第六章

文件管理

主 要 内 容

6.1 文件系统概述

6.2 文件的逻辑结构与物理结构

6.3 文件存储空间管理

6.4 文件系统的目录管理

文件系统概述

什么是文件

文件是具有文件名的一组相关信息的集合。

通常，文件由若干个记录组成。

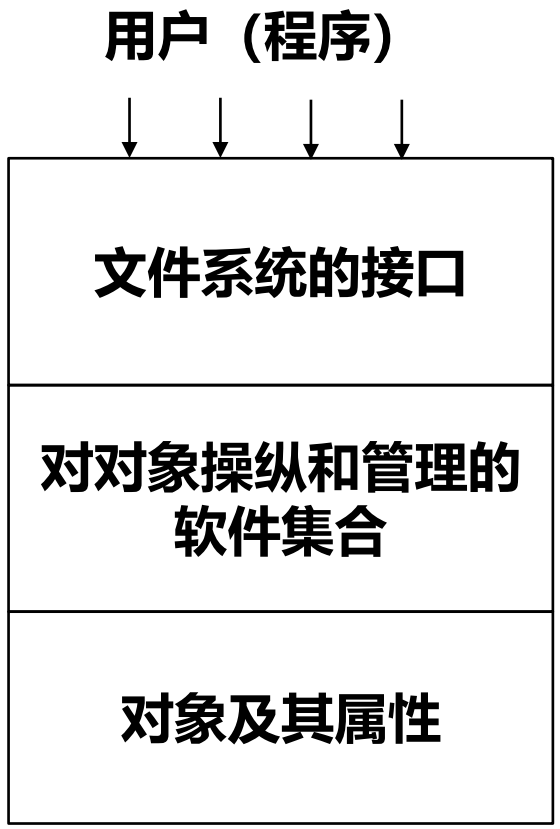
系统或用户可以将一个程序或一组数据命名为一个文件。

操作系统中与管理文件有关的软件和数据统称为**文件管理系统**（简称文件系统）。

从系统角度，文件系统是对文件的存储空间进行组织、分配，负责文件的存储并对存储的文件进行保护、检索的系统。

从用户角度，文件系统主要实现了对文件的按名存取。

文件系统概述



命令接口：用户与文件系统（命令）
程序接口：用户程序与文件系统（系统调用）

文件存储空间的管理、文件目录的管理
文件逻辑地址到物理地址的转换、
文件读写管理、文件的共享与保护等

目录：方便文件的存取和检索（*文件名→文件物理地址*）
文件：　文件管理的直接对象
存储空间：提高利用率和文件存取速度

文件系统概述

文件的类型

按用途分	系统文件：	系统软件构成的文件	可调用
	库文件：	标准或常用例程	可调用
	用户文件：	用户源代码、目标文件、可执行文件、数据文件……	
按存取控制分	只读文件		
	读写文件		
	可执行文件		
按组织形式分	普通文件：	ASCII码或二进制码组成的字符文件	
	目录文件：	由文件目录构成	
	特殊文件：	系统的各类I/O设备	
按文件中的信息流性质分	输入文件		
	输出文件		
	输入/输出文件		

文件系统概述

1. 按用户要求创建或删除文件；
2. 按用户要求进行文件读写；
3. 用户使用文件符号名实现文件访问，文件的物理组织对用户是透明的；
4. 管理文件存储空间，自动分配，建立文件逻辑结构以及物理结构之间的映照关系；
5. 共享和保密。

UNIX文件系统

概况

物理结构：混合索引结构（三级索引结构）

空闲盘块的管理：成组链接法

目录结构：带勾连的树形结构

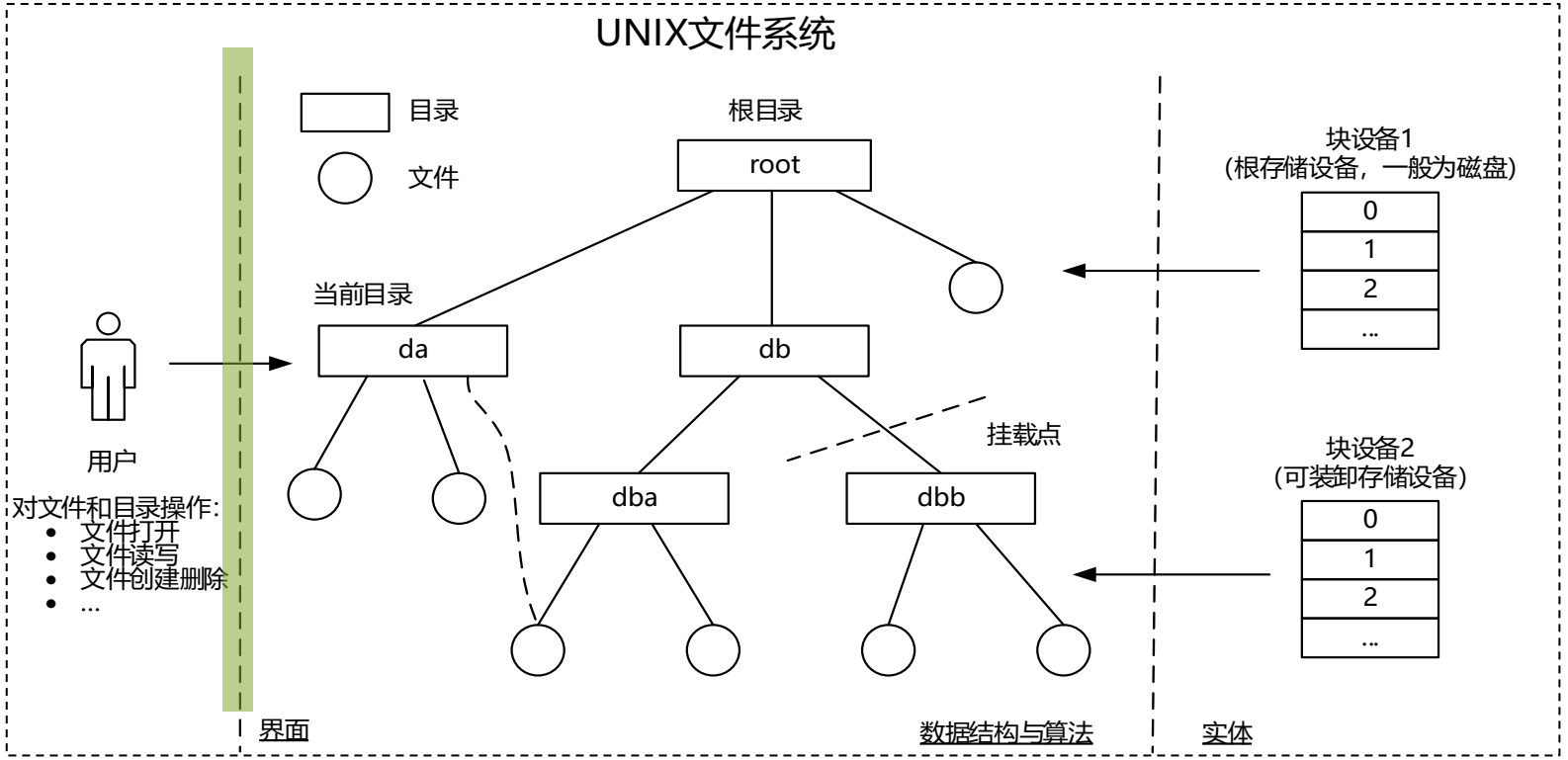
包括：基本文件系统和可装卸的子文件系统

主要数据结构：

外存：　文件索引节点　目录结构

　　文件存储资源管理块

内存：　内存文件打开结构



文件的创建

fd = creat (name, mode)

- fd：打开文件标识数
- name：新文件名
- mode：新文件的工作方式，包括其文件类型和用户对新文件的访问权限

1. 目录搜索该文件是否存在

如果不存在，但之前的路径都正确，则创建该文件，长度为0

如果存在，则将原文件内容全部删除

```
static const unsigned int IREAD = 0x100;  /* 对文件的读权限 */
static const unsigned int IWRITE = 0x80;  /* 对文件的写权限 */
static const unsigned int IEXEC = 0x40;   /* 对文件的执行权限 */
/* 文件主对文件的读、写、执行权限 */
static const unsigned int IRWXU = (IREAD|IWRITE|IEXEC);
/* 文件主同组用户对文件的读、写、执行权限 */
static const unsigned int IRWXG = ((IRWXU) >> 3);
/* 其他用户对文件的读、写、执行权限 */
static const unsigned int IRWXO = ((IRWXU) >> 6);
```

文件主　文件主同组用户　其他用户

最低9位：R W E　R W E　R W E

2. 设置文件访问权限

例：fd = creat("/usr/Jessy", 0666);

文件的打开和关闭

fd = open (name, mode)

- fd：文件标识符
- name：文件名
- mode：打开方式

```
FREAD = 0x1,    /* 读请求类型 */
FWRITE = 0x2,   /* 写请求类型 */
```

即：1-只读；2-只写；3-读写

\+

用户类型

(文件主？同组？其他？)

1. 目录搜索该文件是否存在

2. 文件访问是否合法

例：fd = open("/usr/Jessy",01);

close (fd)

- fd：文件标识符

所有对文件的操作进行之前，必须先打开文件，获得文件标识符(文件句柄)

文件的创建

fd = creat (name, mode)

- fd：打开文件标识数
- name：新文件名
- mode：新文件的工作方式，包括其文件类型和用户对新文件的访问权限

1. 目录搜索该文件是否存在

如果不存在，但之前的路径都正确，则创建该文件，长度为0

如果存在，则将原文件内容全部删除

3. 以可写的方式打开文件

```
static const unsigned int IREAD = 0x100;  /* 对文件的读权限 */
static const unsigned int IWRITE = 0x80;  /* 对文件的写权限 */
static const unsigned int IEXEC = 0x40;   /* 对文件的执行权限 */
/* 文件主对文件的读、写、执行权限 */
static const unsigned int IRWXU = (IREAD|IWRITE|IEXEC);
/* 文件主同组用户对文件的读、写、执行权限 */
static const unsigned int IRWXG = ((IRWXU) >> 3);
/* 其他用户对文件的读、写、执行权限 */
static const unsigned int IRWXO = ((IRWXU) >> 6);
```

文件主　文件主同组用户　其他用户

最低9位：R W E　R W E　R W E

2. 设置文件访问权限

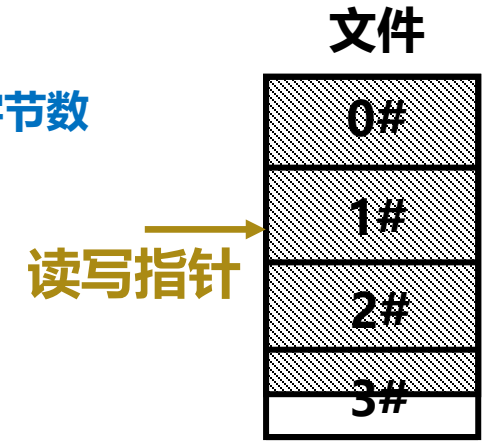
例：fd = creat("/usr/Jessy", 0666);

文件的顺序读写

```
n = read ( fd, buf, nbytes)
```

- n：实际读取字节数
- fd：文件标识符
- buf：进程地址空间中存放读回数据的首址
- nbytes：读取的字节数

例：count = read(fd, data2, 12);



默认：一次读写的起始位置是上次读写结束位置的下一个字节

```
n = write( fd, buf, nbytes)
```

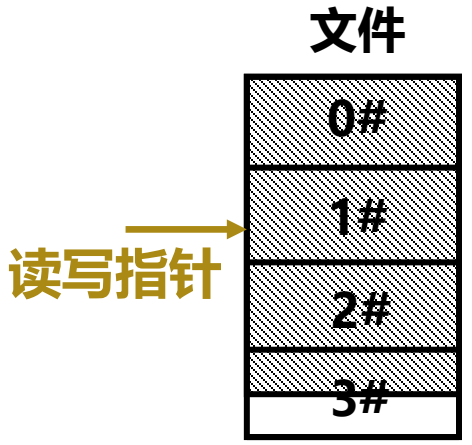
- n：实际写入字节数
- fd：文件标识符
- buf：进程地址空间中存放写入文件数据的首址
- nbytes：写入的字节数

例：count = write(fd, data1, 12);

文件的随机存取　seek (fd, offset, ptrname)

文件标识符　配合调整文件读取位置

例：seek(fd,5,0);



默认：一次读写的起始位置是上次读写结束位置的下一个字节

ptrname = 0：读写指针位置设置为 offset （正）

ptrname = 1：读写指针位置设置为 当前位置 + offset（可正可负）

ptrname = 2：读写指针位置设置 文件结束位置 + offset（负）

文件的勾连与取消

link (oldpath, newpath)

为文件oldpath勾连一个新的路径名newpath

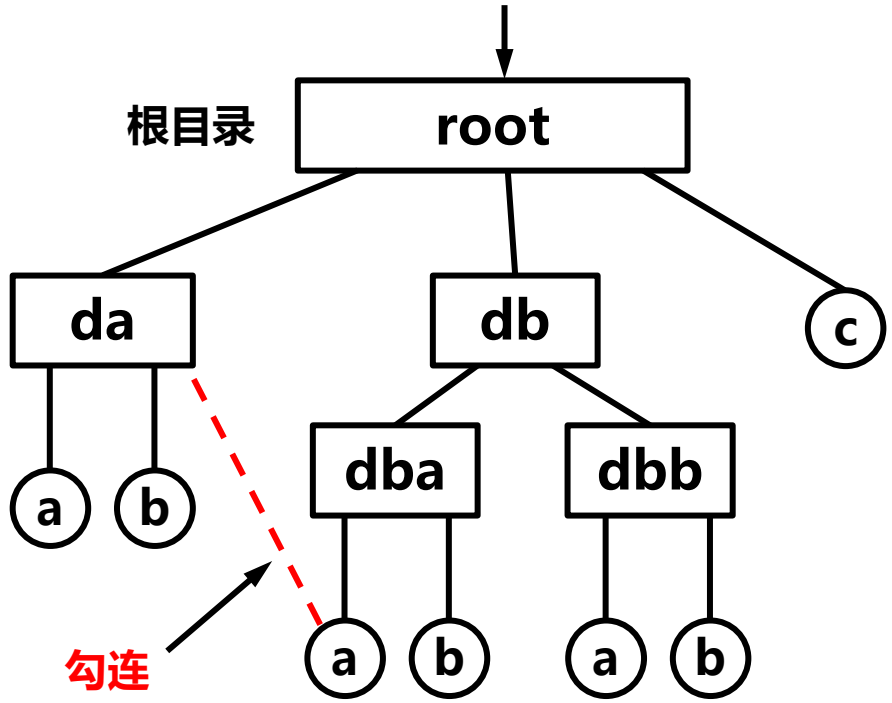
unlink (path)

取消文件的路径名path

例：link("/db/dba/a", "/da/a");

所有路径都是等价的

例：unlink("/db/dba/a");
仍可以通过路径“/da/a”来访问该文件



所有某一文件的最后一条路径被unlink，该文件被物理删除

主 要 内 容

6.1 文件系统概述

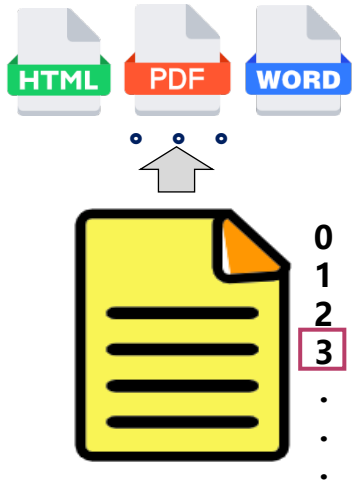
6.2 文件的逻辑结构与物理结构

- 常见的文件物理结构
- UNIX文件的物理结构
- UNIX文件的打开结构
- UNIX文件系统的读写操作

6.3 文件存储空间管理

6.4 文件系统的目录管理

文件的逻辑结构和物理结构



文件的逻辑地址空间是一维的：

<逻辑块号/*bn*>

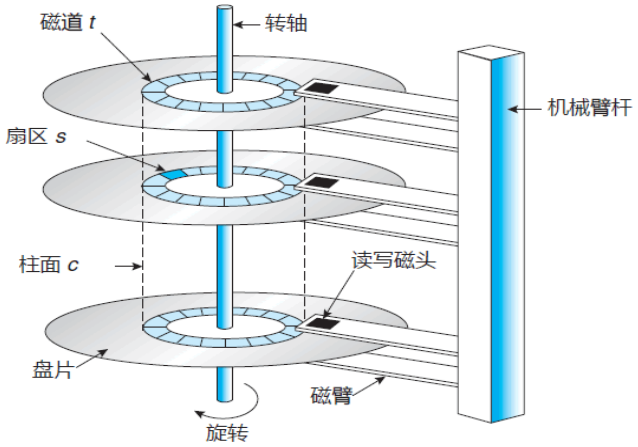
（一个逻辑块大小和物理块相同，从文件起始位置开始）

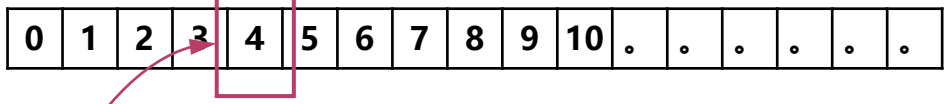
由操作系统提供给上层应用的文件的逻辑地址

逻辑结构：从用户角度（应用程序层）观察到的文件的组织形式，是程序可直接处理的数据及其结构。

例：UNIX操作系统将文件看作顺序的字符流

文件的逻辑结构和物理结构

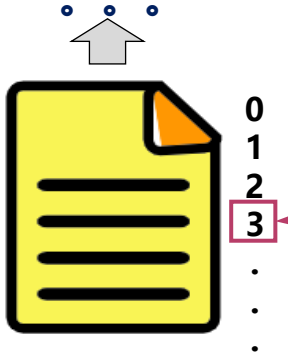




物理结构：文件在存储介质上由操作系统如何保存。

①如何有效利用外存空间?

②如何提高对文件的访问速度?



文件的逻辑地址空间是一维的：

<逻辑块号/bn>

(一个逻辑块大小和物理块相同，从文件起始位置开始)

由操作系统提供给上层应用的文件的逻辑地址

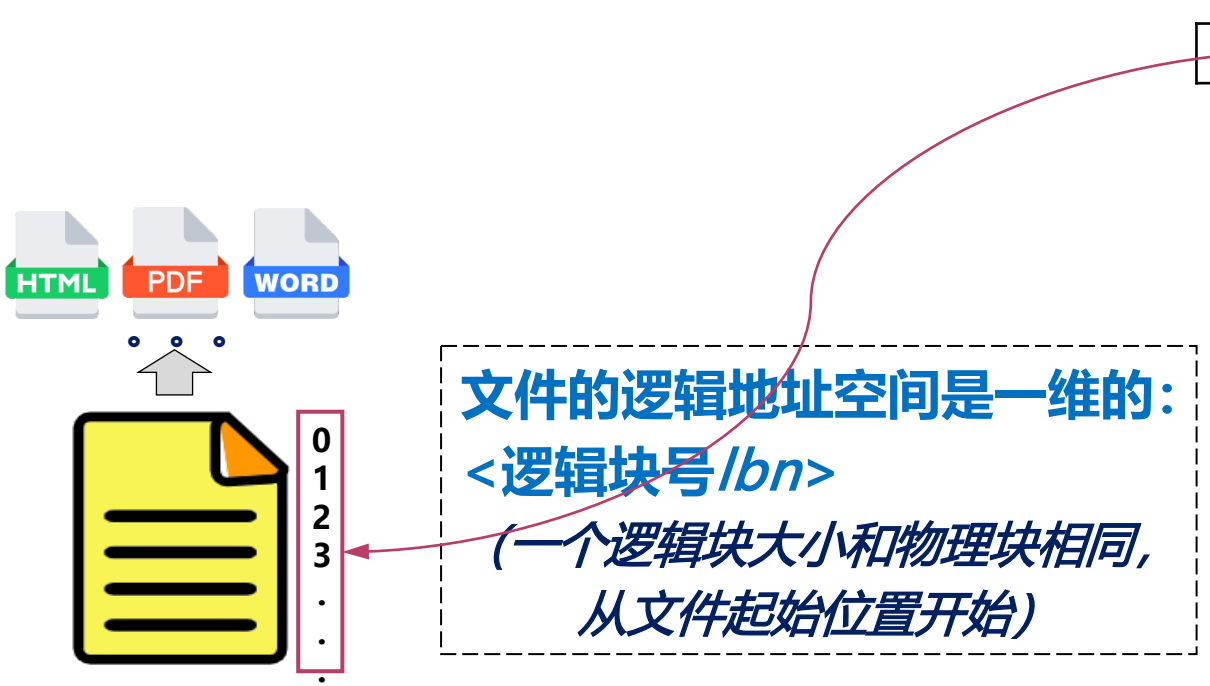
逻辑结构：从用户角度（**应用程序层**）观察到的文件的组织形式，是程序可直接处理的数据及其结构。

例：UNIX操作系统将文件看作顺序的字符流

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

连续结构文件：为每个文件分配一组相邻接的盘块。文件存放在**连续编号的物理块**中。保证了文件中逻辑顺序与占用盘块顺序的一致性。



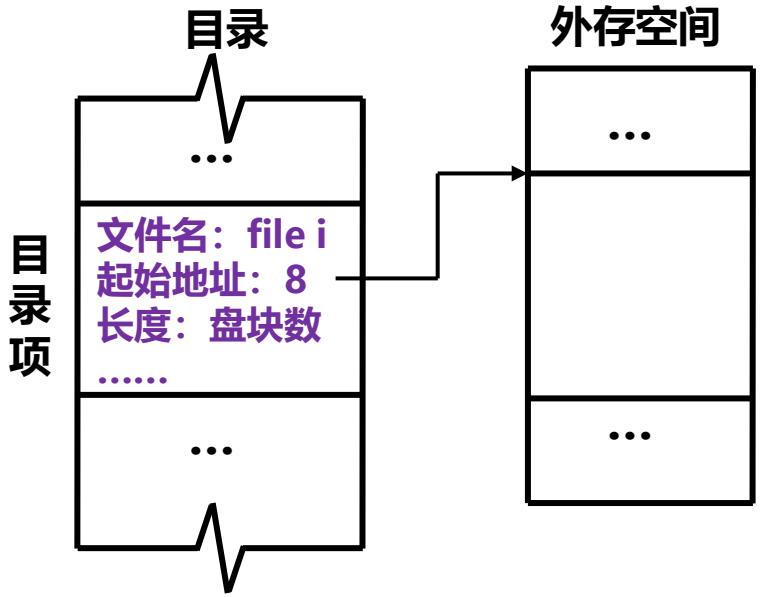
由操作系统提供给上层应用的文件的逻辑地址

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

连续结构文件：为每个文件分配一组相邻接的盘块。文件存放在**连续编号的物理块**中。保证了文件中逻辑顺序与占用盘块顺序的一致性。

建立连续文件时，**用户给出文件最大长度**，系统**分配足够的连续外存空间**，并在目录项中**登记其起始物理地址（起始盘号）及长度（块数）**。



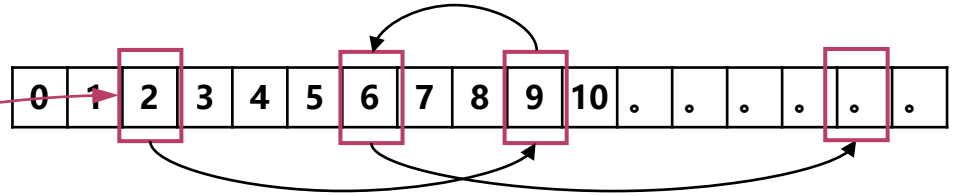
优点：
顺序访问容易且速度快

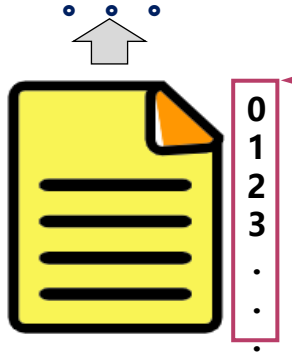
缺点：
① 要求连续的存储空间（磁盘碎片）
② 创建时需确定文件长度，不利于动态增长

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

链接结构文件：非连续的存储结构（将文件装入到多个离散的盘块中）。通过链接指针，将同属于一个文件的多个离散的盘块链接成一个链表。





文件的逻辑地址空间是一维的：

<逻辑块号*lbn*>

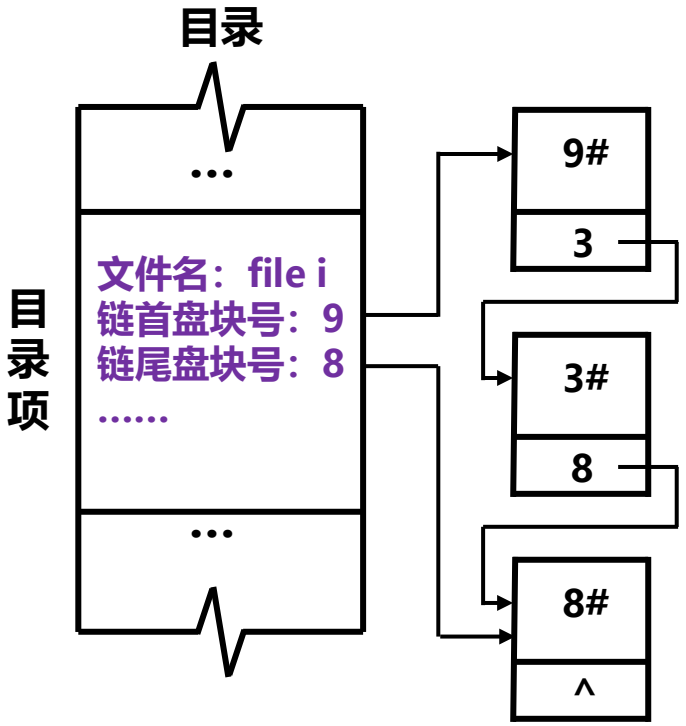
（一个逻辑块大小和物理块相同，从文件起始位置开始）

由操作系统提供给上层应用的文件的逻辑地址

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

链接结构文件：非连续的存储结构（将文件装入到多个离散的盘块中）。通过链接指针，将同属于一个文件的多个离散的盘块链接成一个链表。



优点：

离散分配方式有效利用空间

缺点：

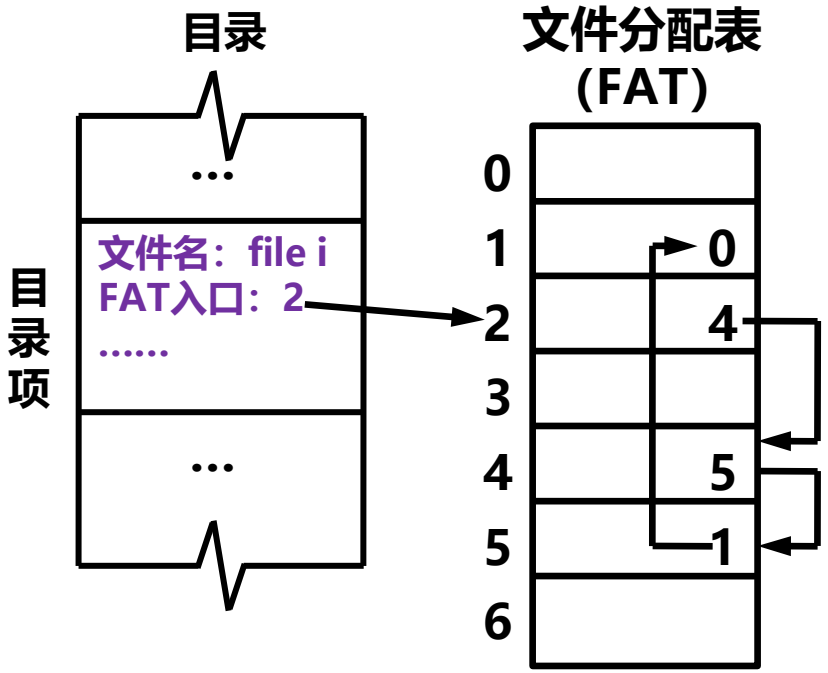
① 适合顺序存取
② 随机存取时有较大难度
③ 可靠性较差（其中一个指针出现问题，其后的文件都将丢失）

隐式链接

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

链接结构文件：非连续的存储结构（将文件装入到多个离散的盘块中）。通过链接指针，将同属于一个文件的多个离散的盘块链接成一个链表。



指针信息显式存放在内存中的一张文件分配表中
(整个磁盘一张)

优点：

① 查找记录的过程在内存进行，显著提高检索速度

② 减少了访问磁盘的次数

缺点：

① 不支持高效的直接存取

② FAT占用较大的存储空间

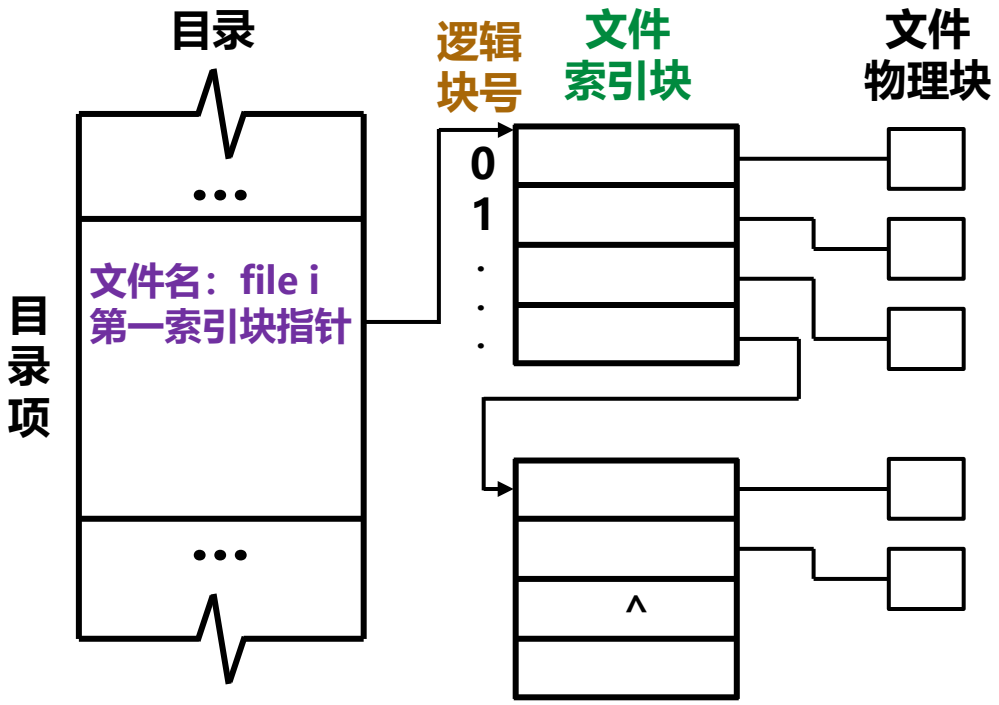
FAT16-FAT32-NTFS都是基于显式链接的文件物理结构

显式链接

文件的逻辑结构和物理结构

物理结构：文件在存储介质上由操作系统如何保存。

索引结构文件：每个文件分配一个索引块（表），建立逻辑块号与物理块号的对照表。



优点：

① 可方便地实现随机存取

缺点：

① 先读索引块，才能获得所需的物理块号

② 增删物理块时，必须对索引表中所有后续项做移位操作

③ 索引块占用一定存储空间

当文件太大，索引块太多时，可建立多级索引。

UNIX 采用混合索引分配方式　i node

主 要 内 容

6.1 文件系统概述

6.2 文件的逻辑结构与物理结构

- 常见的文件物理结构
- UNIX文件的物理结构
- UNIX文件的打开结构
- UNIX文件系统的读写操作

6.3 文件存储空间管理

6.4 文件系统的目录管理

UNIX文件索引节点

Super Block	inode区	文件数据区

文件系统将磁盘从0柱面，0磁道，0扇区拉直，从0#盘块（物理块）开始顺序编号

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)

Super Block	Inode区	文件数据区

i node

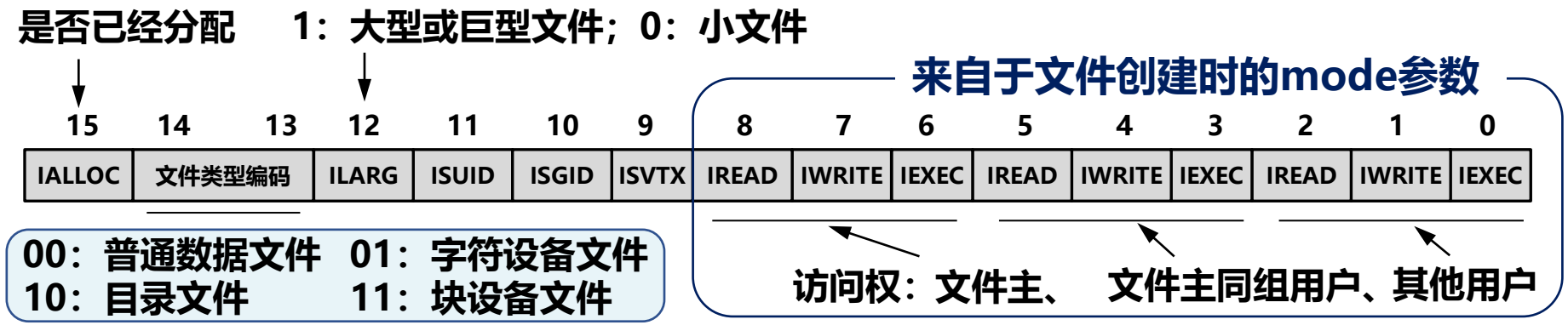
文件索引节点：文件控制块，FCB

```
class DiskInode
{
public:
    unsigned int    d_mode;        /* 状态的标志位/
    int             d_nlink;       /* 该文件在目录树中不同路径名的数量 */
    short           d_uid;         /* 文件所有者的用户标识数 */
    short           d_gid;         /* 文件所有者的组标识数 */
    int             d_size;        /* 文件大小，字节为单位 */
    int             d_addr[10];    /* 文件逻辑块号和物理块号转换的混合索引表 */
    int             d_atime;       /* 最后访问时间 */
    int             d_mtime;       /* 最后修改时间 */
};
```

每个文件在Inode区有一个外存文件控制块DiskInode
(外存索引节点，64个字节)

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)



i node

文件索引节点：文件控制块，FCB

```
class DiskInode
{
public:
    unsigned int    d_mode;     /* 状态的标志位/
    int             d_nlink;    /* 该文件在目录树中不同路径名的数量 */
    short           d_uid;      /* 文件所有者的用户标识数 */
    short           d_gid;      /* 文件所有者的组标识数 */
    int             d_size;     /* 文件大小，字节为单位 */
    int             d_addr[10]; /* 文件逻辑块号和物理块号转换的混合索引表 */
    int             d_atime;    /* 最后访问时间 */
    int             d_mtime;    /* 最后修改时间 */
};
```

每个文件在Inode区有一个外存文件控制块DiskInode
(外存索引节点，64个字节)

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)

Super Block	Inode区	文件数据区

i node

文件索引节点：文件控制块，FCB

```
class DiskInode
{
public:
    unsigned int    d_mode;     /* 状态的标志位/
    int             d_nlink;    /* 该文件在目录树中不同路径名的数量 */
    short           d_uid;      /* 文件所有者的用户标识数 */
    short           d_gid;      /* 文件所有者的组标识数 */
    int             d_size;     /* 文件大小，字节为单位 */
    int             d_addr[10]; /* 文件逻辑块号和物理块号转换的混合索引表 */
    int             d_atime;    /* 最后访问时间 */
    int             d_mtime;    /* 最后修改时间 */
};
```

每个文件在Inode区有一个外存文件控制块DiskInode
(外存索引节点，64个字节)

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)

lbn = Offset / 512

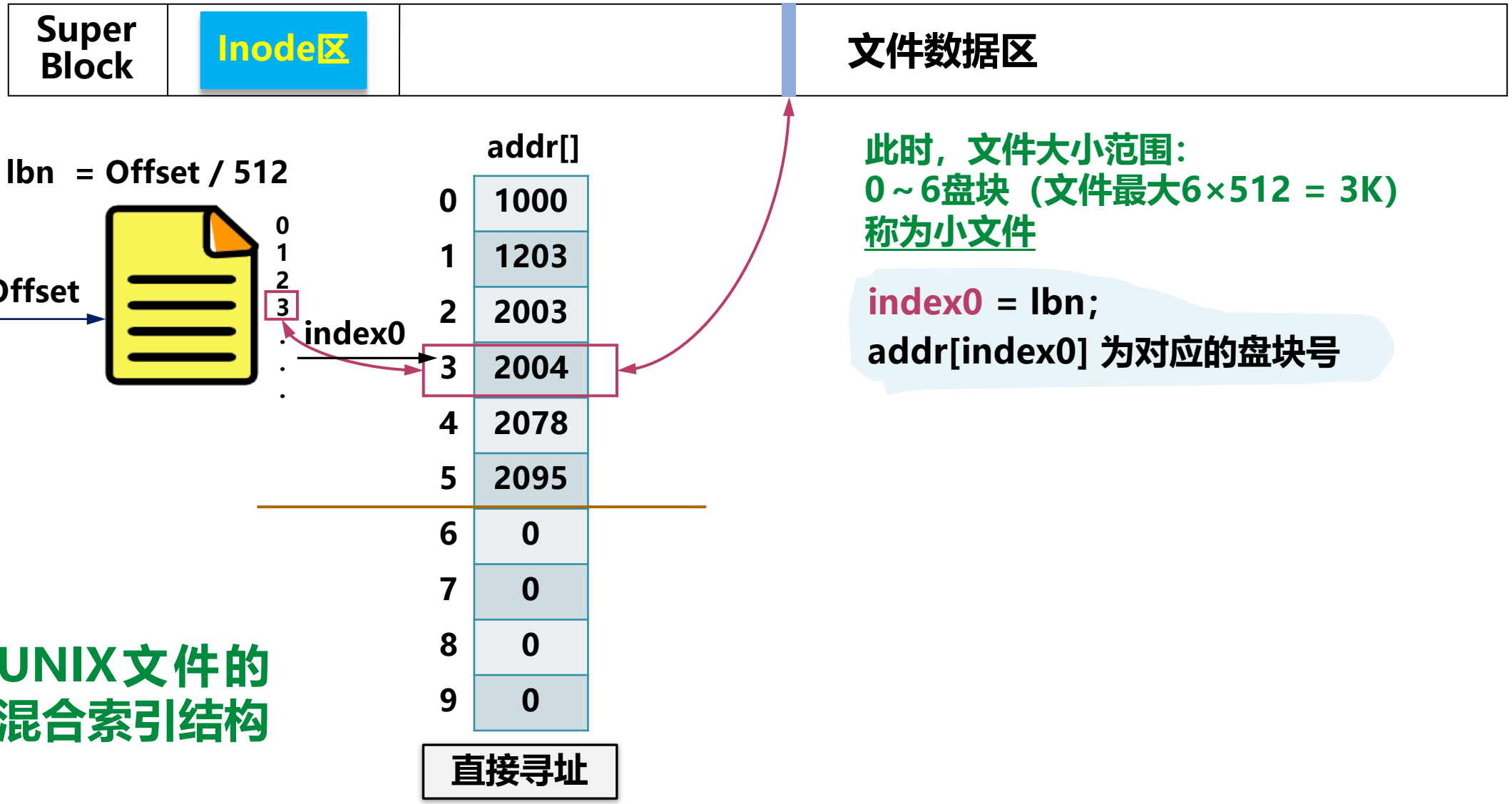
	addr[]
0	1000
1	1203
2	2003
3	2004
4	2078
5	2095
6	0
7	0
8	0
9	0

直接寻址

UNIX文件的混合索引结构

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)



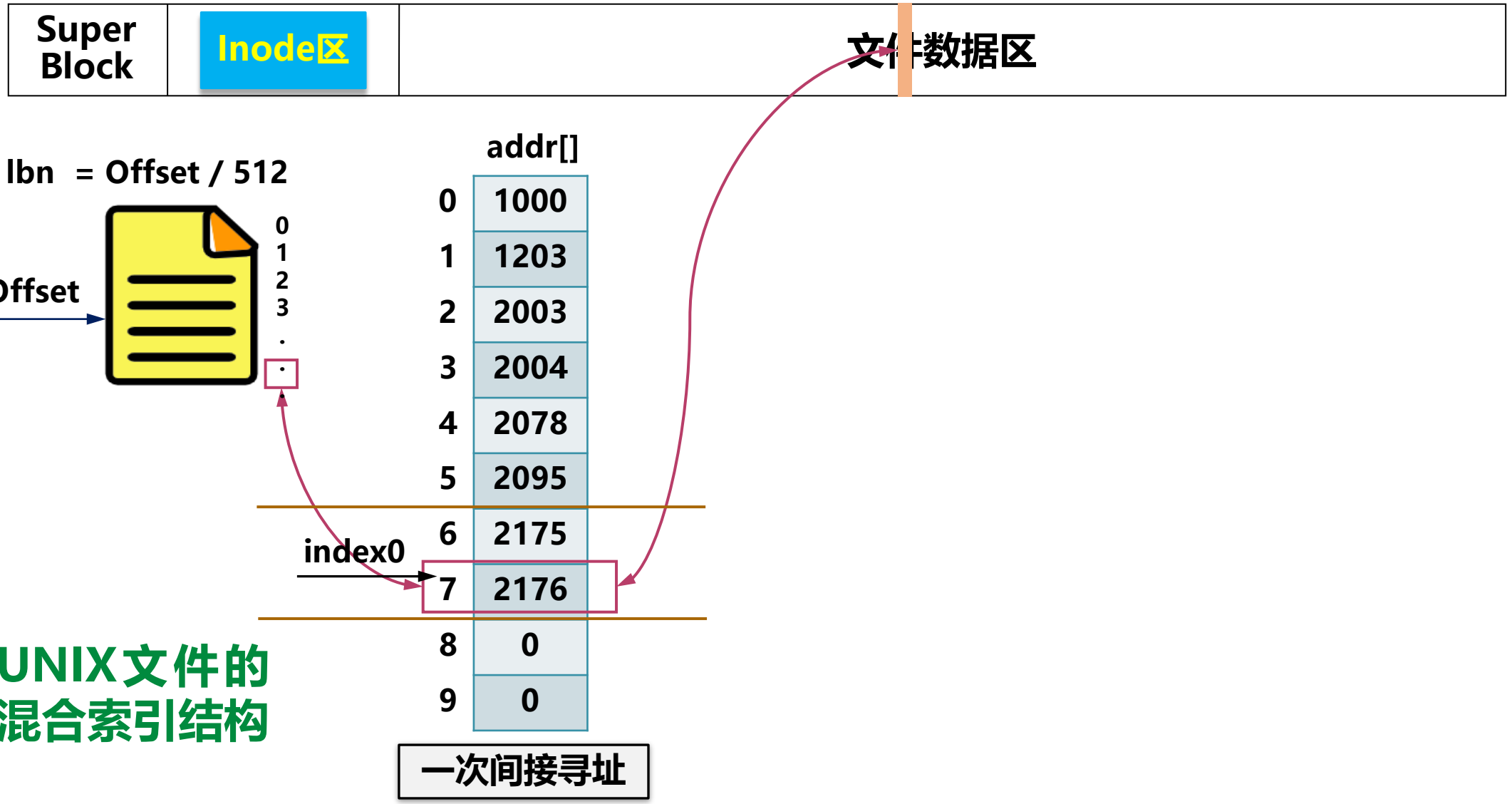
此时，文件大小范围：
0～6盘块（文件最大6×512 = 3K）
称为小文件

index0 = lbn;
addr[index0] 为对应的盘块号

UNIX文件的
混合索引结构

UNIX文件索引节点

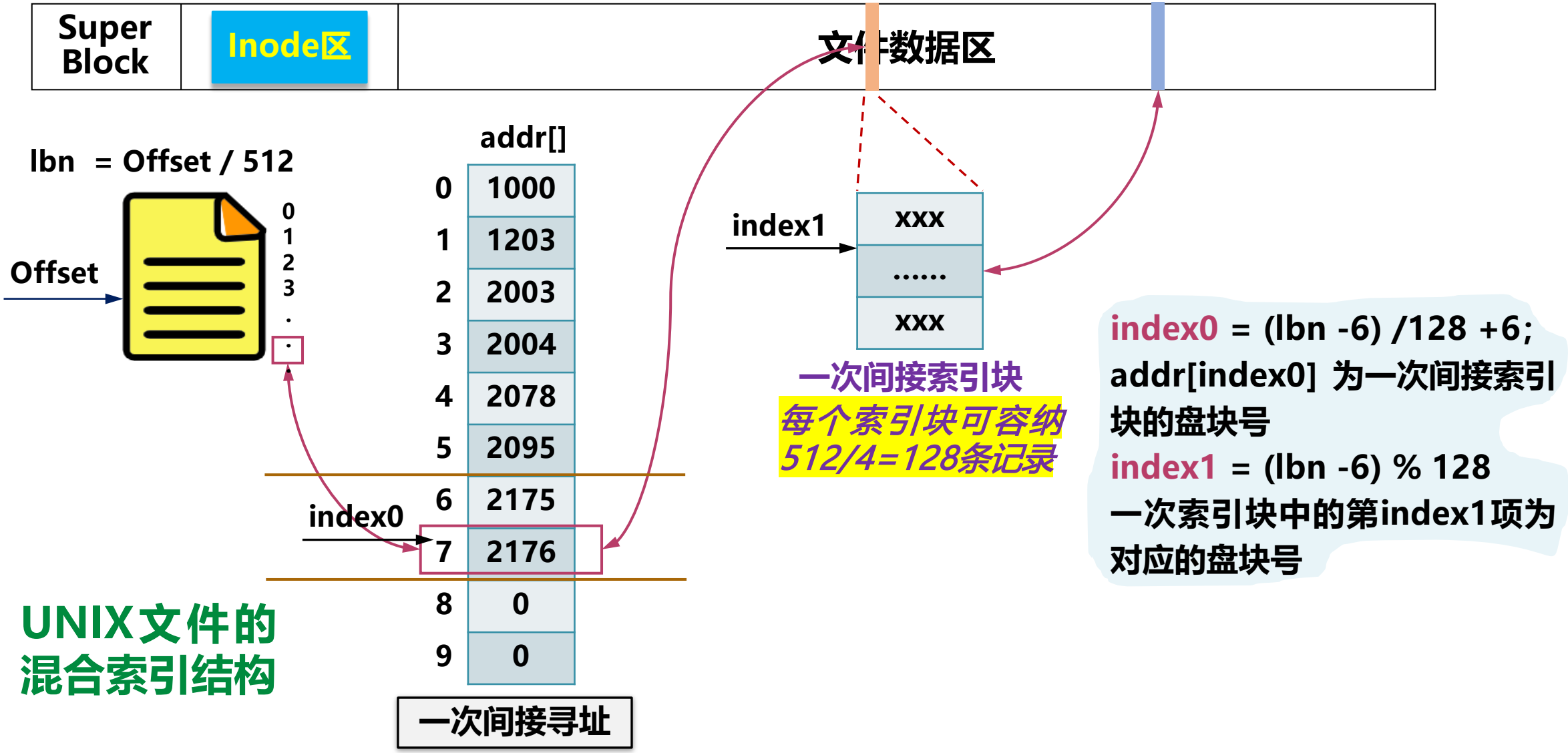
外存文件控制块区
(Inode区，202~1023#盘块)



UNIX文件的
混合索引结构

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)



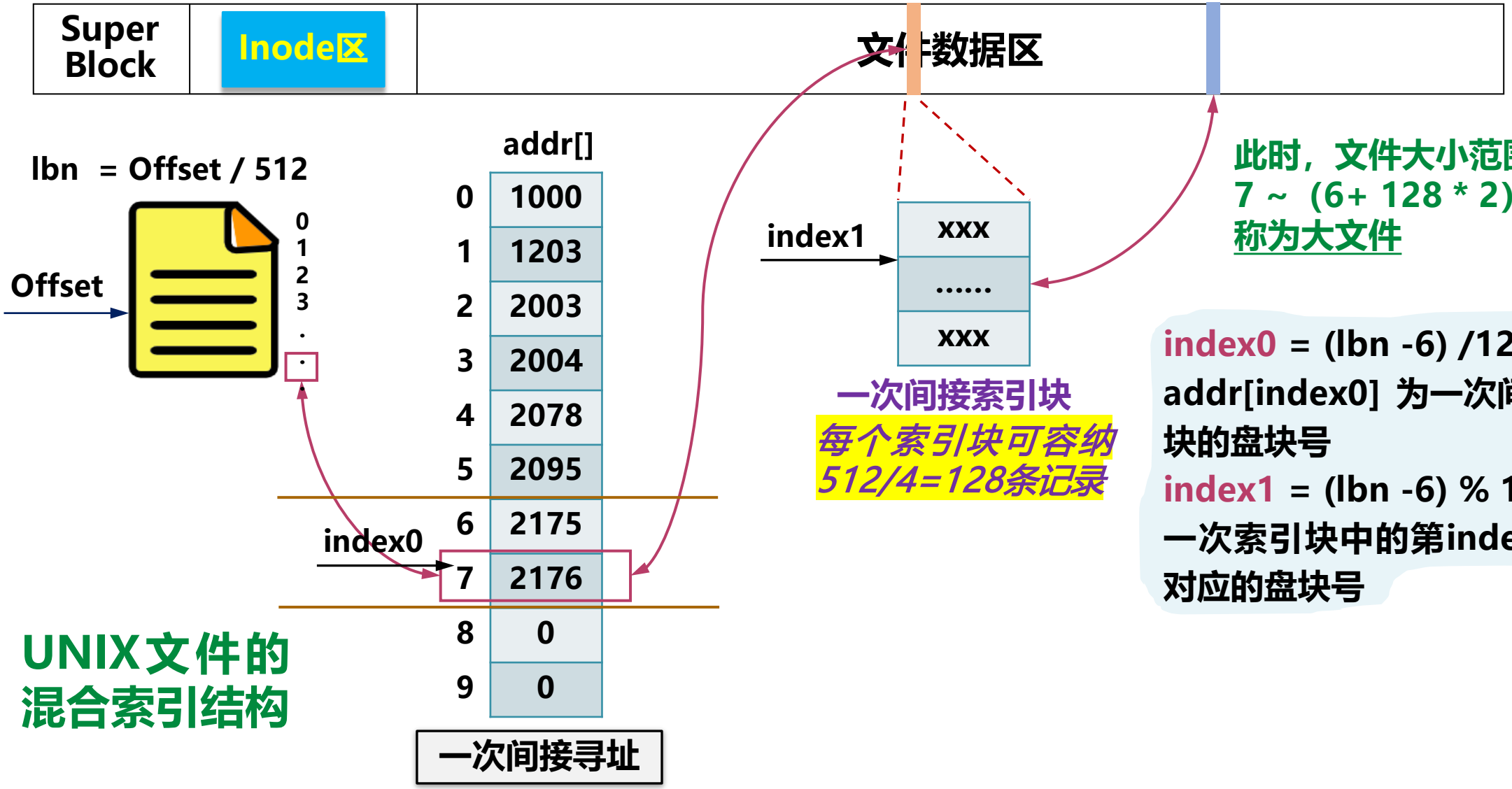
UNIX文件的混合索引结构

index0 = (lbn -6) /128 +6;
addr[index0] 为一次间接索引块的盘块号

index1 = (lbn -6) % 128
一次索引块中的第index1项为对应的盘块号

UNIX文件索引节点

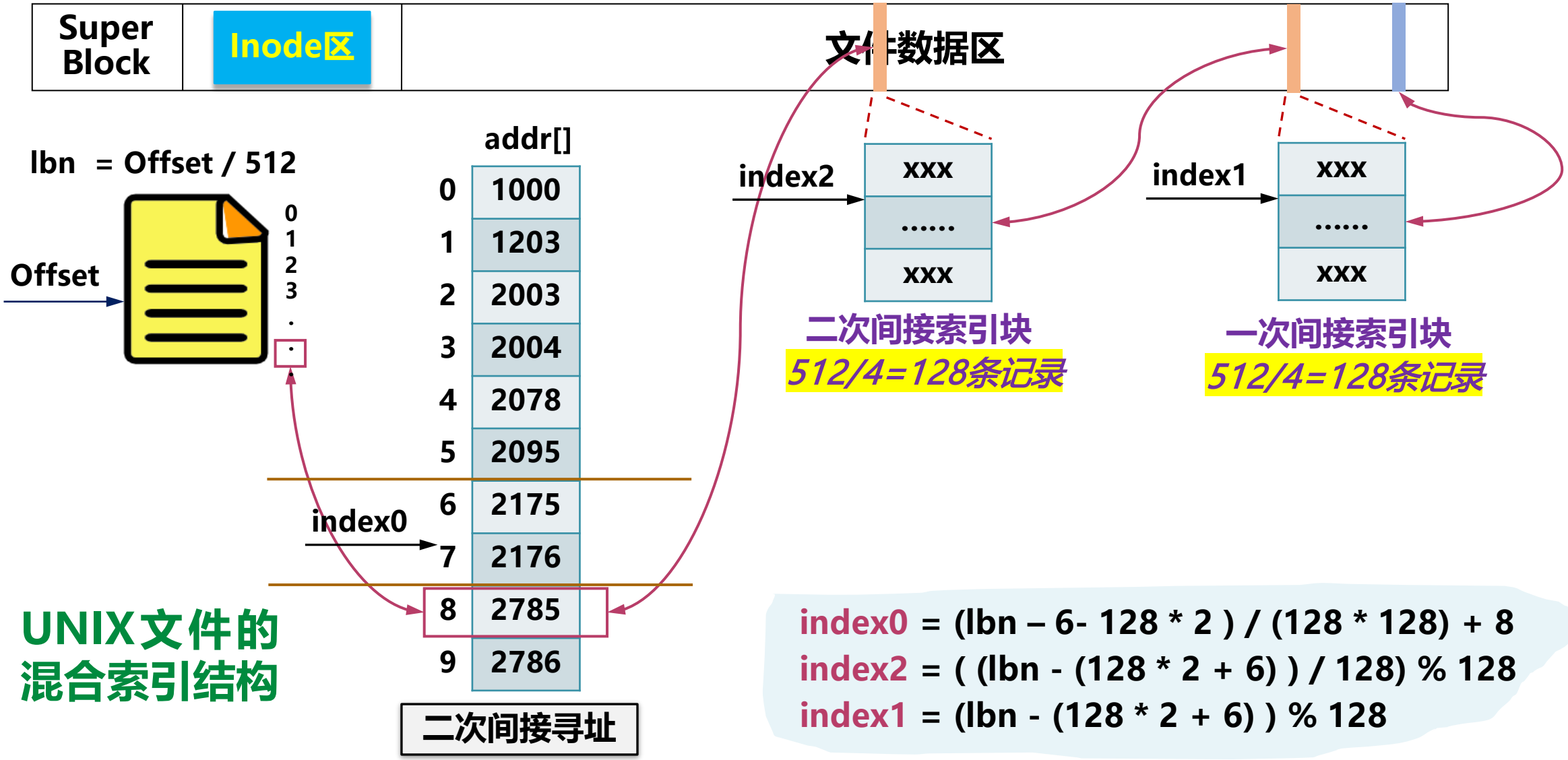
外存文件控制块区
(Inode区，202~1023#盘块)



UNIX文件的混合索引结构

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)

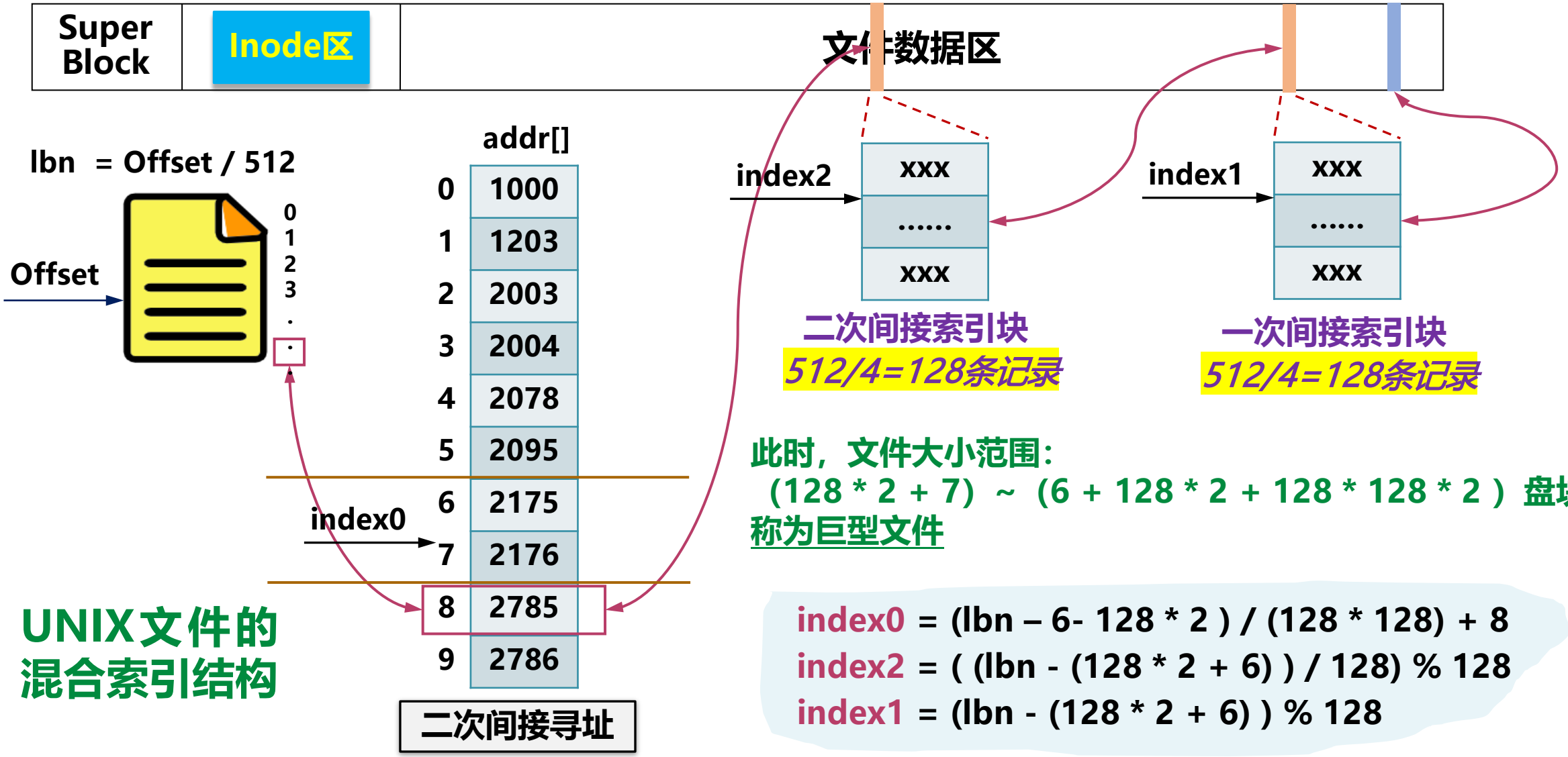


UNIX文件的
混合索引结构

index0 = (lbn – 6- 128 * 2) / (128 * 128) + 8
index2 = ((lbn - (128 * 2 + 6)) / 128) % 128
index1 = (lbn - (128 * 2 + 6)) % 128

UNIX文件索引节点

外存文件控制块区
(Inode区，202~1023#盘块)



UNIX文件的混合索引结构

此时，文件大小范围：
（128 * 2 + 7）~（6 + 128 * 2 + 128 * 128 * 2 ）盘块
称为巨型文件

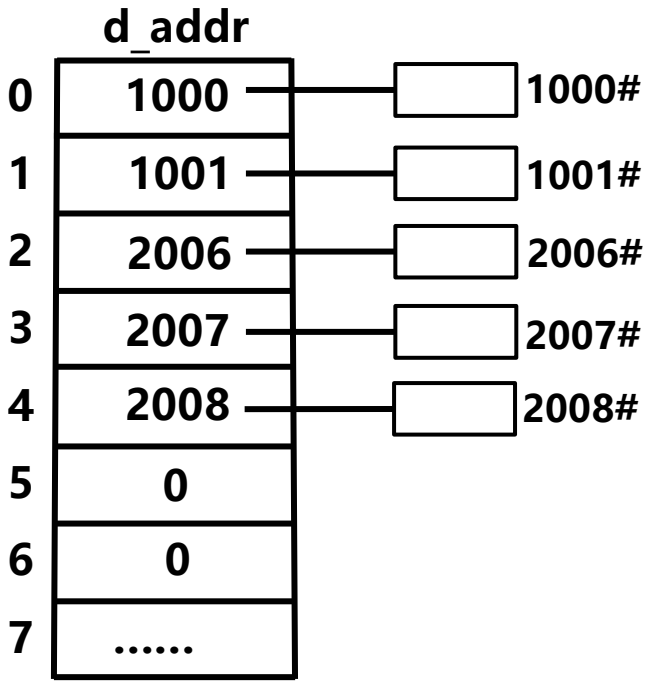
index0 = (lbn – 6- 128 * 2) / (128 * 128) + 8
index2 = ((lbn - (128 * 2 + 6)) / 128) % 128
index1 = (lbn - (128 * 2 + 6)) % 128

UNIX文件索引节点

三级索引结构

假如现在有三个文件，其大小分别为2248字节、65100字节和2M字节：

2248/512 = 4　2248%512 = 200
所以2248个字节共占5个字符块，
前4个字符块为满块，
第5字符块占用200个字节。



文件的逻辑块号lbn对应的物理块号

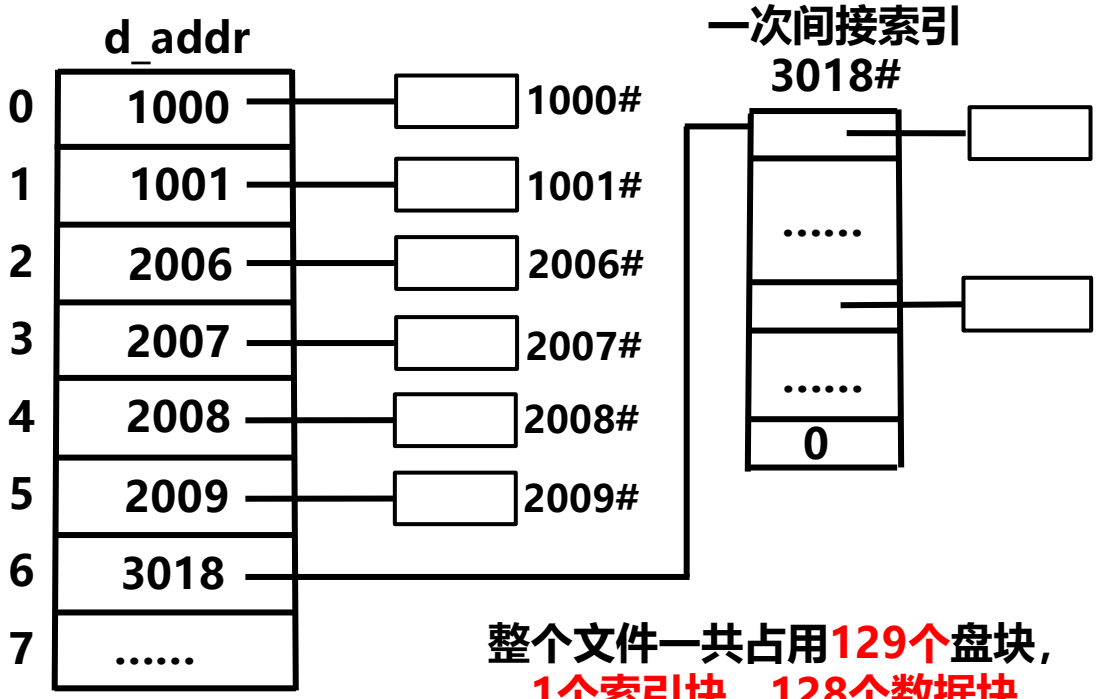
index0 = lbn;
addr[index0] 为对应的盘块号

例：lbn=3，
index0 = 3，
即：d_addr[3] = 3号逻辑块对应的盘块号

UNIX文件索引节点

假如现在有三个文件，其大小分别为2248字节、65100字节和2M字节：

65100/512 = 127　65100%512 = 76
所以65100个字节共占128个字符块，
前127个字符块为满块，
第128字符块占用76个字节。



整个文件一共占用129个盘块，
1个索引块，128个数据块

文件的逻辑块号lbn对应的物理块号

index0 = (lbn -6) /128 +6;
addr[index0] 为一次间接索引块的盘块号
index1 = (lbn -6) % 128
一次索引块中的第index1项为对应的盘块号

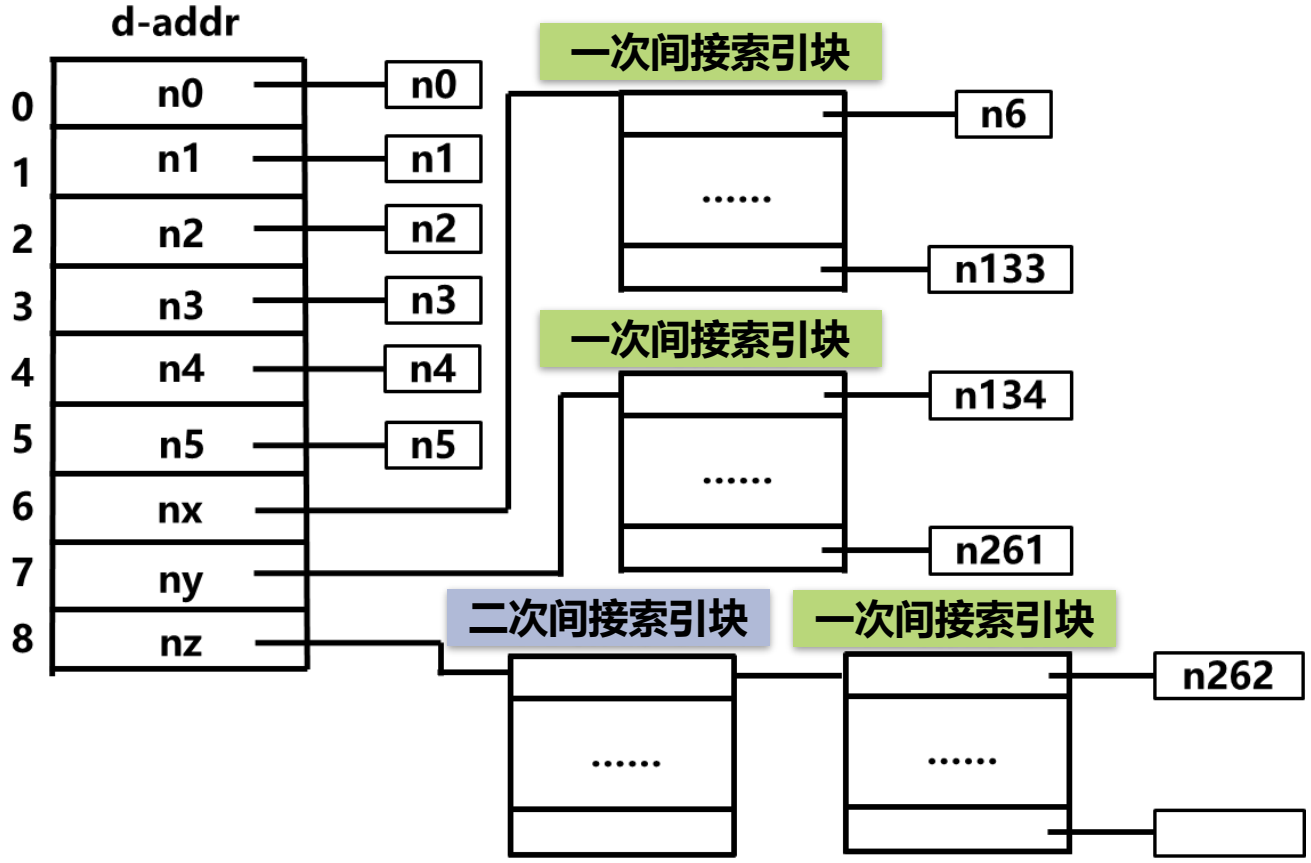
例：lbn=173
index0 = (173 - 6) / 128 + 6 = 7,
即：d_addr[7] = 一次间接索引块所在的盘块号

index1 = (173 - 6) %128 = 39,
即：在一次间接索引块中的第39项保存173号逻辑块对应的盘块号

UNIX文件索引节点

假如现在有三个文件，其大小分别为2248字节、65100字节和2M字节：

2M/512 = 4096 所以2M个字节共占4096个字符块，为巨型文件。



整个文件一共占用的盘块数：

（1）4096个数据块；

（2）前6块为直接地址，后4090个数据块需要⌈4090/128⌉=32个索引块；

（3）后30个索引块需要1个间接索引块。

整个文件共占用：

4096+32+1=4129个物理盘块

UNIX文件索引节点

三级索引结构

假如现在有三个文件，其大小分别为2248字节、65100字节和2M字节：

文件的逻辑块号lbn对应的物理块号

index0 = (lbn – 6- 128 * 2) / (128 * 128) + 8
index2 = ((lbn - (128 * 2 + 6)) / 128) % 128
index1 = (lbn - (128 * 2 + 6)) % 128

例：lbn= 1730。
Index0 = (lbn – 6- 128 * 2) / (128 * 128) +8 = 8,
即：d_addr[8] = 二次间接索引块所在的盘块号

index2 = ((1730 - (128 * 2 + 6)) /128) %128 = 11,
即：在二次间接索引块中的第11项保存一次间接索引块所在的盘块号

index1 = (1730 - (128 * 2 + 6)) %128 = 60,
即：在一次间接索引块中的第60项保存1730号逻辑块对应的盘块号

UNIX文件索引节点

存储资源管理信息块（SuperBlock，200~201#盘块）

Super Block	inode区	文件数据区

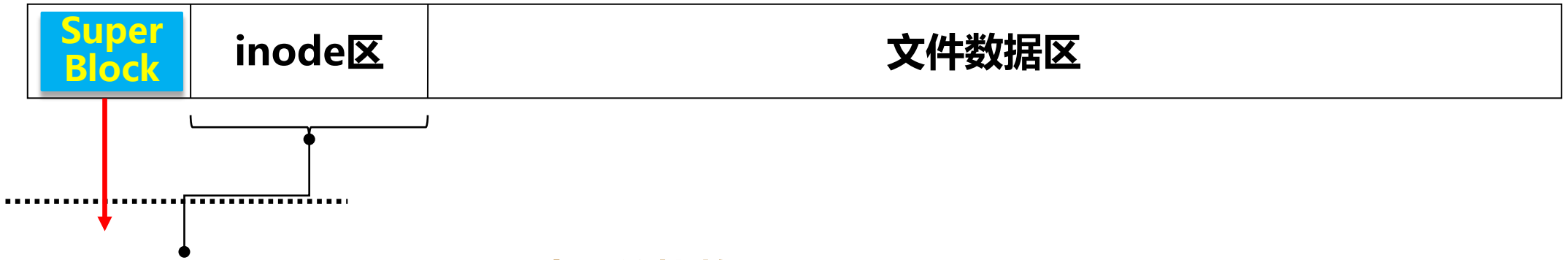
```
class SuperBlock
{
/* Functions */
public:
SuperBlock(); /* Constructors */
~SuperBlock(); /* Destructors */
/* Members */
public:

int   s_fsize;          /* 盘块总数 */
int   s_nfree;          /* 直接管理的空闲盘块数量 */
int   s_free[100];      /* 直接管理的空闲盘块索引表 */
int   s_flock;          /* 封锁空闲盘块索引表标志 */

int   s_isize;          /* 外存Inode区占用的盘块数 */
int   s_ninode;         /* 直接管理的空闲外存Inode数量 */
int   s_inode[100];     /* 直接管理的空闲外存Inode索引表 */
int   s_ilock;          /* 封锁空闲Inode表标志 */

int   s_fmod;           /* 内存中super block副本被修改标志，意味着需要更新外存对应的Super Block */
int   s_ronly;          /* 本文件系统只能读出 */
int   s_time;           /* 最近一次更新时间 */
int   padding[47];      /* 填充使SuperBlock块大小等于1024字节，占据2个扇区 */
};
```

对INODE区的管理

SuperBlock占用两个盘块，一共1024个字节

UNIX文件索引节点

存储资源管理信息块（SuperBlock，200~201#盘块）



s_isize; ← inode区占用的块数

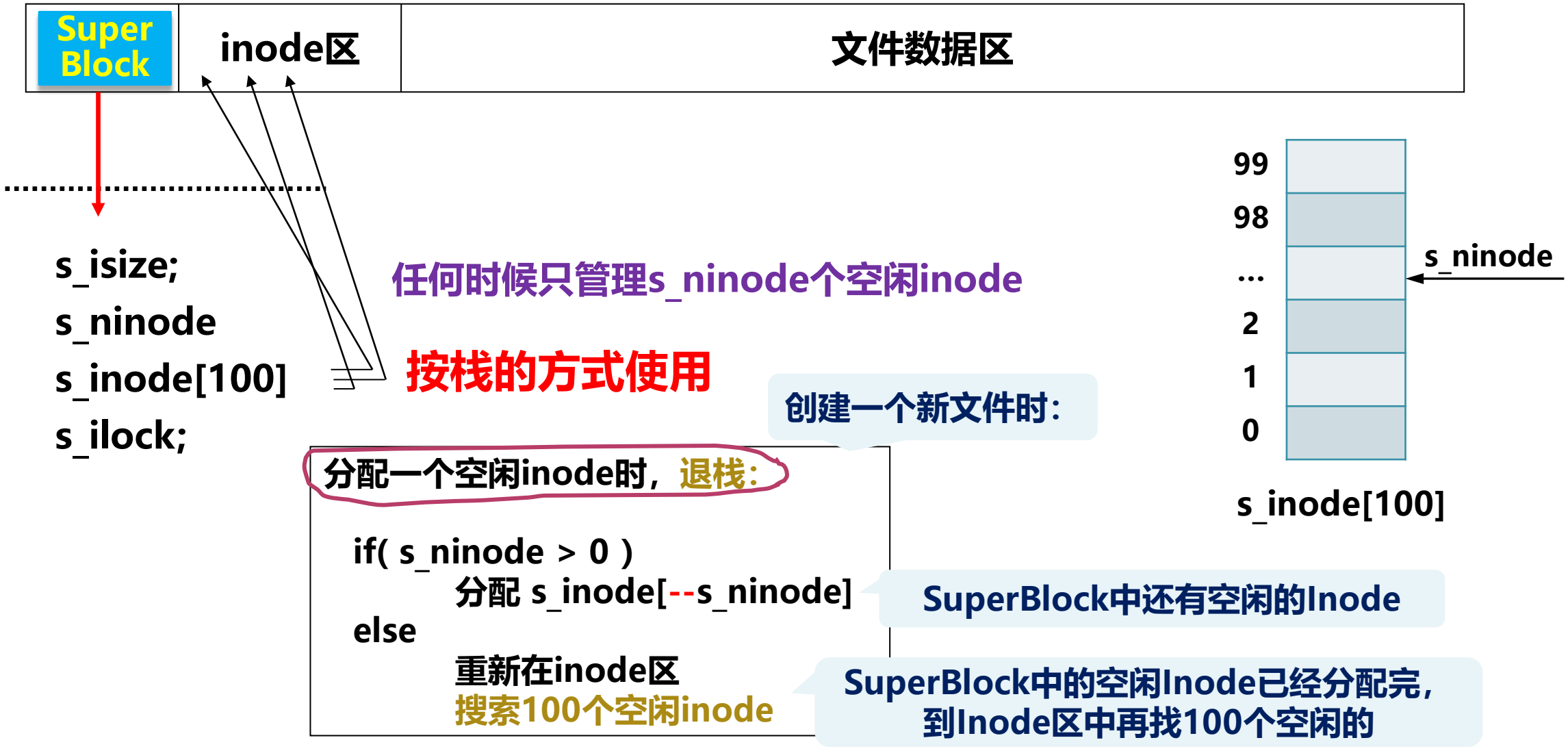
s_ninode ← SuperBlock直接管理的空闲inode个数 ≤100

s_inode[100] ← SuperBlock直接管理的空闲inode号索引表

s_ilock; ← 临界区锁

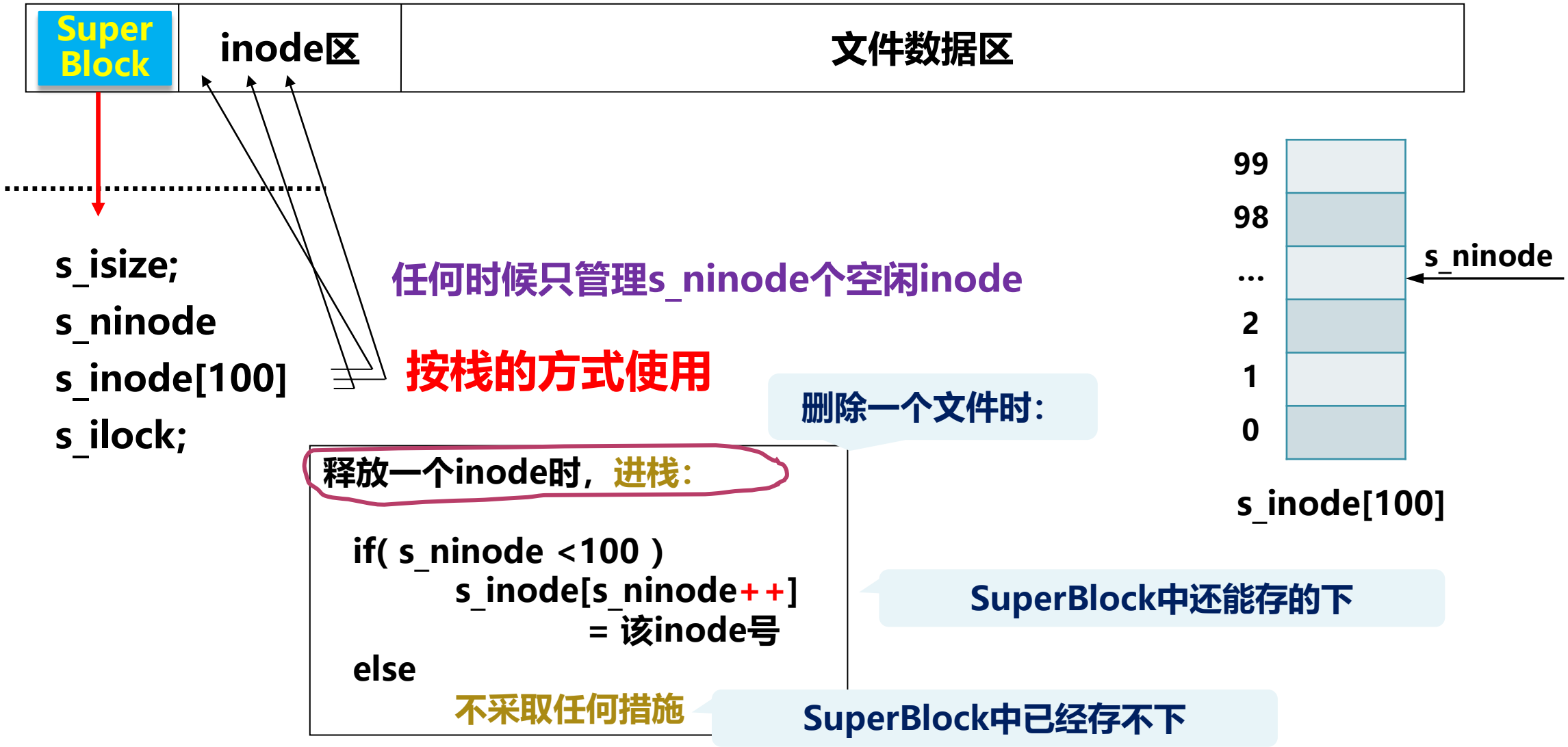
UNIX文件索引节点

存储资源管理信息块（SuperBlock，200~201#盘块）



UNIX文件索引节点

存储资源管理信息块（SuperBlock，200~201#盘块）



本节小结

1　了解文件的逻辑结构与物理结构

2　掌握几种文件物理结构的特征和优缺点

3　熟悉UNIX文件系统的物理结构

阅读教材：262页 ~ 278页

E17：文件管理（UNIX文件系统的物理结构）