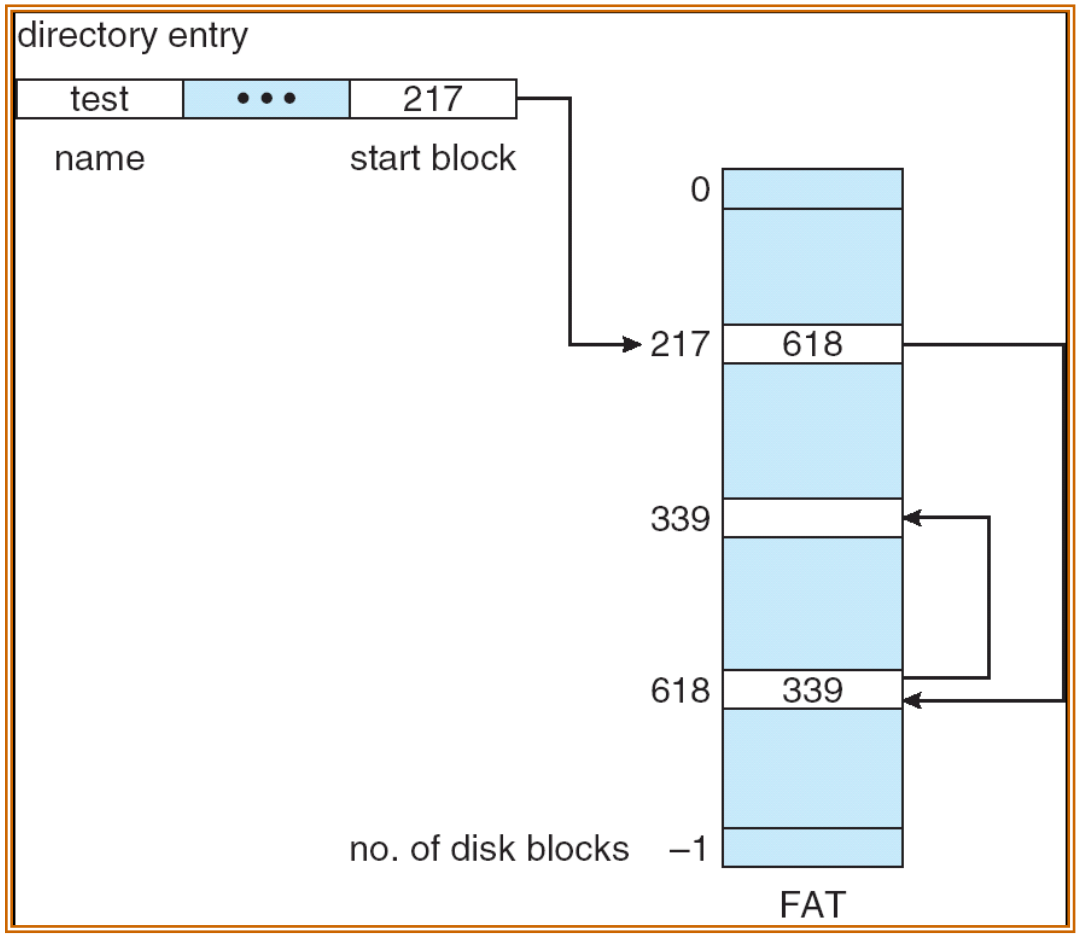
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



Linked Allocation



File-Allocation Table



File-Allocation Table

FAT16由于受到先天的限制，因此每超过一定容量的分区之后，它所使用的簇(Cluster)大小就必须扩增，以适应更大的磁盘空间。所谓簇就是磁盘空间的配置单位，每个要存到磁盘的文件都必须配置足够数量的簇，才能存放到磁盘中。

分区大小	FAT16簇大小
16MB-127M	2KB
128MB-255MB	4KB
256MB-511MB	8KB
512MB-1023MB	16KB
1024MB-2047M	32KB

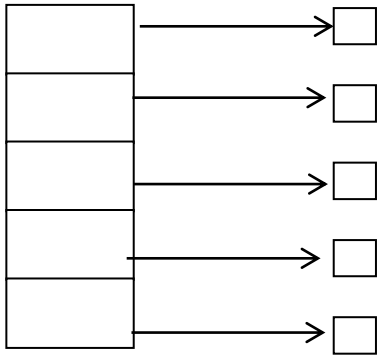
FAT位数	簇数量
12	4,096
16	65,536
32	4,294,967,296

总容量 簇大小（KB）	FAT12（MB）	FAT16（MB）	FAT32（TB）
1	4.1	67	4.398
4	16.7	268	17.59
8	33.4	536	35.16
16	66.8	1070	70.36
32	134.2	2140	140.7

FAT32 发布时间：1996年8月发布(Windows 95 OSR2)

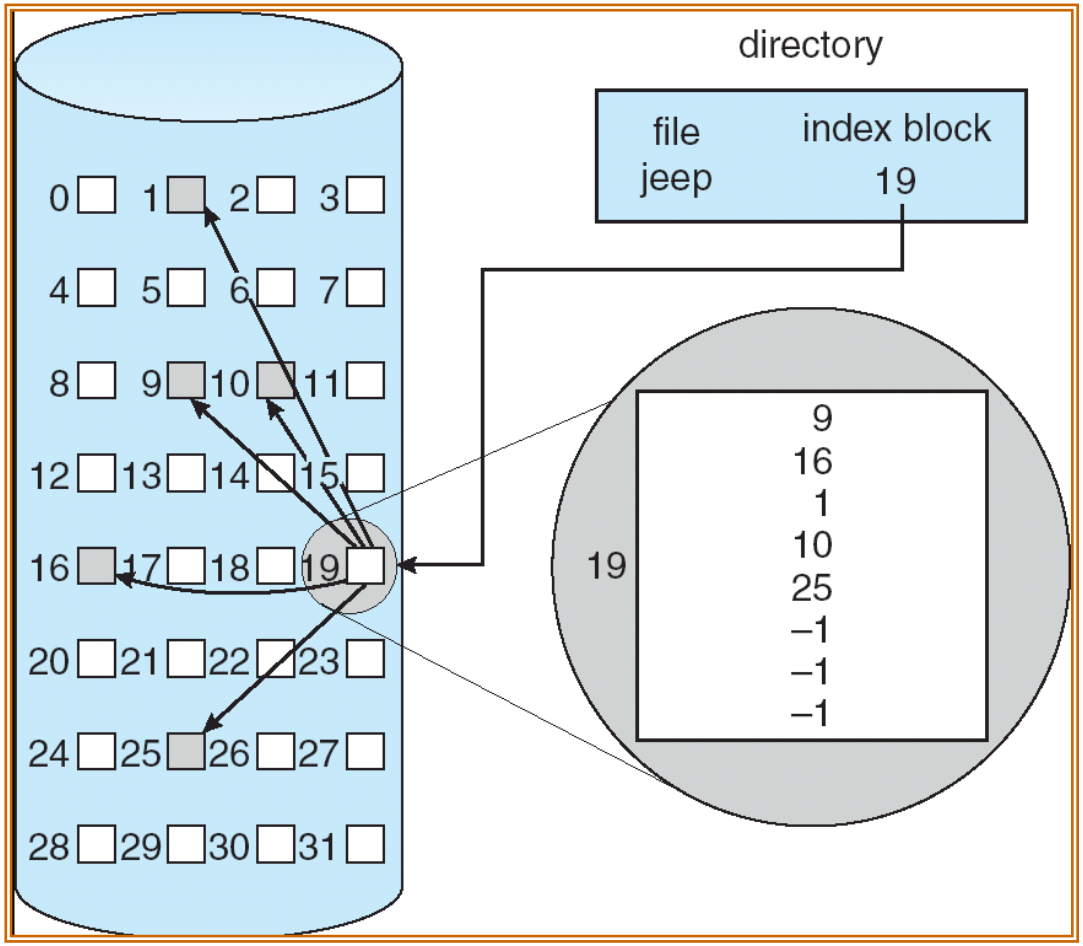
Indexed Allocation

- Brings all pointers together into the *index block*.
- Logical view.

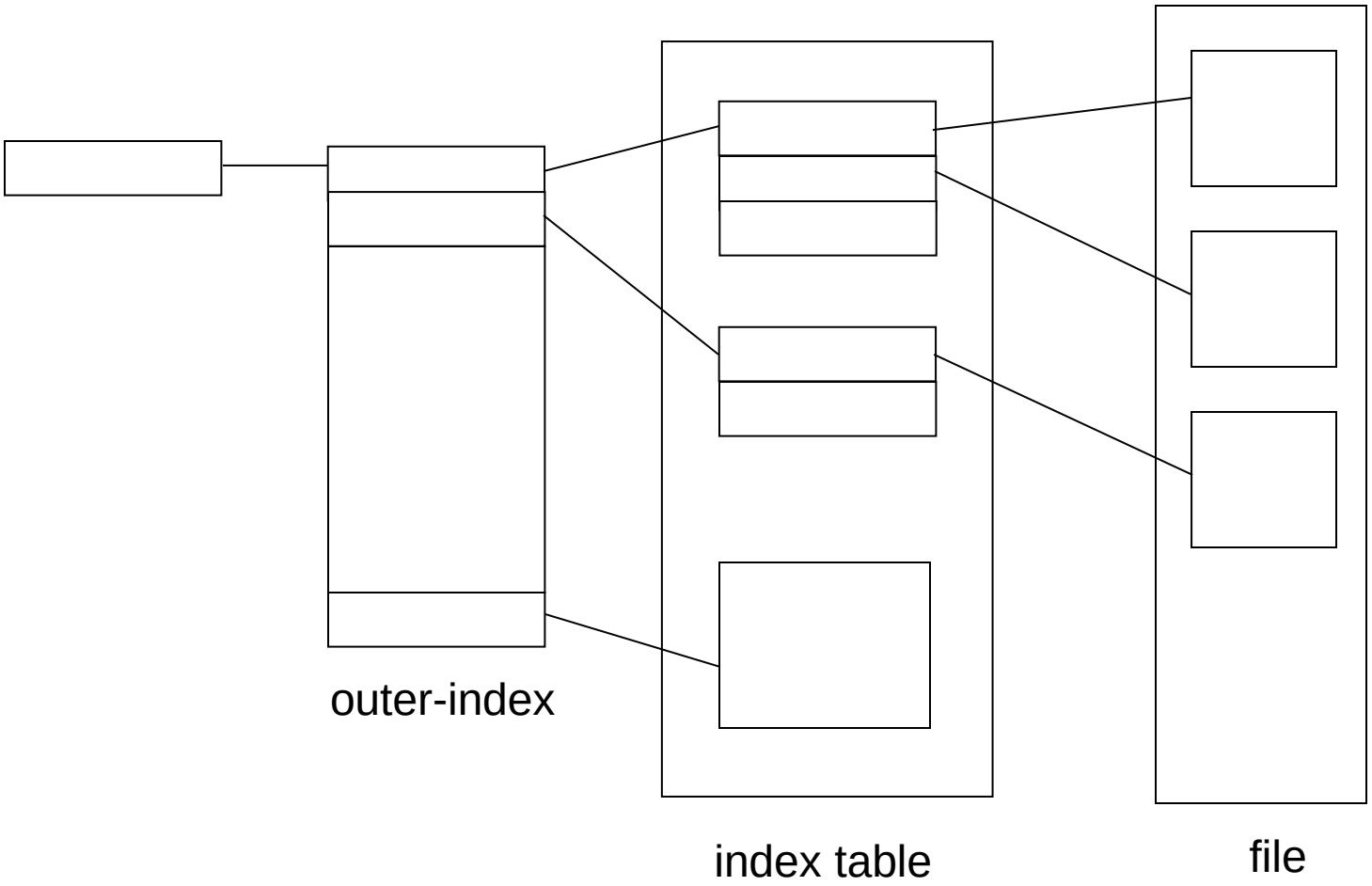


index table

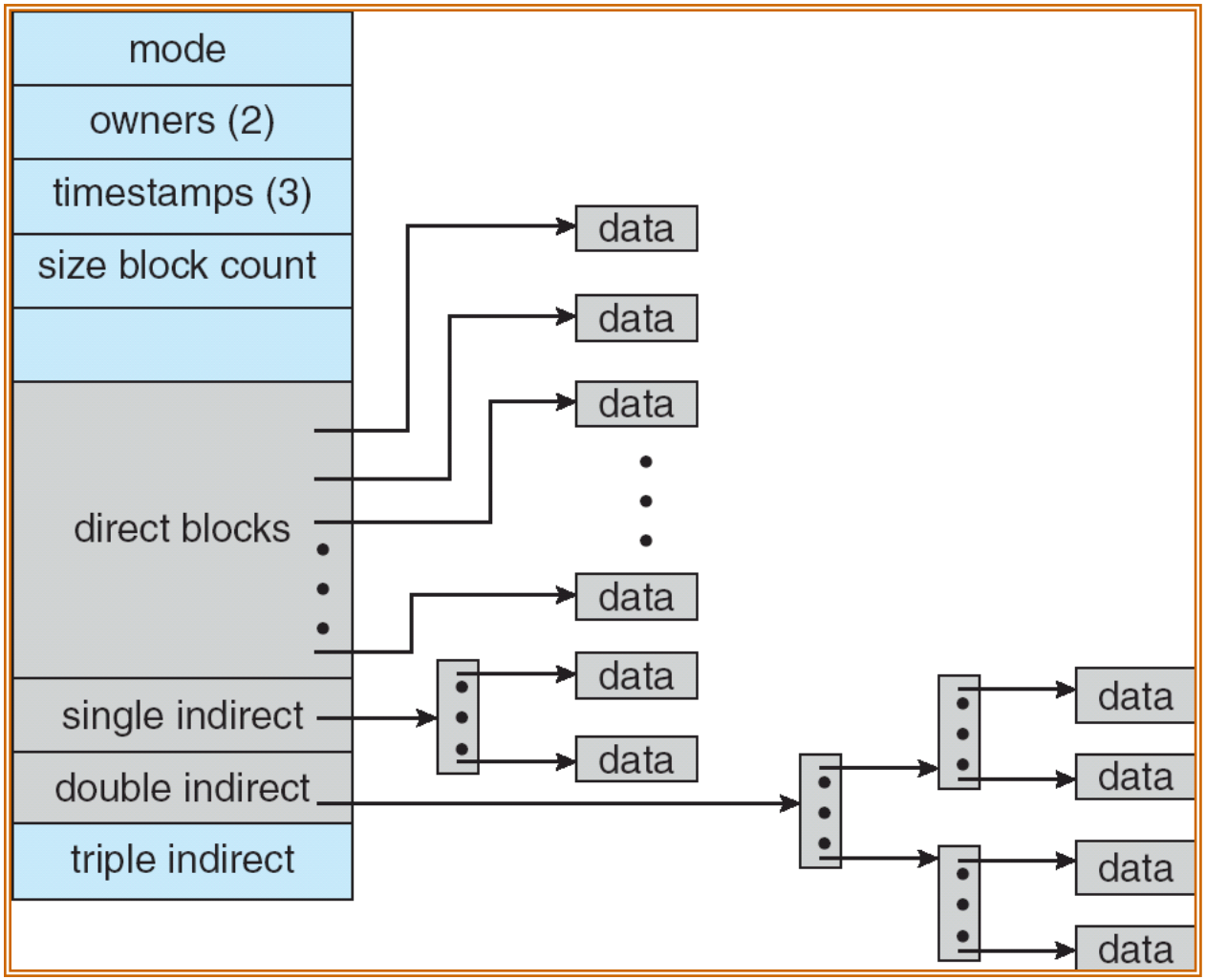
Example of Indexed Allocation



Indexed Allocation – Mapping (Cont.)



Combined Scheme: UNIX (4K bytes per block)

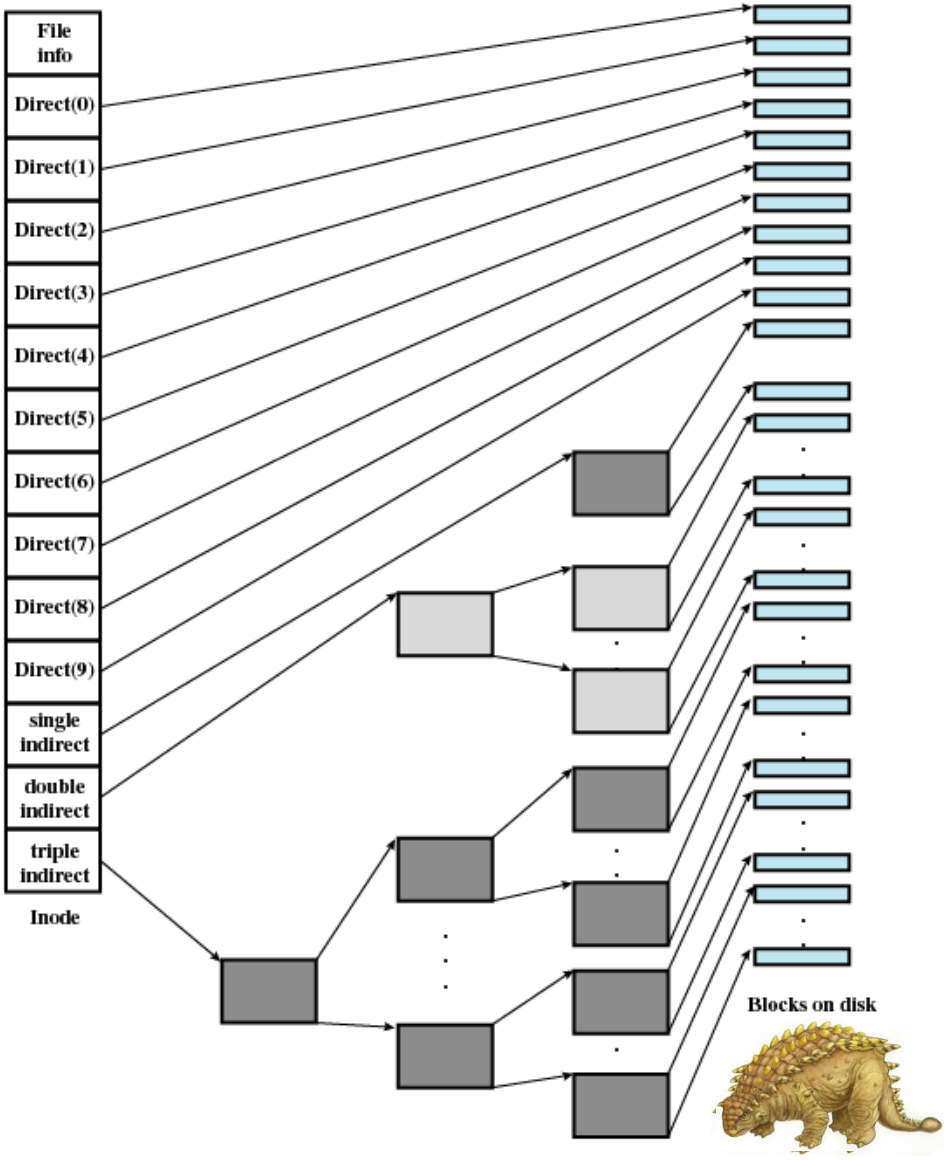


UNIX文件管理

- 索引节点
  - 文件的控制结构
- 文件分配
  - 一级间接块;
  - 二级间接块;
  - 三级间接块;

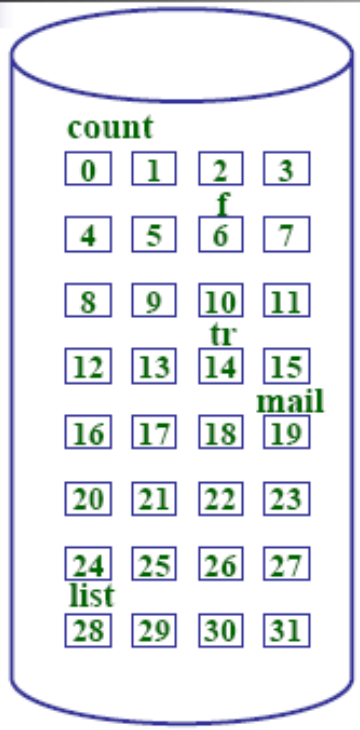
UNIX文件容量

级别	块数	字节数
直接	10	10KB
一级间接	256	256KB
二级间接	65K	65MB
三级间接	16M	16G



UNIX文件管理

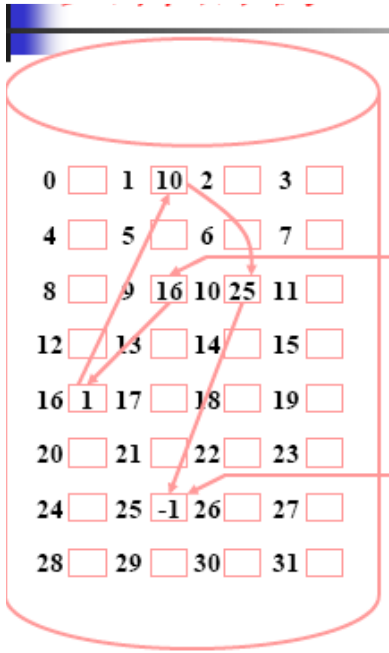
- **优点：**
  - 索引节点大小固定，并且相对较小；
  - 小文件可以快速访问，减少处理时间；
  - 文件大小满足应用程序需求；



文件目录

文件名	始址	块数
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

连续分配

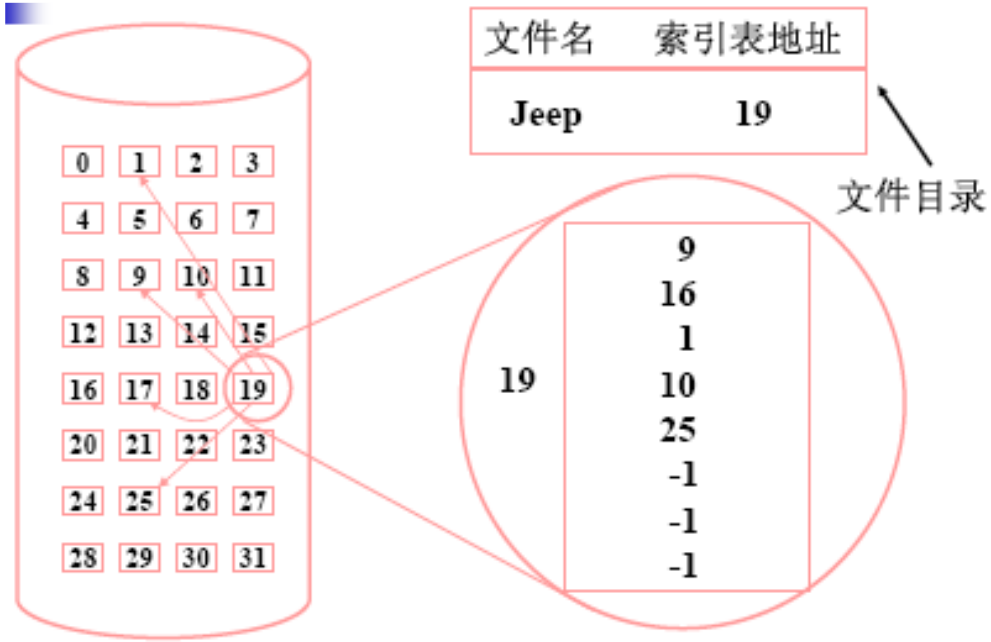


文件目录

文件名	始址	末址
jeep	9	25

链接分配

索引分配



连续分配方法

连续分配

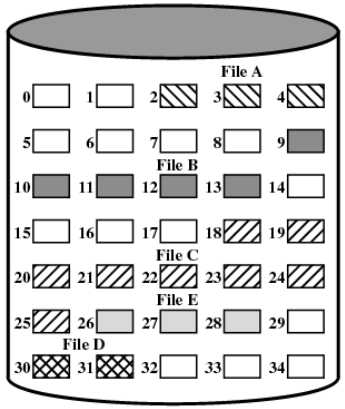
创建文件时，给文件分配一组连续的块

优点：

- 简单
- 支持顺序存取和随机存取
- 顺序存取速度快

缺点：

- 外部碎片；
- 预分配；



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

链式分配方法

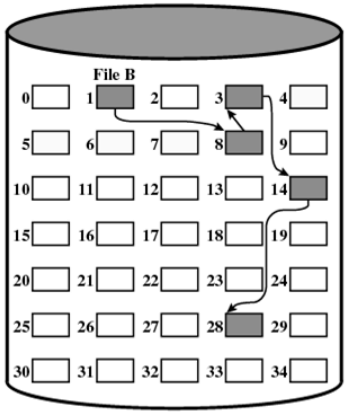
创建文件时，每块都包含指向下一块的指针

优点：

- 提高了磁盘空间利用率，不存在外部碎片问题
- 有利于文件插入、删除和扩充

缺点：

- 存取速度慢，适宜顺序存取，不适于随机存取
- 链接指针占用一定的空间

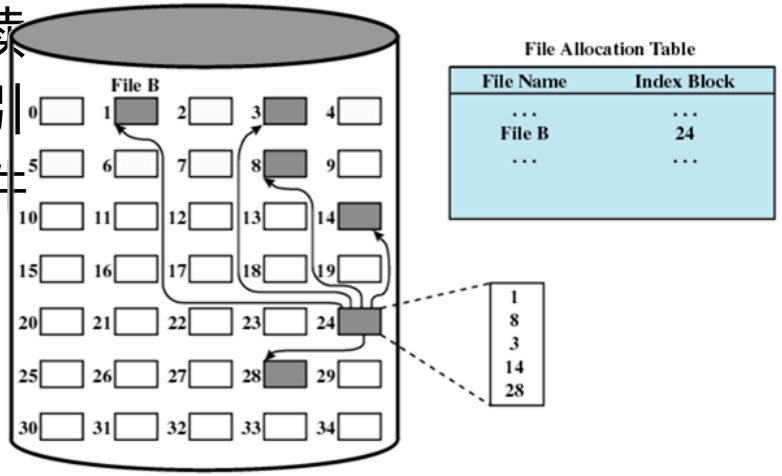


File Allocation Table

File Name	Start Block	Length
...	...	...
File B	1	5
...	...	...

连续分配方法

创建文件时，文件的信息存放在若干不连续物理块中。在文件分配表中有一个一级索引。支持顺序和直接访问，最普遍的一种文件分配形式。

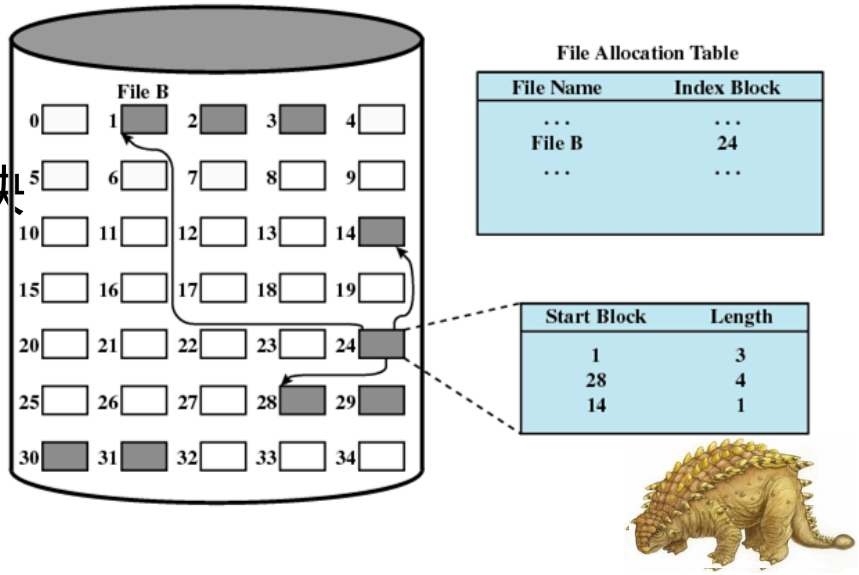


优点：

- 既能顺序存取，又能随机存取
- 满足了文件动态增长、插入删除的要求
- 能充分利用外存空间

缺点：

- 需要访问两次内存：索引表、具体物理块
- 索引表本身带来了系统开销



Windows文件管理

FAT/FAT32

- FAT(File Allocation Table) “文件分配表” 。FAT16： DOS 、Windows 95 都使用 FAT16 文件系统，最大可以管理大到 2GB 分区，但每个分区最多只能有 65525 个簇。随着硬盘或分区容量增大，每个簇所占的空间将越来越大，从而导致硬盘空间的浪费。
- FAT32： 从 Windows 98 开始， 是 FAT16 的增强版本，可以支持大到 2TB （2048G 的分区。 FAT32 使用的簇比 FAT16 小，从而有效地节约了硬盘空间。

Windows文件管理

NTFS

- 新(N)技术(T)文件(F)系统(S) 微软 Windows NT 内核的系列操作系统支持的、一个特别为网络和磁盘配额、文件加密等管理安全特性设计的磁盘格式。NTFS 也是以簇为单位来存储数据文件，但 NTFS 中簇的大小并不依赖于磁盘或分区的大小。簇尺寸的缩小不但降低了磁盘空间的浪费，还减少了产生磁盘碎片的可能。 NTFS 支持文件加密管理功能，可为用户提供更高层次的安全保证。

Windows文件管理

NTFS的重要特性

- 可恢复性;
- 安全性;
- 大磁盘和大文件;
- 多数据流;
- 通用索引功能;

Windows文件管理

NTFS卷和文件结构

- 扇区
  - 磁盘中最小的物理存储单元。存储数据字节数总量为2的幂，通常为512字节；
- 簇
  - 一个或多个连续的扇区。一个簇中扇区的数目也为2的幂；
- 卷
  - 磁盘上的逻辑分区，由一个或多个簇组成。可以为整个磁盘、部分磁盘或跨越多个磁盘；

Windows文件管理

NTFS卷和文件结构

- NTFS不识别扇区，簇是最基本的分配单位。它包含整数个物理扇区；而扇区是磁盘中最小的物理存储单位。
  - 扇区通常存放512个字节。
  - 簇的大小可由格式化命令或格式化程序按磁盘容量和应用需求来确定，可以为512B、1KB、2KB、…、最大可达64KB 。
- NTFS支持的最大文件2^{32}个簇，2^{48}字节。

Windows文件管理

NTFS卷的布局

- 分区引导扇区;
- 主文件表;
- 系统文件;
- 文件区域;

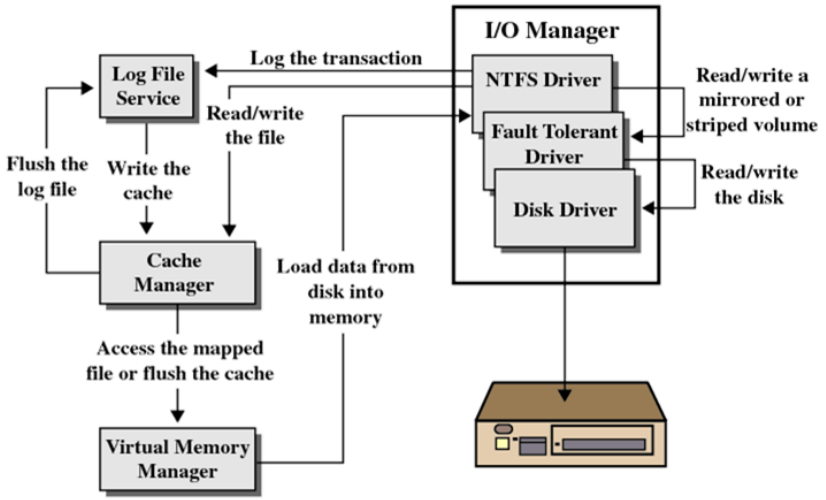
Windows文件管理

NTFS可恢复性：

- I/O管理器；
- 日志文件服务；
- 高速缓存管理器；
- 虚存管理器；

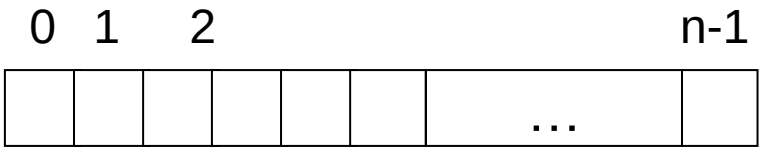
NTFS通过日志记录实现文件的可恢复性。改变文件系统的子操作在磁盘上运行前，先被记录在日志文件中。

当系统崩溃后，NTFS根据记录在日志中的文件操作信息，对那些部分完成的事务进行重做或撤销，保证磁盘上文件的一致性。



Free-Space Management

- Bit vector (*n* blocks)



bit[*i*] = 0 block[*i*] occupied
1 block[*i*] free

Block number calculation

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Free-Space Management (Cont.)

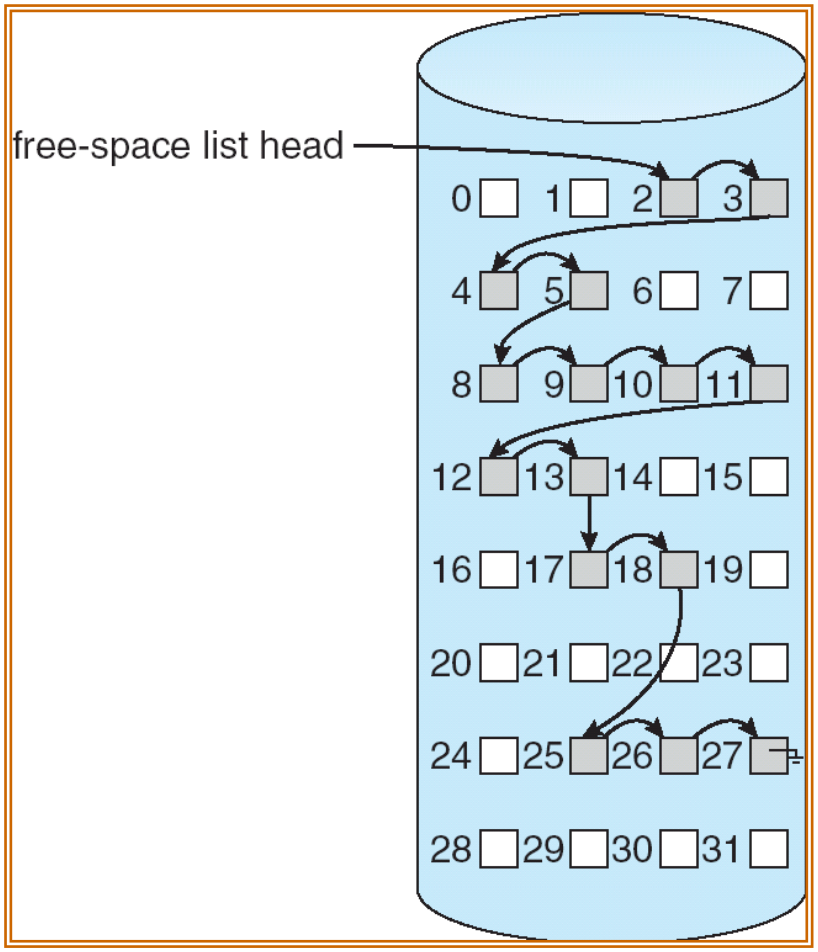
- Bit map requires extra space
  - Example:

    block size = 2^{12} bytes

    disk size = 2^{30} bytes (1 gigabyte)

    $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
- Easy to get contiguous files
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
- Counting

Linked Free Space List on Disk



Grouping分组（成组链接法）

- Store the addresses of n free blocks in the first free block. The first n-1 of these blocks are actually free. The last block contains the addresses of another n free blocks, and so on.

把n个空闲块的地址存放在第一个空闲块中，其中前n-1个块是真正空闲的，最后一个块包含另一组n块的地址，……

- **Advantage:**
  - the addresses of a large number of free blocks can be found quickly, unlike in the standard linked-list approach.

优点：可以迅速找到大量的空闲盘块，不象标准链接列表方法。

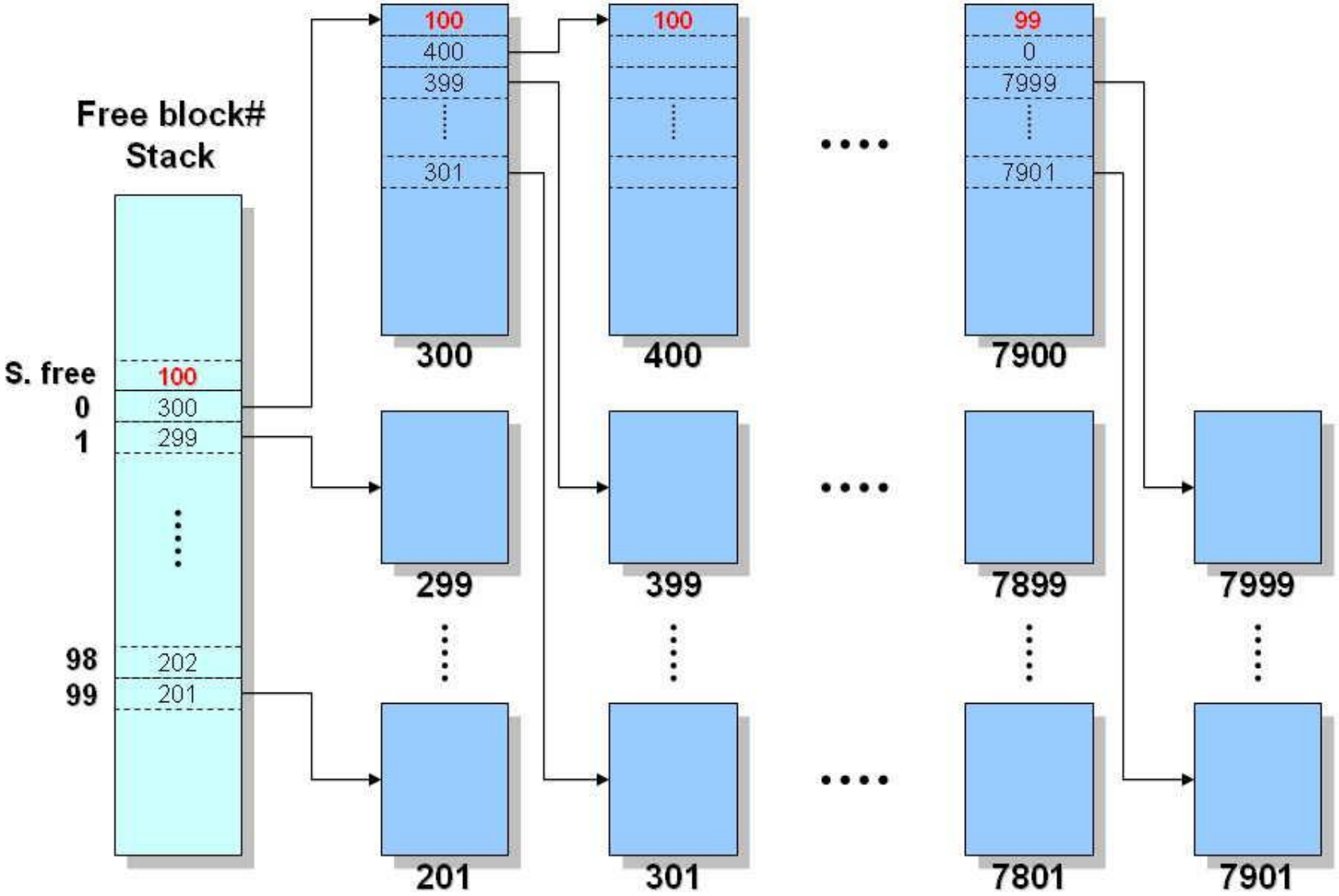


Fig. Grouping

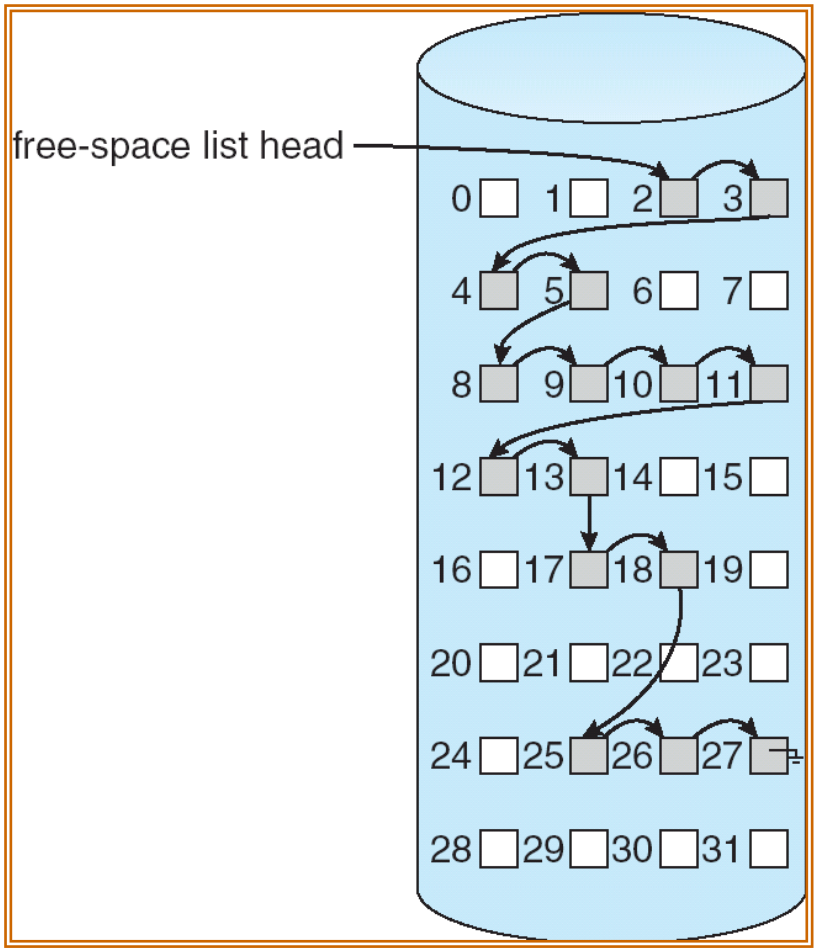
Counting 计数

- Keep the address of the first free block and the number n of free contiguous blocks that follow the first block.

  保持首个空闲块的地址以及其后连续空闲块的数量n.
- Each entry in the free space list then consists of a disk address and a count.

  空闲空间列表的每个条目由一个磁盘地址和一个计数组成。

Linked Free Space List on Disk



End of Chapter 11