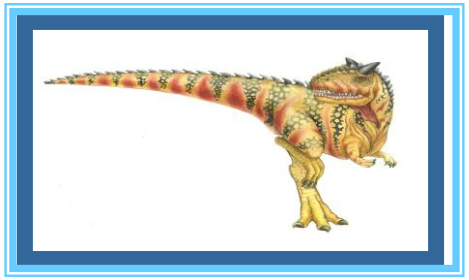
Chapter 11: File System Implementation



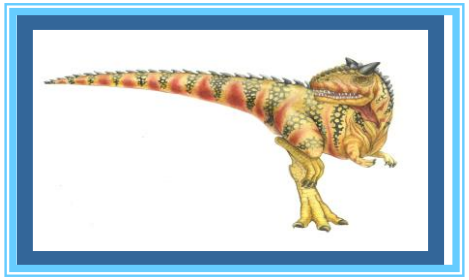
File System Implementation

- 11.1 File-System Structure
- 11.2 File-System Implementation
- 11.3 Directory Implementation
- 11.4 Allocation Methods
- 11.5 Free-Space Management
- 11.6 Efficiency and Performance
- 11.7 Recovery
- 11.8 NFS

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

11.1 File-System(organized) Structure



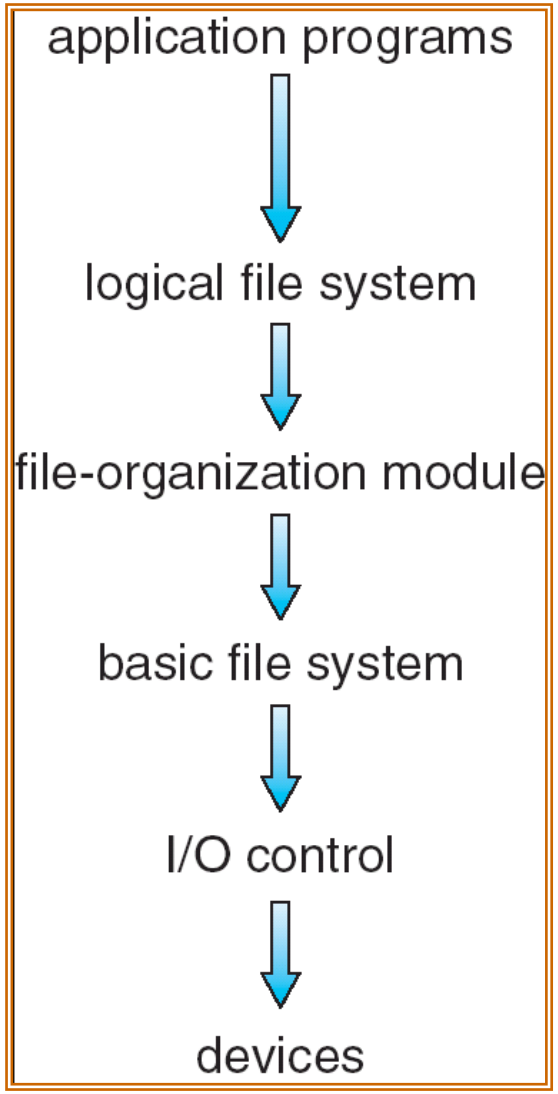
File-System(organized) Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
  - 文件系统是操作系统中以文件方式管理计算机软件资源的软件、被管理的文件和数据结构（如目录和索引表等）的集合。
  - 文件系统是对一个存储设备上的数据和元数据进行组织的机制。

文件系统结构

- 磁盘提供大量的外存空间来维持文件系统，磁盘的两个特点，使其成为存储多个文件的方便媒介
  - 可以原地重写；可以从磁盘上读一块，修改该块，并将它写回到原来的位置
  - 可以直接访问磁盘上的任意一块信息。（随机或顺序方式）
- 为了提供对磁盘的高效且便捷的访问，操作系统通过文件系统来轻松地存储、定位、提取数据。
- 文件系统的两个设计问题
  - 如何定义文件系统对用户的接口
  - 创建数据结构和算法来将逻辑文件系统映射到物理外存设备上
- 文件系统按层来组织

分层设计的文件系统



- **Application Programs:** The code that's making a file request
- **逻辑文件系统**
  - 管理**元数据**：文件系统的所有结构数据，而不包括实际数据（或文件内容）
  - 根据给定符号文件名来管理目录结构
  - 逻辑文件系统通过**文件控制块(FCB)**来维护文件结构

内存

- **文件组织模块**
  - 知道文件及其逻辑块和物理块。
  - 空闲空间管理器

外存

- **基本文件系统**
  - 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
- **I/O控制**
  - 由**设备驱动程序**和**中断处理程序**组成，实现内存与磁盘之间的信息转移

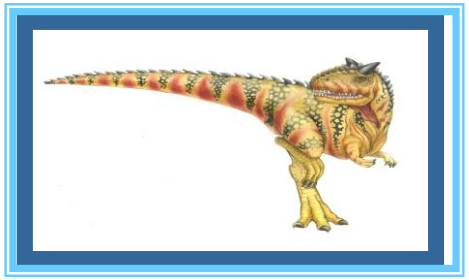
File System Types

- ✓ FAT（MS-DOS文件系统）
- ✓ FAT32（VFAT），exFAT （64位）
- ✓ **NTFS（NT文件系统）**
- ✓ **ReFS(Resilient File System)复原文件系统（Windows 8后、server）**
- ✓ S51K/S52K（AT&T UNIX sysv）
- ✓ ext (minix文件系统)
- ✓ **ext2、ext3、ext4（linux文件系统、Android）**
- ✓ proc、sysfs（linux虚拟文件系统）
- ✓ yaffs (Yet Another Flash File System)
- ✓ ReiserFS (Linux一种日志文件系统)
- ✓ HPFS（OS/2高性能文件系统）
- ✓ UFS(BSD UNIX的一种文件系统)

File System Types

- ✓ **HFS+ (Mac OS，iOS文件系统)**
- ✓ **iso9660 (通用的光盘文件系统)**
- ✓ NFS (网络文件系统)
- ✓ VFS (Linux虚拟文件系统)
  - ✓ VFS是物理文件系统与服务之间的一个接口，它屏蔽各类文件系统的差异，给用户和程序提供一个统一的接口
- ✓ ZFS （Open Solaris文件系统）
- ✓ **LTFS(Liner Tape File System，线性磁带文件系统) 为磁带提供了一种通用、开放的文件系统。**
- ✓ **APFS（Apple File System）苹果新一代的文件系统，MAC OS 10.13以后、iOS 10.3以后**
- ✓ ….

11.2 File-System Implementation



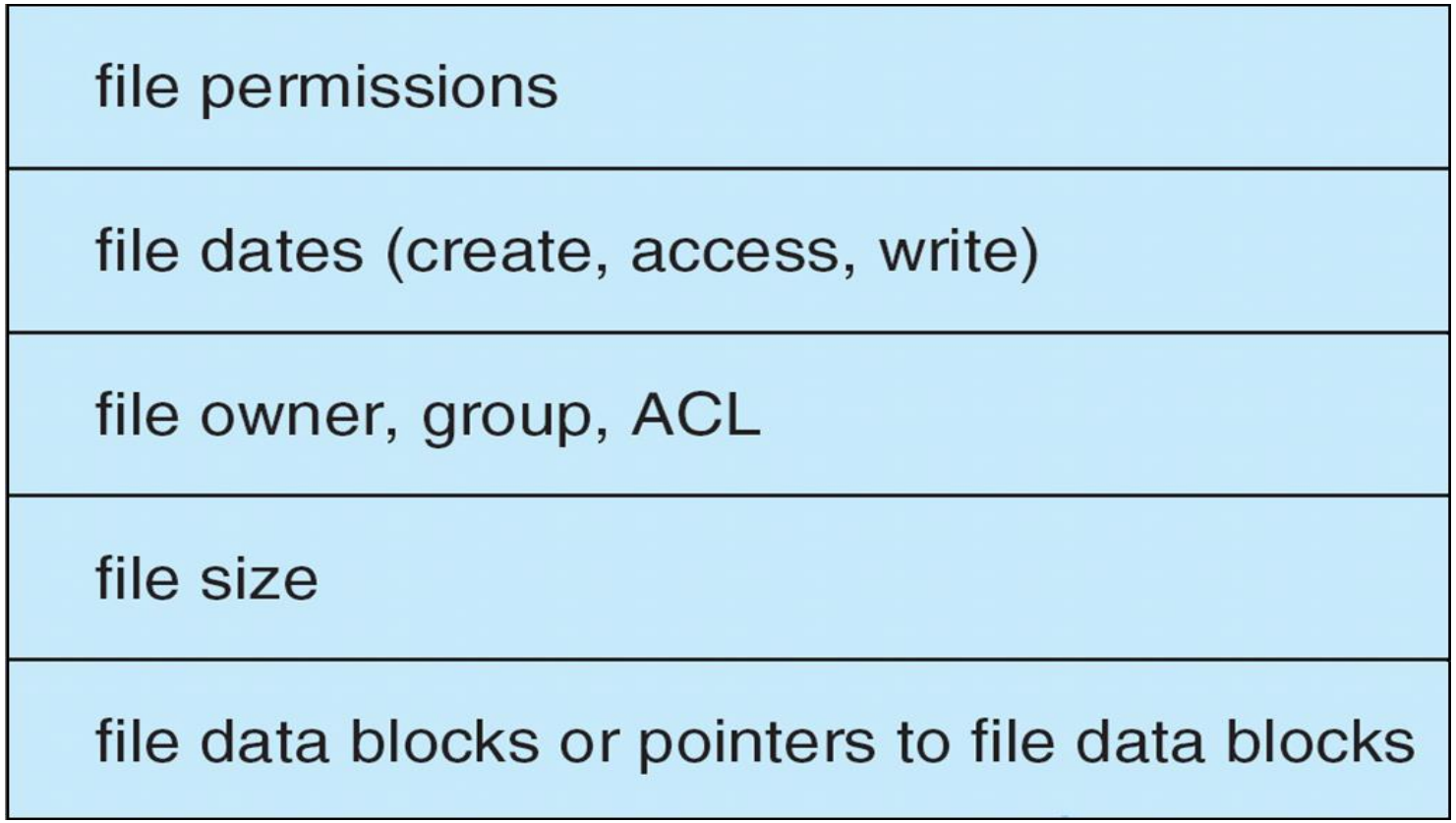
File-System Implementation

1. On-Disk file system structures:

- 在磁盘上，文件系统可能包括如下信息：
  - 如何启动所存储的操作系统
  - 总的块数
  - 空闲块的数目和位置
  - 目录结构以及各个具体文件等
- 磁盘结构包括
  - Boot control block contains info needed by system to boot OS from that volume
  - Volume（卷） control block contains volume details
  - Directory structure organizes the files
  - Per-file File Control Block (FCB，文件控制块) contains many details about the file
- 实例：Linux ext2

A Typical File Control Block

- File control block （文件控制块）– storage structure consisting of information(Attributes) about a file



In-Memory File System Structures

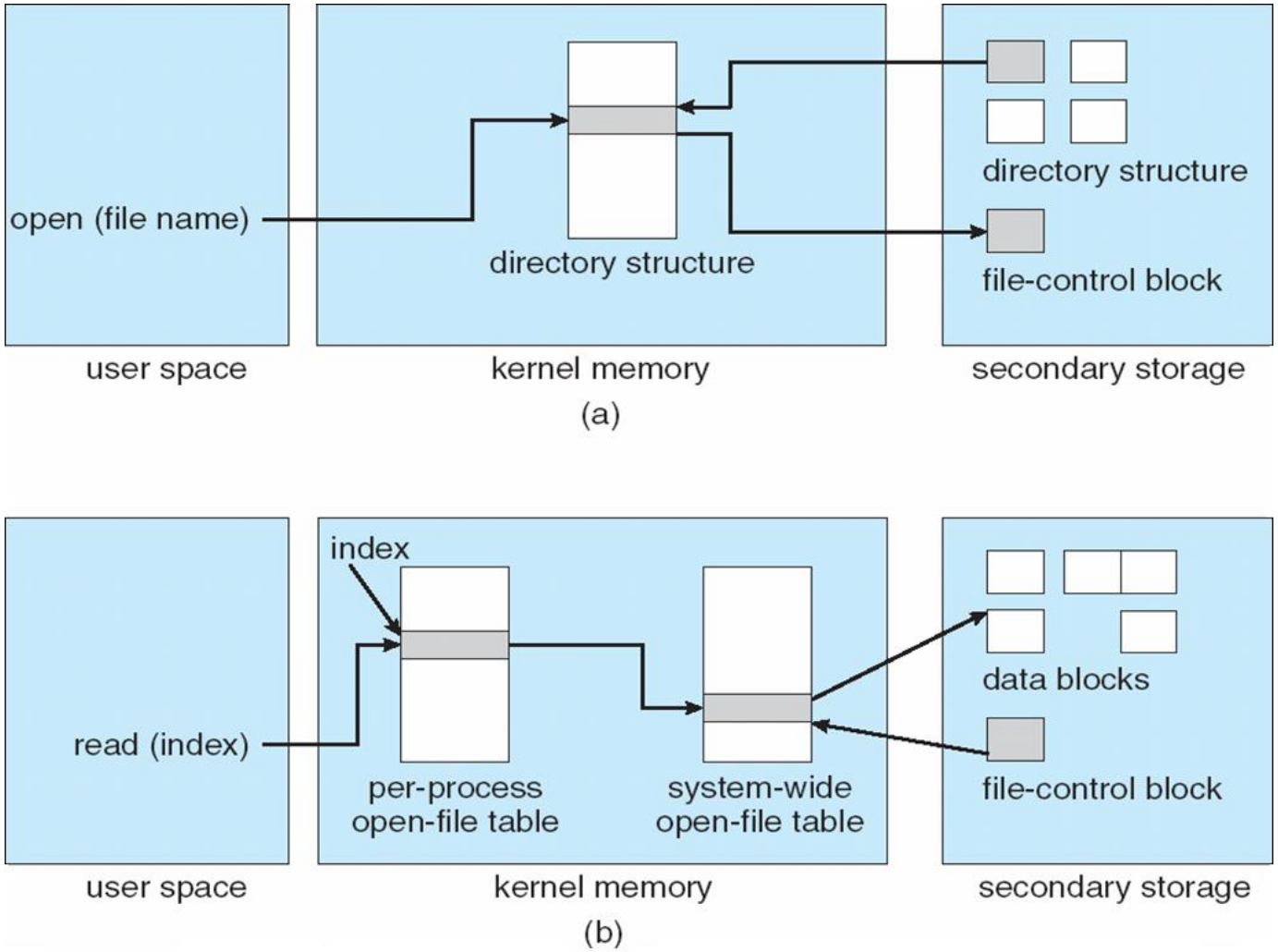
2. In-Memory File System Structures:

- An in-memory partition table(分区表)
- An in-memory directory structure（目录结构）
- The system-wide open-file table （系统打开文件表）
- The per-process open-file table （进程打开文件表）

  - Figure 12-3(a) refers to opening a file.
  - Figure 12-3(b) refers to reading a file.

- 实例：Linux vfs

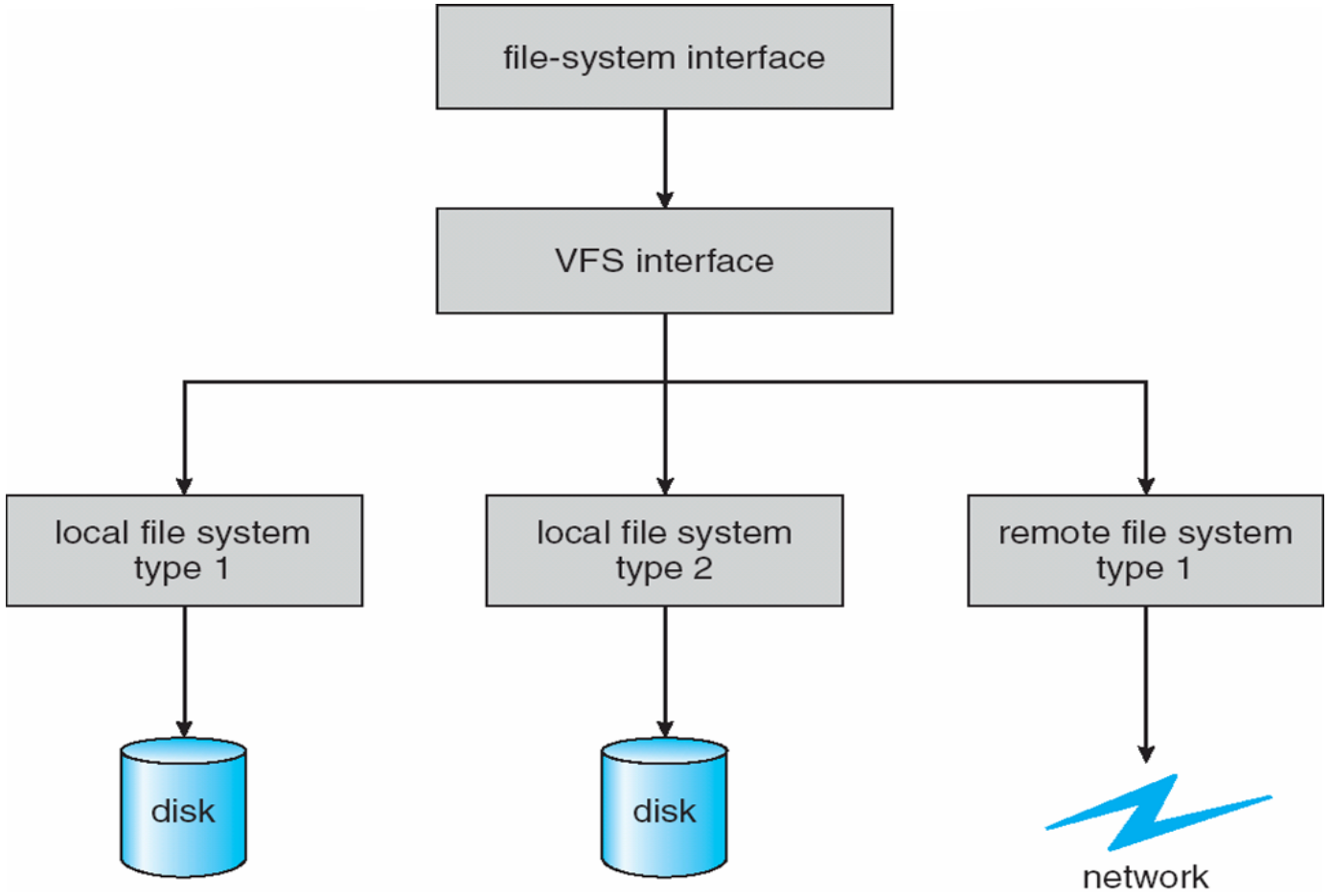
In-Memory File System Structures



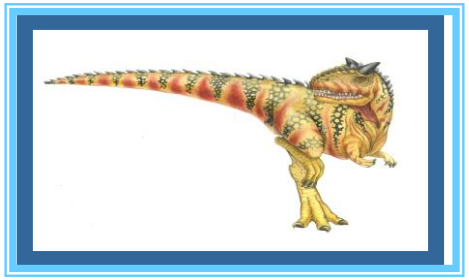
Virtual File Systems

- 虚拟文件系统Virtual File Systems (VFS) 提供了一种面向对象的方法来实现文件系统
- VFS允许在不同类型的文件系统上采用同样的系统调用接口（API）
- API是针对VFS的接口，而非对任何特定类型的文件系统

Schematic View of Virtual File System

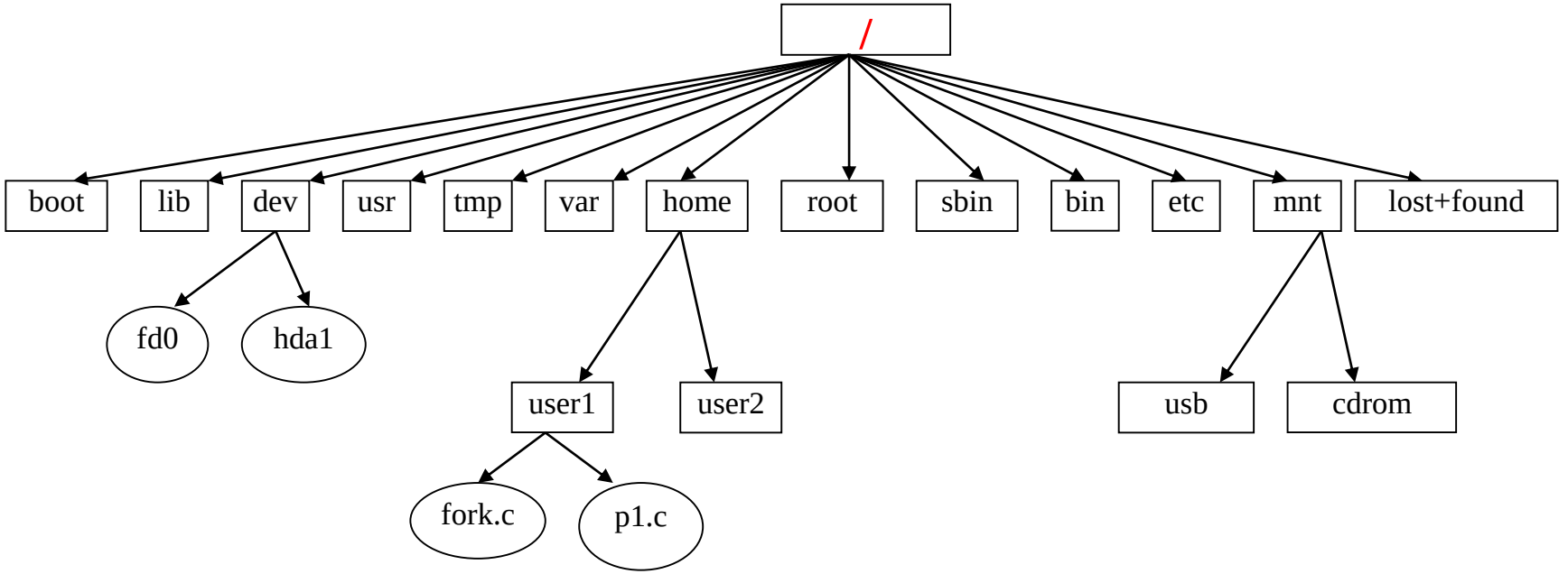


11.3 Directory Implementation



Directory Implementation

- Linear list（线性检索法）：最为简单的目录实现方法是使用存储文件名和数据块指针的线性列表（数组、链表等）--容易实现
  - 但运行费时
  - 采用线性搜索来查找特定条目（缺点）
  - 许多操作系统采用软件缓存来存储最近访问过的目录信息，如：Linux的**目录项对象**

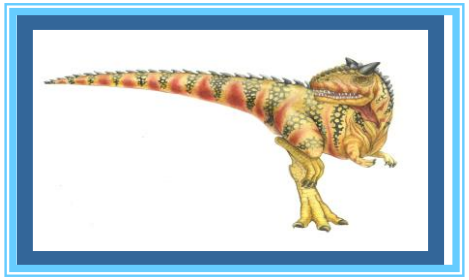


Directory Implementation

- **Hash Table**（哈希表） – linear list with hash data structure.
  - 减少了目录搜索时间
  - **collisions碰撞、冲突**：两个文件名哈希到相同的位置
  - 哈希表的最大困难是其通常固定的大小和哈希函数对大小的依赖性

- 索引

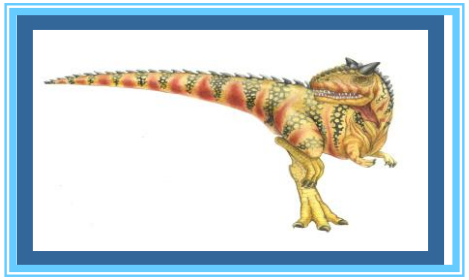
11.4 Allocation Methods，文件物理结构



Allocation Methods，文件物理结构

- An allocation method refers to how disk blocks are allocated for files:
  - Contiguous allocation（连续分配）
  - Linked allocation （链接分配）
  - Indexed allocation（索引分配）
  - Unix、Linux直接间接混合分配方法

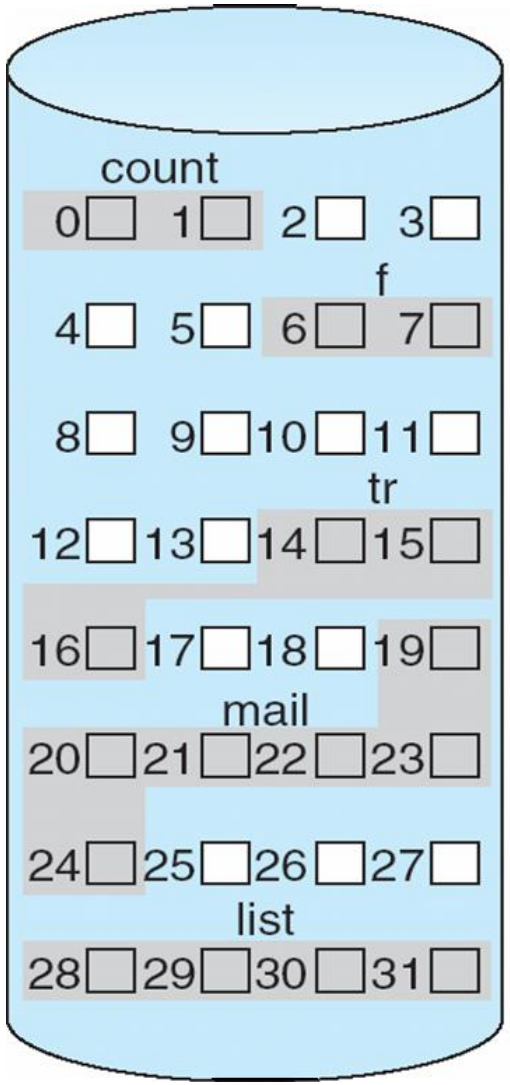
(一) 连续分配 Contiguous Allocation



Contiguous Allocation

- 每个文件占据磁盘上的一组连续的块
- 特点：
  - 简单 — 只需要记录文件的起始位置（块号）及长度。
  - 访问文件很容易，所需的寻道时间也最少
- 存在的问题
  - 为新文件找空间比较困难（类似于内存分配中的连续内存分配方式）
  - 文件很难增长

Contiguous Allocation of Disk Space

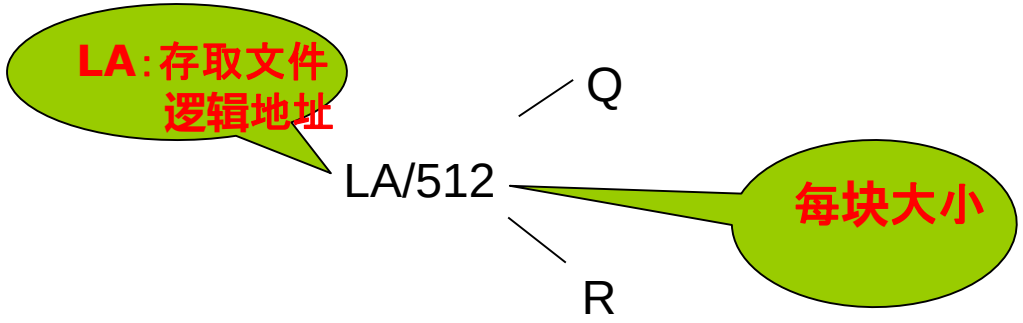


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous Allocation

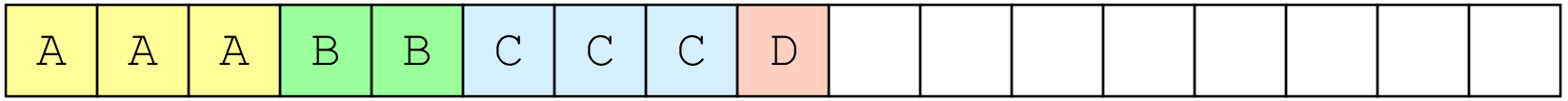
- Mapping from logical to physical



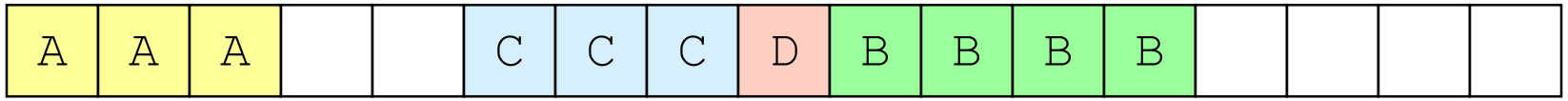
Block to be accessed = Q + starting address
Displacement into block = R

Contiguous Allocation of Disk Space

- Four files allocated contiguously to disk:

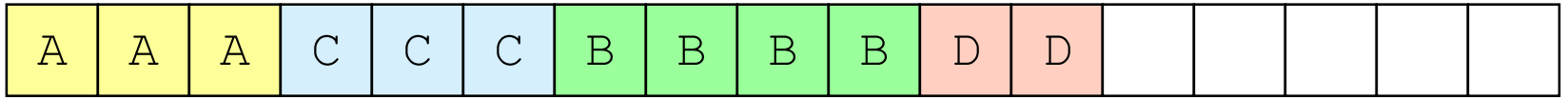


- File B outgrows its space and is reallocated:



- File D outgrows its space and is reallocated:

- Defragmentation combines free disk space:

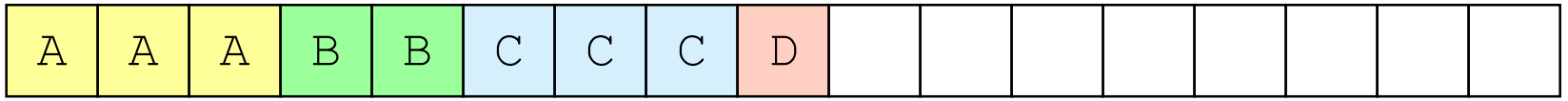


Extent-Based Systems （基于长度系统）

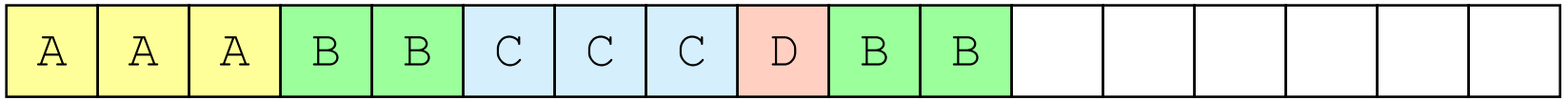
- 许多新的文件系统采用一种修正的连续分配方法
- 该方法开始分配一块连续空间，当空间不够时，另一块被称为扩展的连续空间会添加到原来的分配中。
- **文件块的位置(文件属性之一)**为开始地址、块数、加上一个指向下一扩展的指针。

Extent-Based Systems

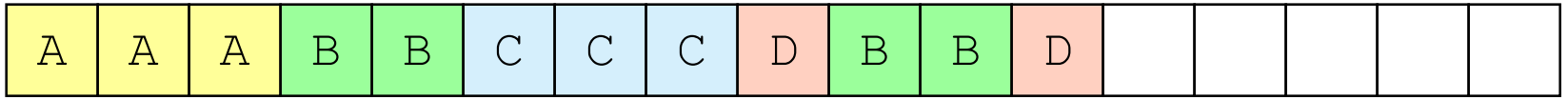
- Four files allocated contiguously to disk:



- File B outgrows its space and a cluster is allocated:

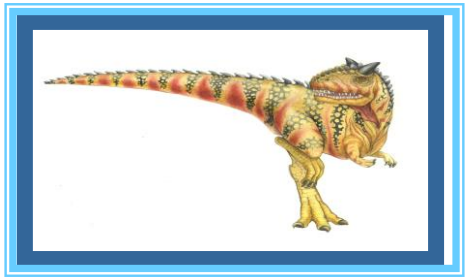


- File D outgrows its space and a cluster is allocated:



- Need for defragmentation is slightly reduced

(二) 链接分配 (linked allocation)

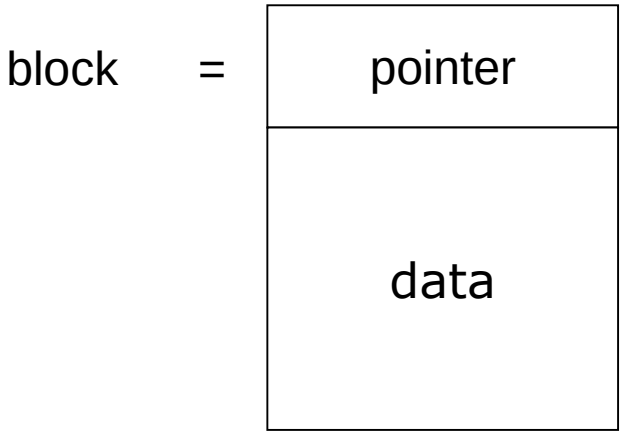


linked allocation

- 每个文件是磁盘块的链表；磁盘块分布在磁盘的任何地方。
- 优点：
  - 简单 — 只需起始位置
  - 文件创建与增长容易
- 缺点：
  - 不能随机访问，必须要遍历
  - 块与块之间的链接指针需要占用空间
    - 簇：将多个连续块组成簇，磁盘以簇为单位进行分配
  - 存在可靠性问题
- 分配簇(Cluster)比分配块(block)更节省指针占用的空间，减少访问时间。如：Windows

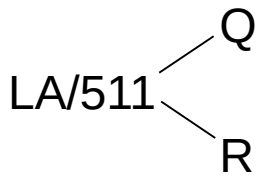
Linked Allocation

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



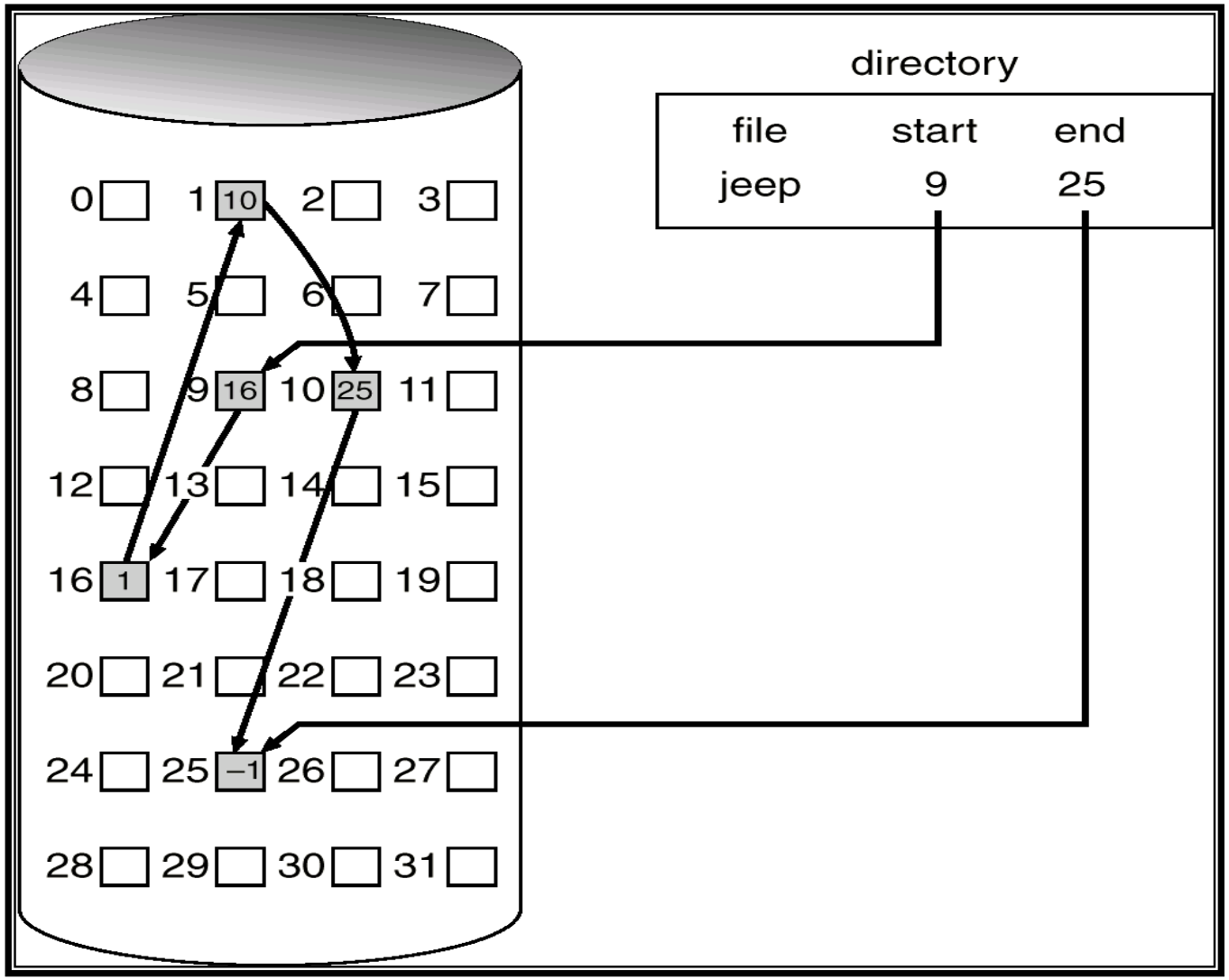
Linked Allocation (Cont.)

- Mapping



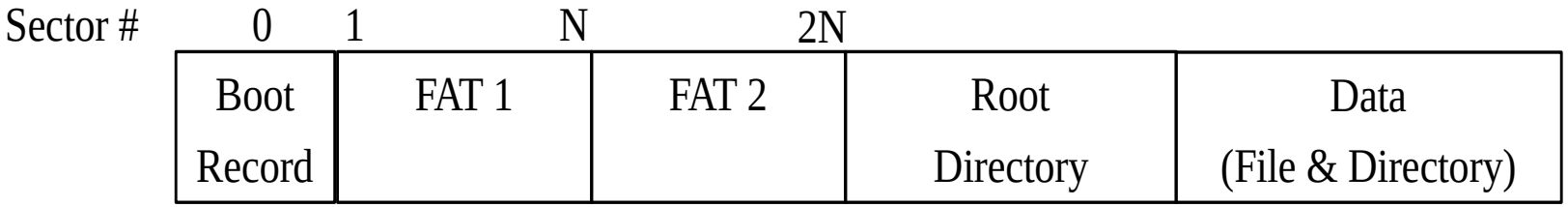
Block to be accessed is the Qth block in the linked chain of blocks representing the file.
Displacement into block = R + 1

Linked Allocation



FAT文件系统

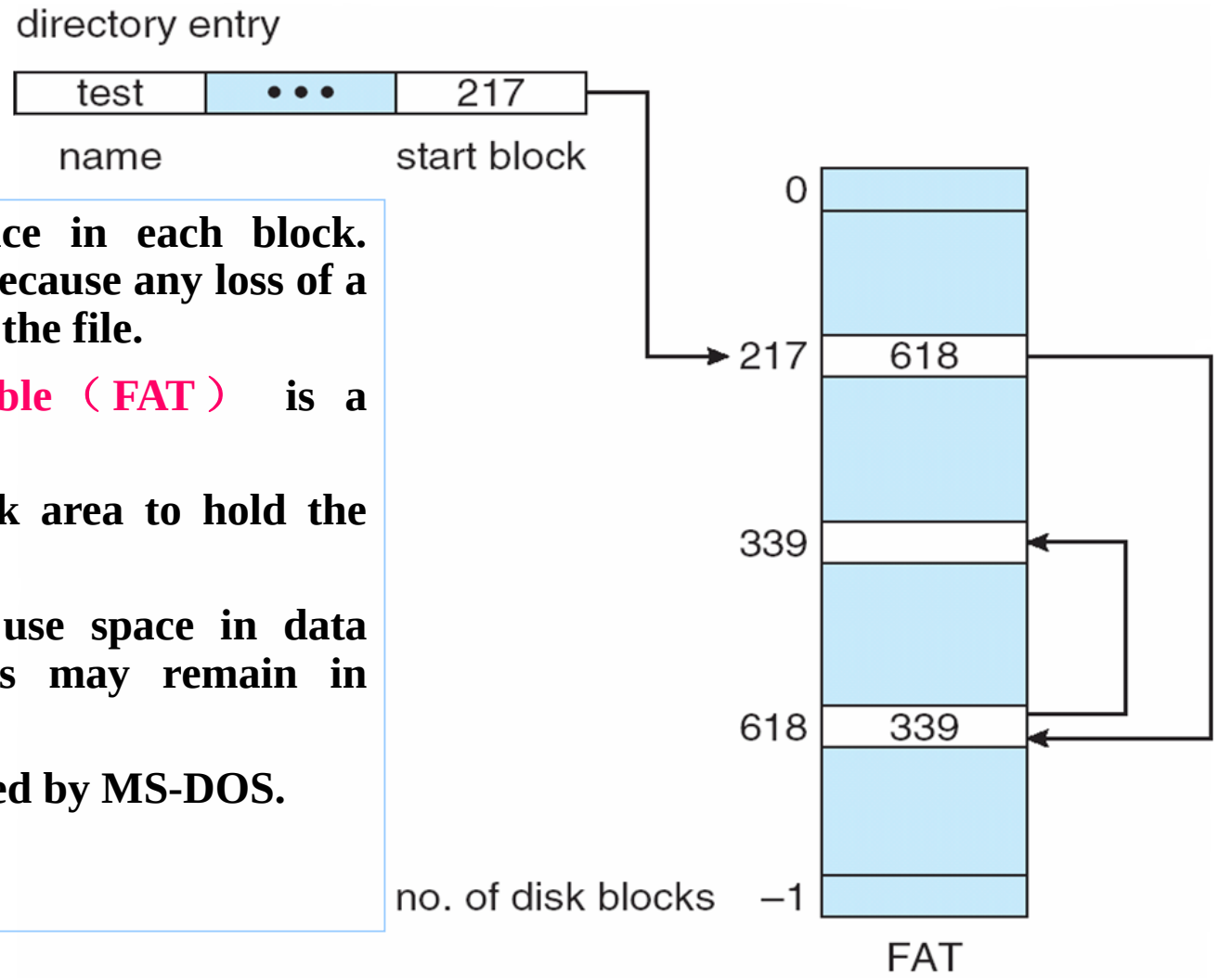
- File-Allocation Table (FAT) – disk-space allocation used by MS-DOS and OS/2.
- FAT12、FAT16、FAT32、exFAT(64位)



Volume Structure in MS DOS

- FAT32引导区记录被扩展为包括重要数据结构的备份，根目录为一个普通的簇链，其目录项可以放在文件区任何地方。

File-Allocation Table



- Pointers use up space in each block. Reliability is not high because any loss of a pointer loses the rest of the file.
- A File Allocation Table（FAT） is a variation of this.
- It uses a separate disk area to hold the links.
- This method doesn't use space in data blocks. Many pointers may remain in memory.
- A FAT file system is used by MS-DOS.

FAT32磁盘的结构

- 主引导记录MBR是主引导区的第一个扇区，它由二部分组成：
  - 第一部分主引导代码，占据扇区的前446个字节，磁盘标识符（FD 4E F2 14）位于这段代码的末尾。
  - 第二部分是分区表，分区表中每个条目有16字节长，分区表最多有4个条目，第一个分区条目从扇区的偏移量位置是0x01BE。
- 扩展引导记录与主引导记录类同，如该扩展分区未装操作系统则第一部分主引导代码为0，标签字也标记一个扩展分区引导区和分区引导区的结束。
- 计算机系统启动时，首先执行的是BIOS引导程序，完成自检，并加载主引导代码和分区表，然后执行主引导程序，由它加载激活分区（安装操作系统的分区）引导记录，再执行分区引导记录程序，加载操作系统，最后执行操作系统，配置系统。

FAT32目录项结构

- FAT的每个目录项为32个字节
- FAT32长文件名的目录项由几个32B表项组成。
- 用一个表项存放短文件名和其他属性（包括簇号、文件大小，最后修改时间和最后修改日期、创建时间、创建日期和最后存取日期），短文件名的属性是0x20。
- 用连续若干个表项存放长文件名，每个表项存放13个字符（使用Unicode编码，每个字符占用2个字节。）
- 长文件名的表项首字节的二进制数低5位值，分别为00001 、00010 、 00011 、……，表示它们的次序，左起第2位为1（也就是在低5位基础上加40H）表示该表项是最后一项。最后项存放13个字符位置多余时，先用2个字节0表示结束，再用FFH填充。长文件名的属性是0x0F。长文件名项的第13、27、28字节为0x00，第14字节为短文件名校验和。

FAT32目录结构

- 长文件名The quick brown.fox（短文件名为THEQUI~1.FOX）
  目录项格式如下：

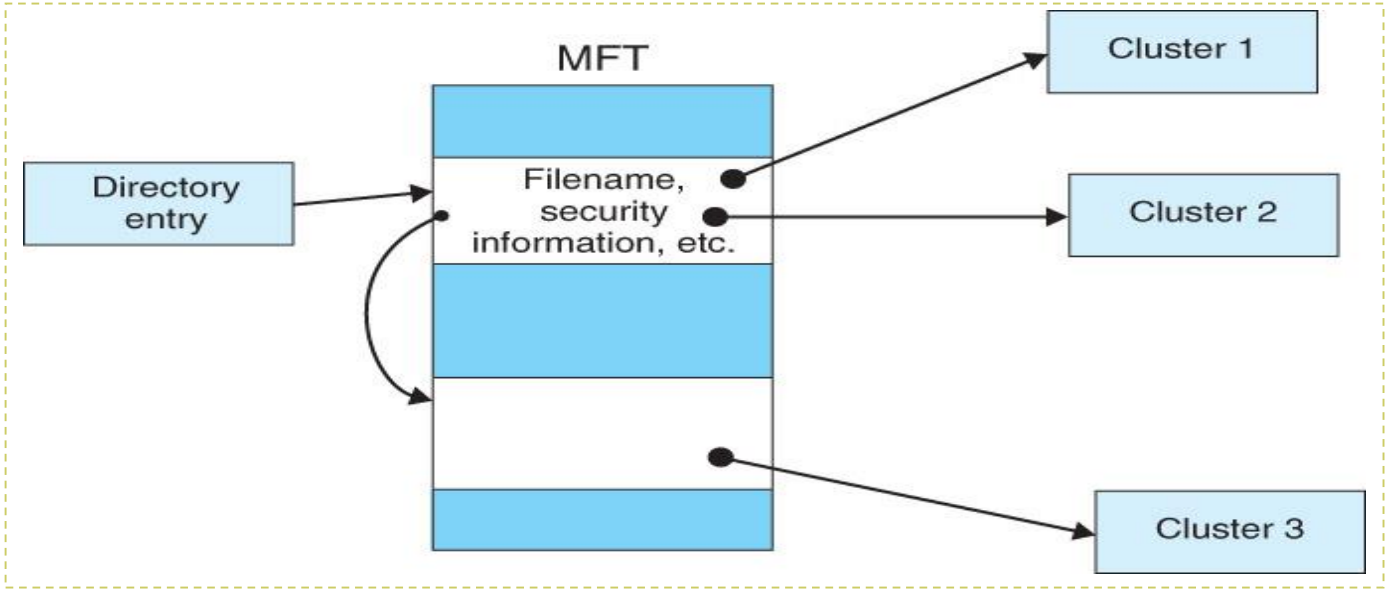
2	42	w	n		f	o	属性		校验和	X	
		77 00	6E 00	2E 00	66 00	6F 00	0F	00	07	78 00	
	00 00	FF FF	FF FF	FF FF	FF FF	FF FF	00 00		FF FF		FF FF
1	01	T	h	e		q	属性	00	校验和	U	
		54 00	68 00	65 00	20 00	71 00	0F		07	75 00	
	i	c	k		b	00 00		r		O	
	69 00	63 00	6B 00	00	62 00			72 00		6F 00	
0	短 文 件 名					扩展名		属性		创建时间	
	T H	E Q	U I	~ 1		F O X		20			
	创建日期	最后存取日期	00 00	最后修改时间		最后修改日期	第一簇号		文 件 大 小		

File System (NTFS)

- **NTFS卷布局：**

引导扇区	主文件表(MFT)（前16个为系统文件）	文件区	MFT前16个元数据文件备份	文件区

- Each partition has a **Master File Table (MFT)**:



MFT用数据库记录形式组织，每条记录（MFT表项）长度1K

File System (NTFS)

- MFT由一个个MFT项（也称为文件记录）组成，每个MFT项占用1024字节的空间。
- MFT前16个记录用来存放元数据文件的信息，它们占有固定的位置。
- 每个MFT项的前部几十个字节有着固定的头结构，用来描述本MFT项的相关信息。后面的字节存放着文件属性等。
- 每个文件或目录的信息都包含在MFT中，每个文件或目录至少有一个MFT项。

MFT表项信息

ReFS文件系统

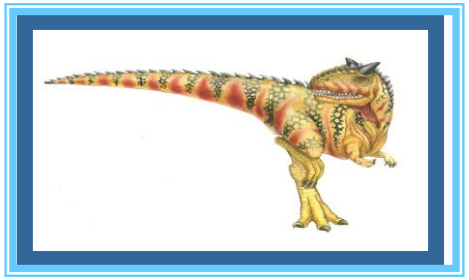
- ReFS (复原文件系统，Windows server版) 是 Microsoft 的最新文件系统，旨在最大限度地提高数据可用性、跨不同的工作负荷高效地扩展到大型数据集，并提供数据完整性，使其能够恢复损坏。它旨在解决存储方案的扩展集问题以及为将来的革新打造基础。

ReFS与NTFS

功能	ReFS	NTFS
最大文件名称长度	255 个 Unicode 字符	255 个 Unicode 字符
最大路径名称长度	32K Unicode 字符	32K Unicode 字符
文件大小上限	35 PB (pb)	256 TB
最大卷大小	35 PB	256 TB

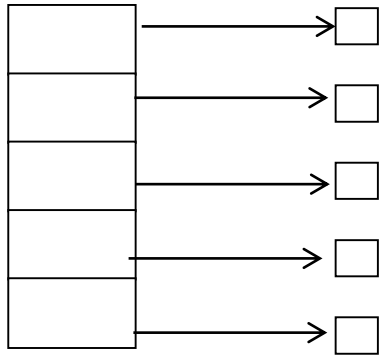
复原文件系统（ReFS）概述 | Microsoft Docs

(三) 索引分配(indexed allocation)



indexed allocation

- 将所有的数据块指针集中到索引块中
  - 索引块中的第i个条目指向文件的第i块。
  - 目录条目包括索引块的地址

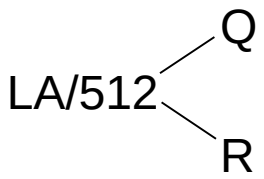


index table

- 索引分配支持直接访问，且没有外部碎片问题
- 索引块本身可能会浪费空间
  - 链接方案：一个索引块通常为一个磁盘块。对于大文件，可以将多个索引块链接起来。
  - 多层索引：类似于内存的间接寻址方式（一级、二级间接...）
  - 组合方案：如Unix的inode

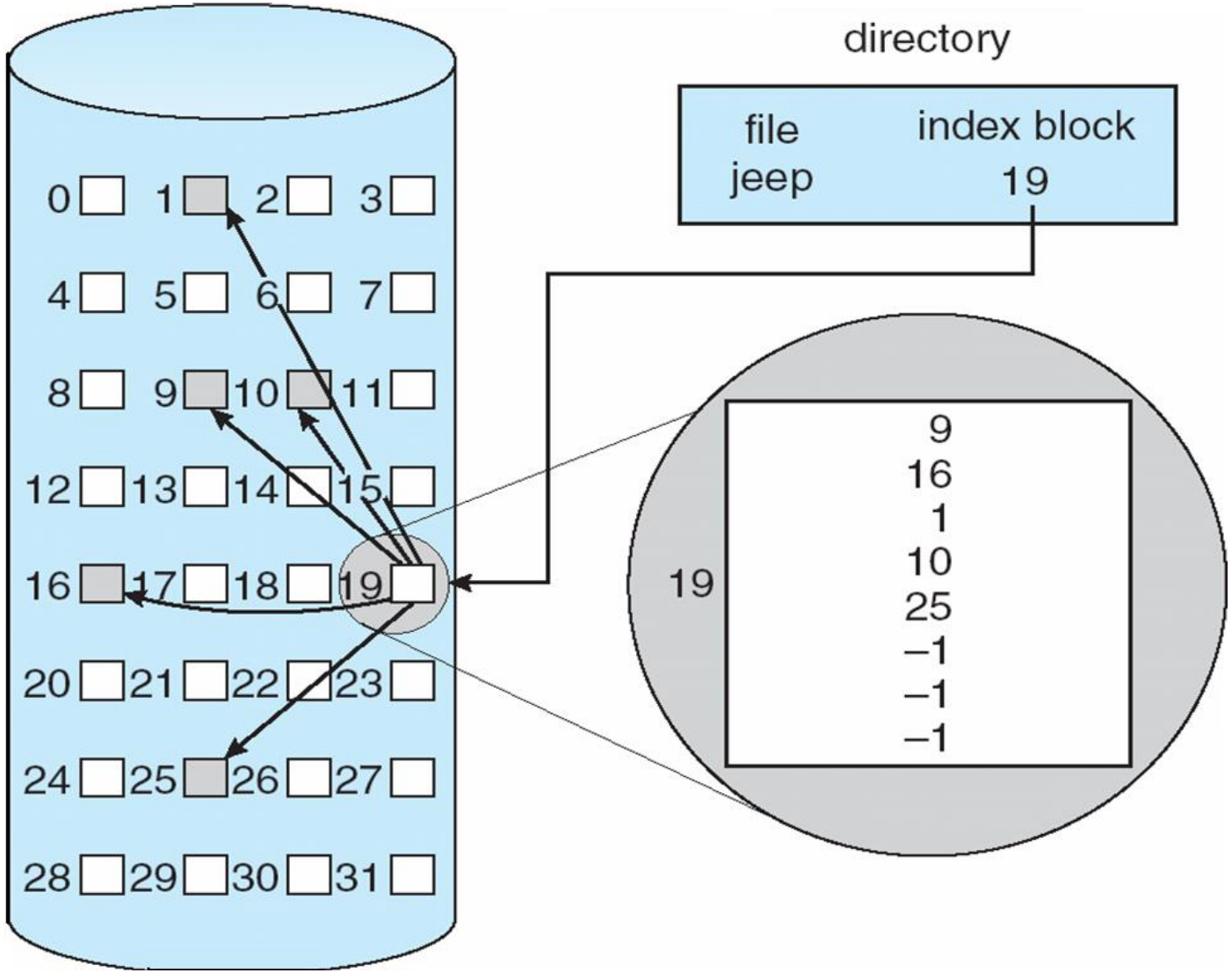
Indexed Allocation (Cont.)

- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table



Q = displacement into index table
R = displacement into block

Example of Indexed Allocation



Indexed Allocation – Mapping (Cont.)

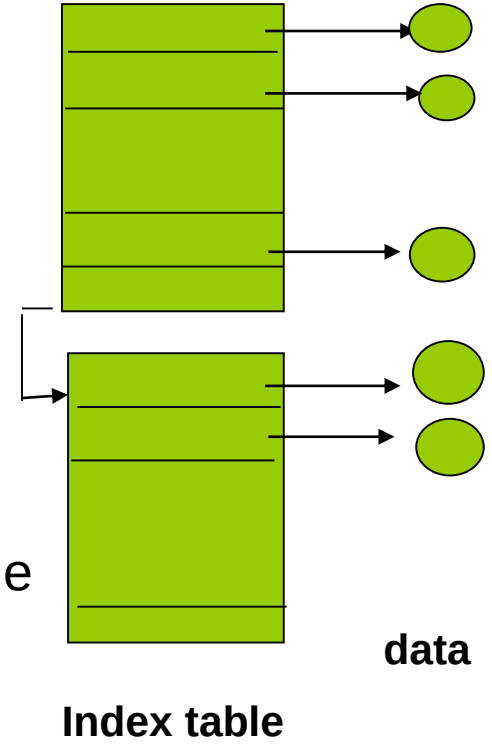
- Mapping from logical to physical in a file of unbounded length (block size of 512 words)
- Linked scheme （链接索引）– Link blocks of index table (no limit on size)

LA / (512 x 511) → Q_1, R_1

Q_1 = block of index table
R_1 is used as follows:

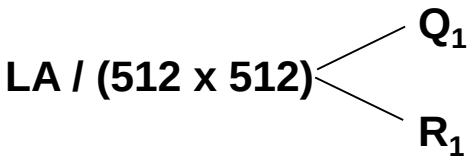
R_1 / 512 → Q_2, R_2

Q_2 = displacement into block of index table
R_2 displacement into block of file:

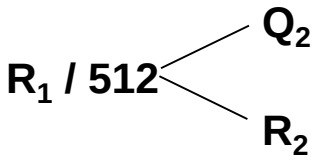


Indexed Allocation – Mapping (Cont.)

- Two-level index (maximum file size is 512^3) 二级索引

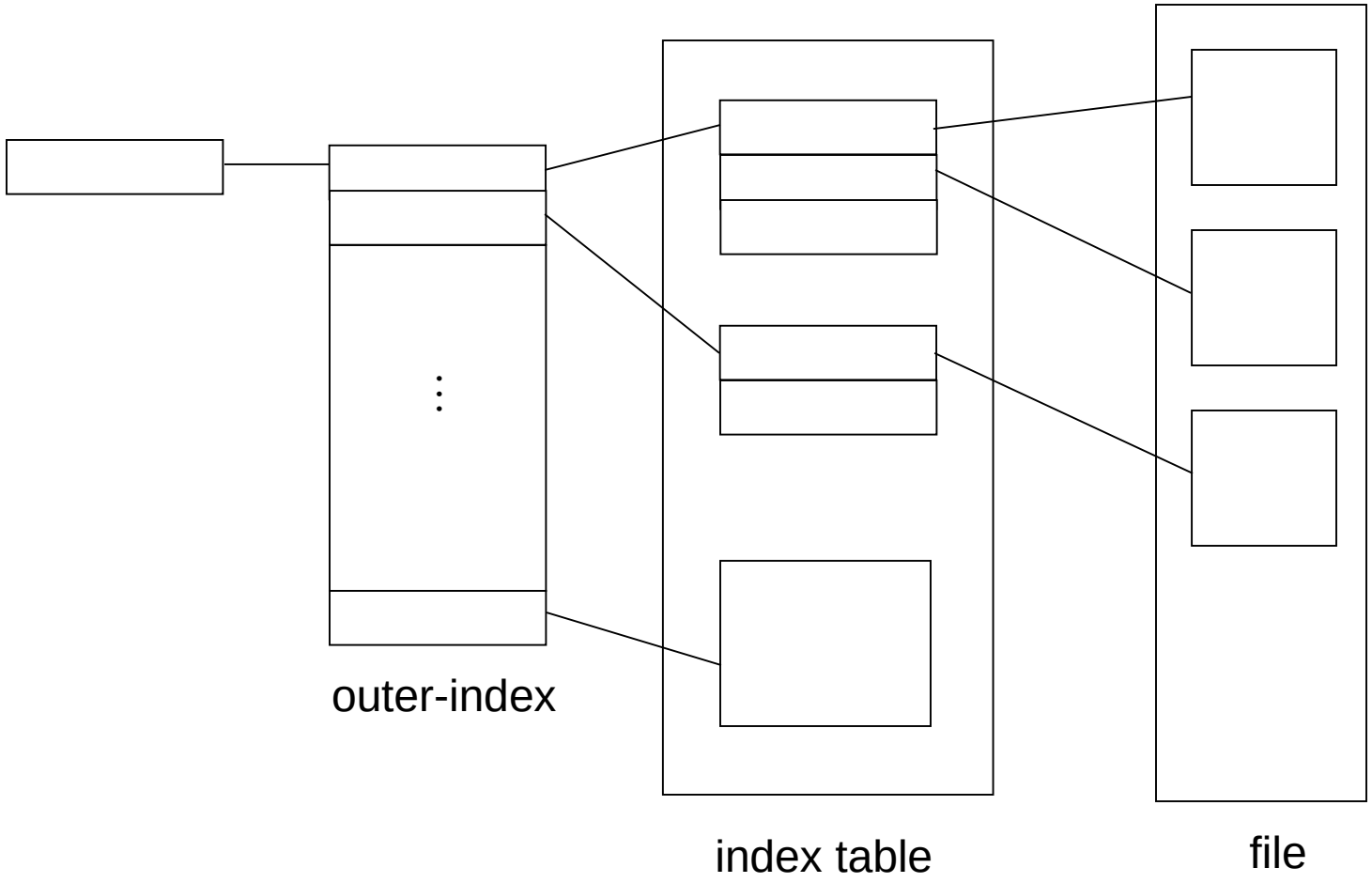


Q_1 = displacement into outer-index
R_1 is used as follows:

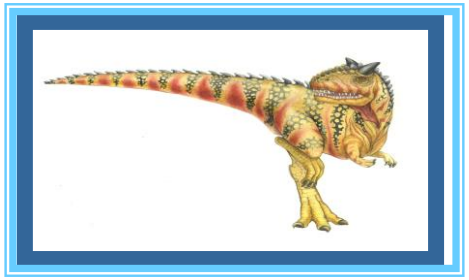


Q_2 = displacement into block of index table
R_2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)

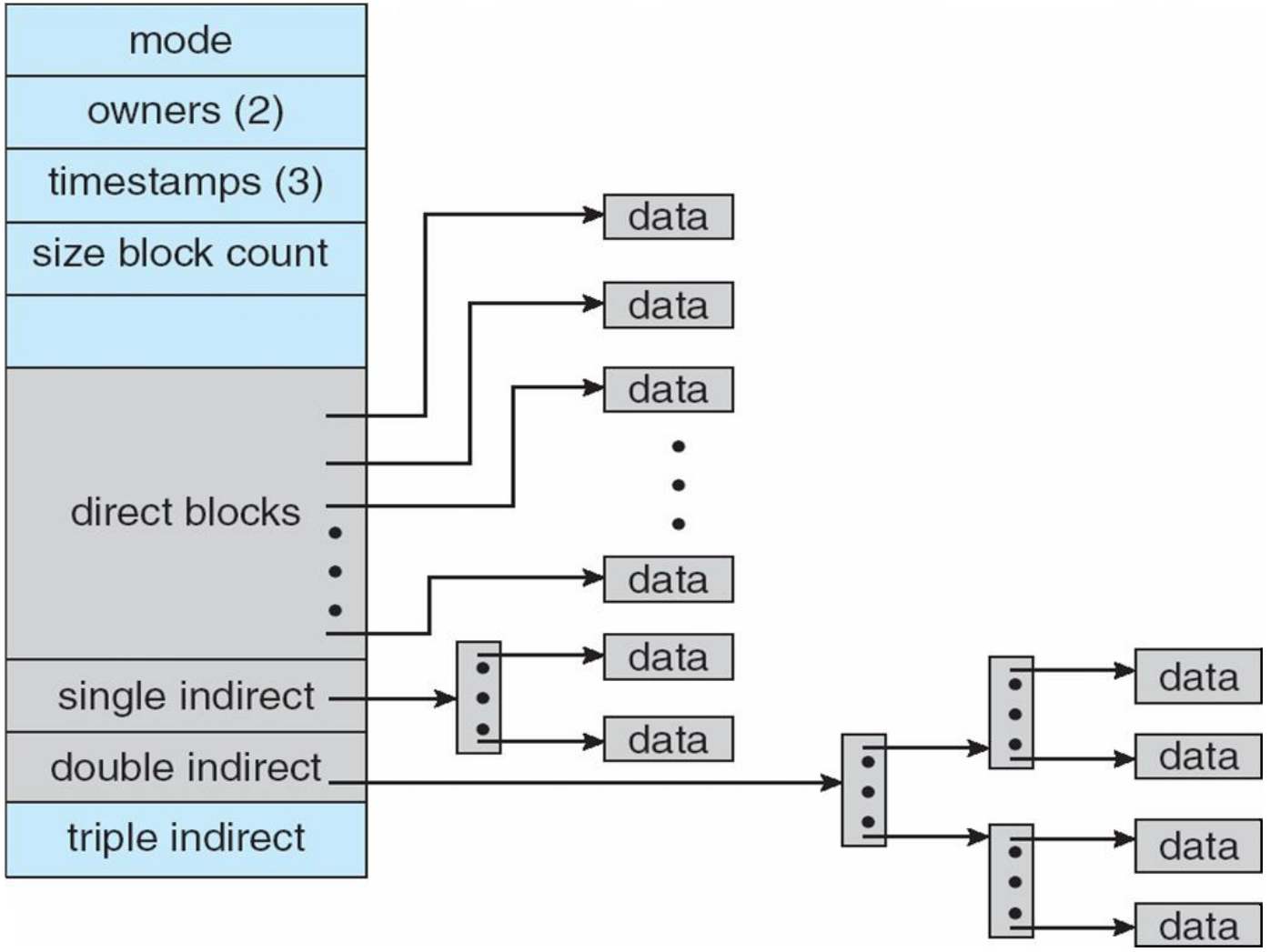


四、 UNIX、Linux直接间接混合分配方法



Combined Scheme: UNIX UFS (4K bytes per block)

- Linux ext2/ext3



UNIX、Linux直接间接混合分配方法

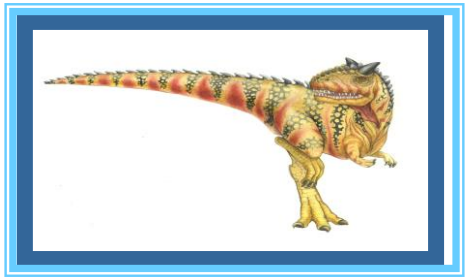
- 由于80%以上文件是小文件，为了解决能高速存取小文件和管理大文件的矛盾，UNIX将直接寻址、一级索引、二级索引和三级索引结合起来，形成混合寻址方式，如下图所示。
- 在Linux ext2的索引结点中设有15个地址项，它们把所存的地址项分成两类，其中最后三个地址项分别是一级索引、二级索引和三级索引的指针，而前面12个为直接寻址的地址项，即存放文件逻辑块第0—11块的盘块号。
- 如每个盘块大小为4KB时，当文件不大于48KB时，便可直接从索引结点中读出该文件全部盘块号，这样读小文件时速度快；如文件大于 48KB时，系统再逐步增加一级索引、二级索引和三级索引，这样最大管理的文件为48KB＋4MB＋4GB＋4TB，达到管理大文件的目标。

实例

- 一个文件系统中有一个20MB大文件和一个20KB小文件,当分别采用连续、链接、链接索引、二级索引和LINUX 分配方案时，每块大小为4096B,每块地址用4B表示，问:
  - （1）各文件系统管理的最大的文件是多少?
  - （2）每种方案对大、小两文件各需要多少专用块来记录文件的物理地址(说明各块的用途)？
  - （3）如需要读大文件前面第5.5KB的信息和后面第（16M＋5.5KB）的信息，则每个方案各需要多少次盘I/O操作?

答案

11.5 Free-Space Management

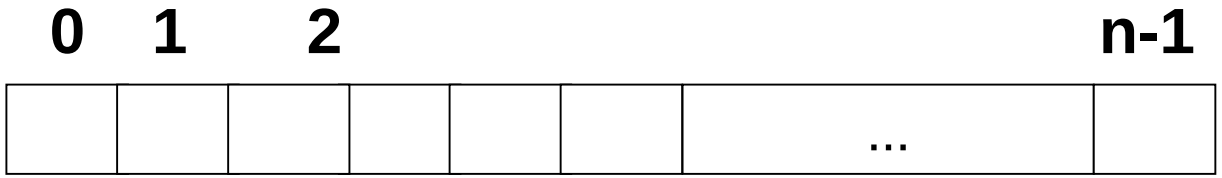


空闲空间管理

- 为了记录空闲磁盘空间，系统需要维护一个空闲空间链表，它记录了所有空闲磁盘空间，即未分配给文件或目录的空间。（不一定以链表的方式实现）
- 常用方法：
  - 位图
  - 链表
  - 分组
  - 计数

Free-Space Management (Cont.)

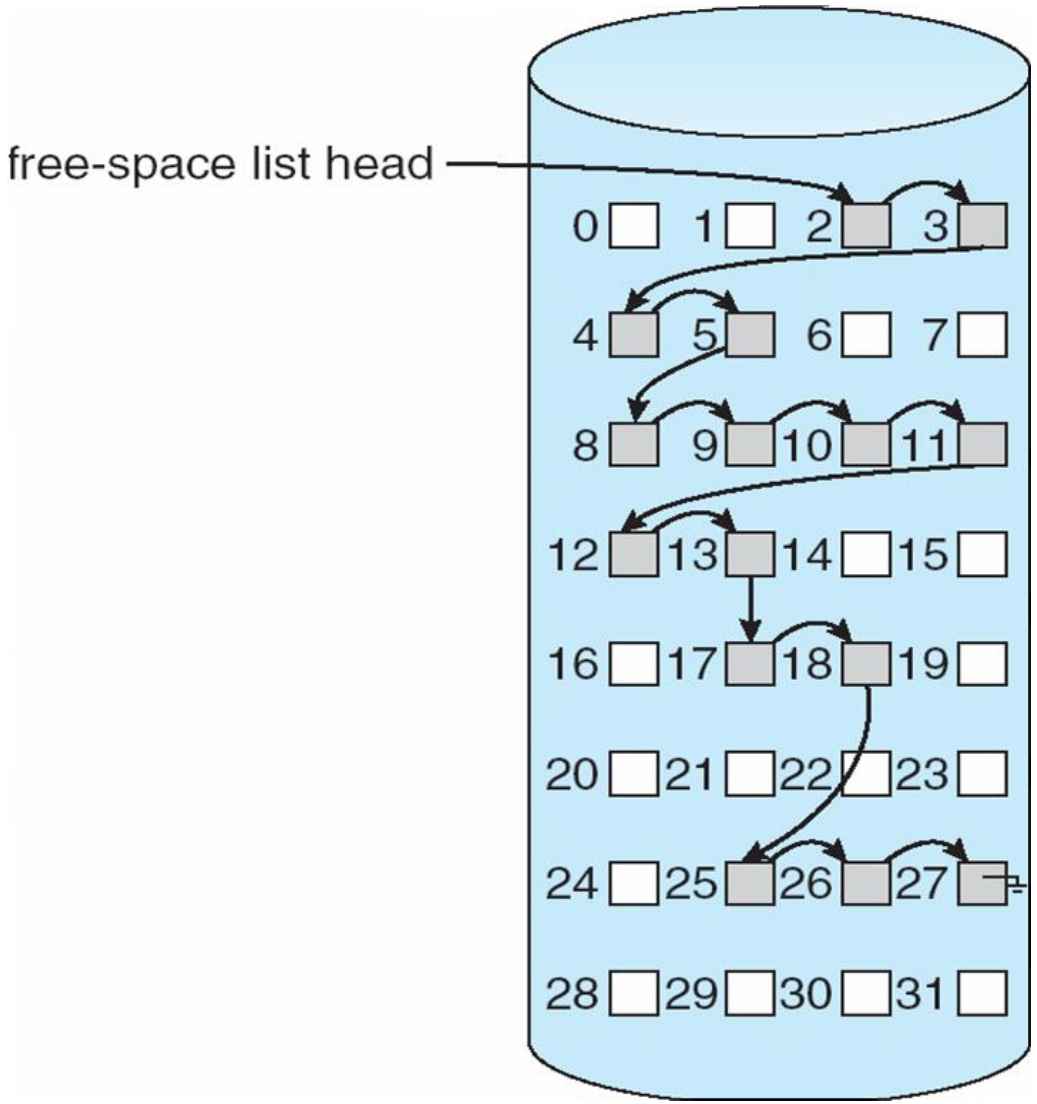
- **位图**（n块）（位向量）
  - bit[i] = 0 ➔ block[i]空闲
  - bit[i] = 1 ➔ block[i]被占用
  - 第一个空闲块的计算：
    - 一个字的位数 × 值为1的字数 + 第一个值为0的位的偏移
  - 位图需要额外的空间
    - 设块大小为2^{12}字节
    - 磁盘大小为2^{30}字节 (1GB)
    - N = 2^{30} / 2^{12} = 2^{18} (即32K bytes)
  - 容易得到连续的文件
  - Windows、Liunx、Mac OS使用。



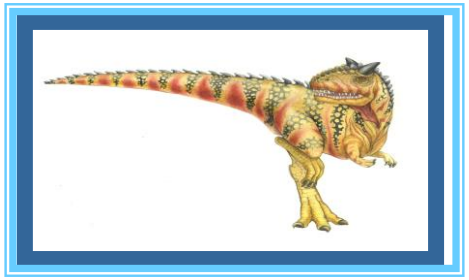
Free-Space Management (Cont.)

- **链表**(空闲链表)：将所有空闲磁盘块用链表连接起来，并将指向第一空闲块的指针保存在磁盘的特殊位置，同时也缓存在内存中。
  - 不易得到连续空间
  - 没有空间浪费
- **分组**：将n个空闲块的地址存在第一个空闲块中，而最后一块包含另外n个空闲块的地址，如此继续。--Unix 成组连接法(next page)
- **计数**：通常，有多个连续块需要同时分配或释放。因此，可以记录第一块的地址和紧跟第一块的连续的空闲块的数量n。---空闲表法

Linked Free Space List on Disk



11.6 Efficiency and Performance



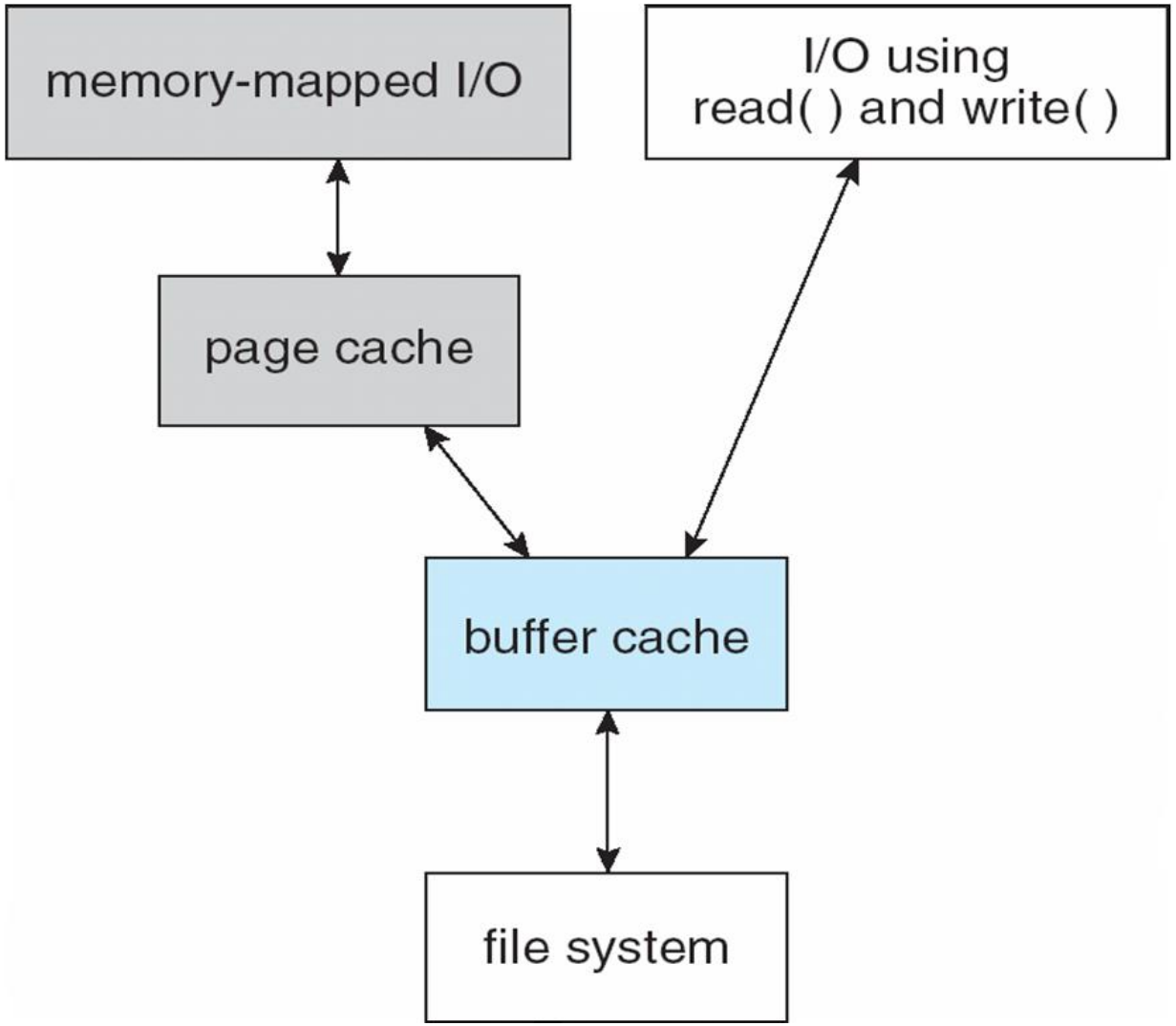
Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks
  - free-behind and read-ahead – techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk

Page Cache

- A page cache caches pages rather than disk blocks using virtual memory techniques
- Memory-mapped I/O uses a page cache
- Routine I/O through the file system uses the buffer (disk) cache
- This leads to the following figure

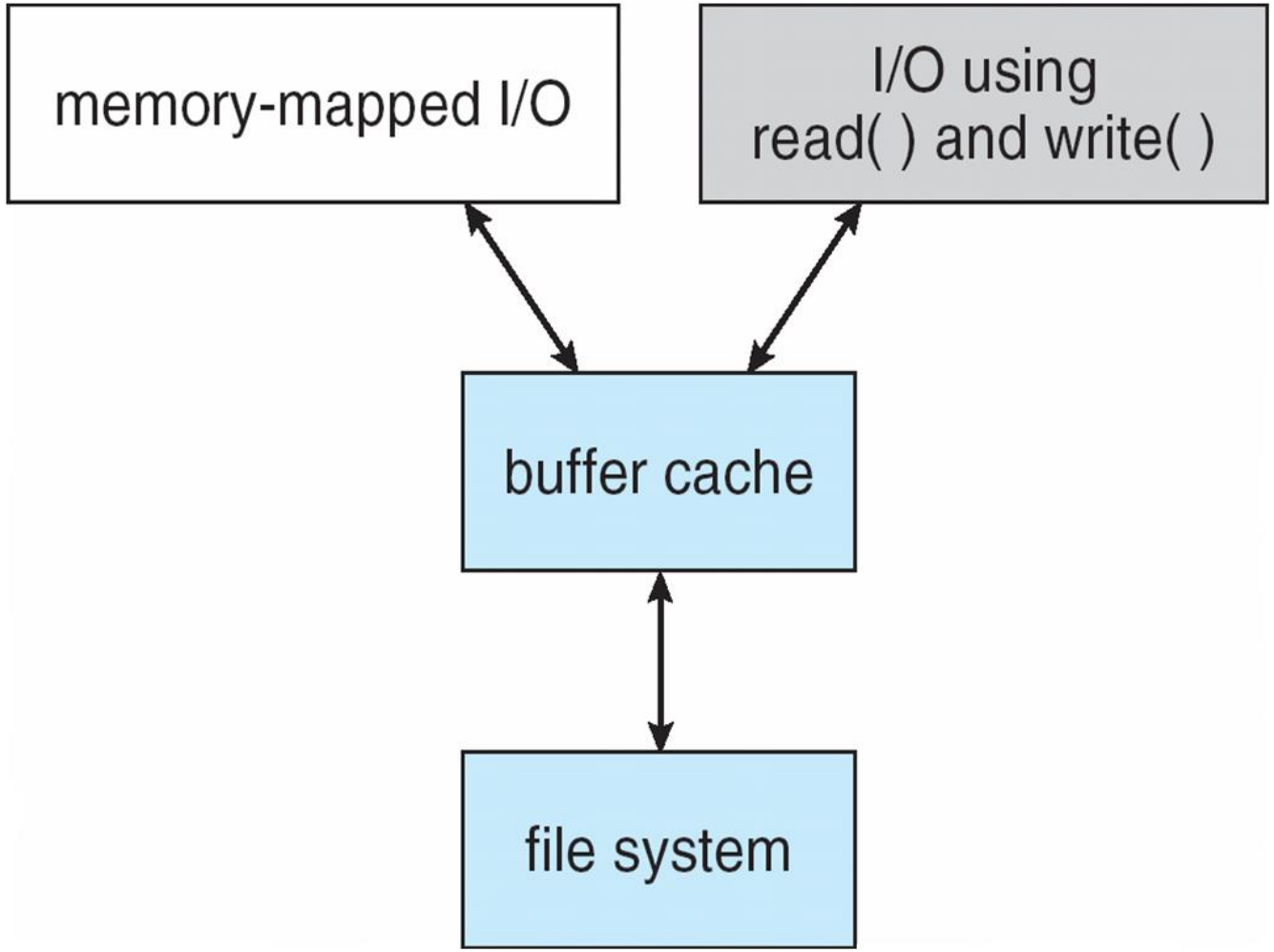
I/O Without a Unified Buffer Cache



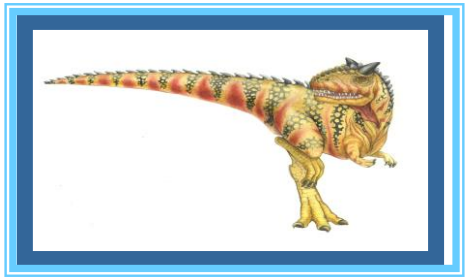
Unified Buffer Cache

- A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O

I/O Using a Unified Buffer Cache



11.7 Recovery



Recovery

- Consistency checking – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
- Use system programs to back up data from disk to another storage device (magnetic tape, other magnetic disk, optical)
- Recover lost file or disk by restoring data from backup

11.8 Log Structured File Systems

- Log structured (or journaling) file systems record each update to the file system as a transaction
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log
  - However, the file system may not yet be updated
- The transactions in the log are asynchronously written to the file system
  - When the file system is modified, the transaction is removed from the log
- If the file system crashes, all remaining transactions in the log must still be performed

*11.9 The Sun Network File System (NFS)

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)

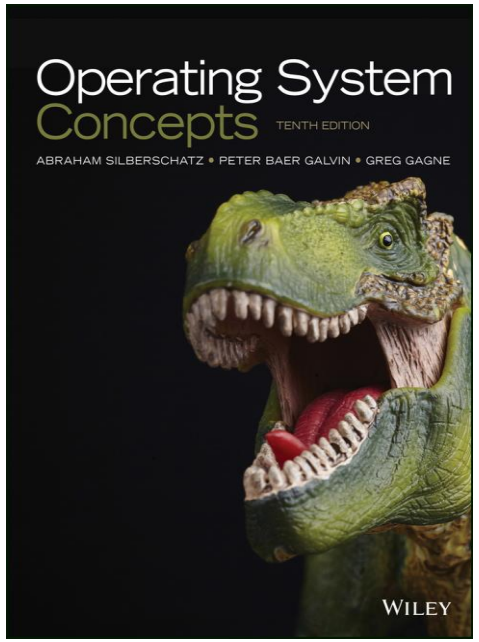
- The implementation is part of the Solaris and SunOS operating systems running on Sun workstations using an unreliable datagram protocol (UDP/IP protocol and Ethernet

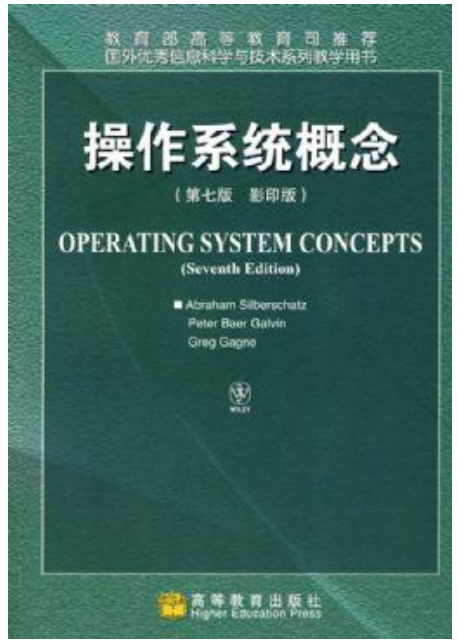
HOMEWORK

- 学在浙大
- 习题分析

Reading Assignments

- Read for this week:
  - Chapters 11
    of the text book:
- Read for next week:
  - Chapters 12
    of the text book:





End of Chapter 11